Learning the dynamics of multispectral satellite image time series

Anthony Frion

COLLEGES BRETAGNE LOIRE | SCIENCES POUR L'INGENIEUR ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE BRETAGNE PAYS DE LA LOIRE – IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Signal, image et vision*

Par
Anthony FRION

Apprentissage de dynamiques dans les séries temporelles d'images satellites multispectrales

Thèse présentée et soutenue à IMT Atlantique, Brest, le 3 décembre 2024
Unité de recherche : Département Mathematical and Electrical Engineering, Lab-STICC, UMR CNRS 6285
Thèse N° : 2024IMTA0435

Rapporteurs avant soutenance :

Florence TUPIN	Professeur, Télécom Paris
Julien LE SOMMER	Directeur de recherche CNRS, Institut des Géosciences de l'Environnement

Composition du Jury :

Président :	Ronan FABLET	Professeur, IMT Atlantique
Examinateurs :	Ricardo BORSOI	Chargé de recherche CNRS, Université de Lorraine
	Julien LE SOMMER	Directeur de recherche CNRS, Institut des Géosciences de l'Environnement
	Etienne MÉMIN	Directeur de recherche Inria, Inria Rennes
	Charlotte PELLETIER	Maître de conférences, Université Bretagne Sud
	Florence TUPIN	Professeur, Télécom Paris
Dir. de thèse :	M. Abdeldjalil AÏSSA EL BEY	Professeur, IMT Atlantique
Co-encadrant :	Lucas DRUMETZ	Maître de conférences, IMT Atlantique

Invités :

Guillaume TOCHON	Maître de conférences, EPITA
Mauro DALLA MURA	Maître de conférences HDR, Grenoble INP

Acknowledgements

Although I theoretically started working on this thesis 3 years ago, I somehow find it hard to mentally accept that so much time has passed since then. I was 23 years old and I am now 26, which means that this thesis has lasted a little bit more than 10% of the duration of my life. And yet it felt short, since I just had so much fun, both in the scientific and non-scientific aspects.

I obviously did not make this journey alone, and there are plenty of people whom I need to thank for contributing (in various ways) to this thesis.

First, I would like to thank the people that enabled me to start this journey. So, thank you to Nicolas and Minh-Tan for accompanying me through my first research experience during an internship in 2021, just before I started my PhD. Thanks also for recommending me to Lucas for his PhD subject (although you clearly exaggerated on my skills). And thanks to the other people at IMT Atlantique who helped me to better understand the kind of PhD that I was looking for: Elsa, Raphaël and Vincent among others.

Thank you, of course, to my supervisors Lucas, Guillaume, Mauro and Djalil who accompanied me during this scientific journey, with particular thanks to Lucas who designed the project from the beginning.

Thank you to Tev and Clément, for joining me in the last months of this journey. I hope you will keep enjoying research.

Thank you to all the people who made the lab a great place to work during these 3 years. This includes, but is not limited to: Daniel, Ewen, Daria, Ismaila, Oscar, Oscar, Ahcehn, Elsa, Lucas, Simon, Tev, Clément, Amélie, Erwan, Parth, Alexandre, Ilyass, Camila. In particular, thank you to the tea drinkers: Daniel, Tev, Solène, Ismaila. And thank you to the chess players: Ewen, Oscar and Tev.

Thank you to people from the lab with whom I had a great time going to conferences: Lucas, Daria, Yassir, Sarah, Nicolas, Bastien and others.

Concerning aspects not directly related to work now, thank you to the people with whom I played a lot of table-top RPG and murder parties (this is legal I promise) during these 3 years: Mari Lena, Emilie, Tiban, Ambre, Tev, Sarah, Gweltaz, Aude, Maxine, Morgane, FM, Lucie, Guillaume, Chapoline, Antoine, Jules.

Thank you to the people who I can usually be found hanging out with in various creperies and tea salons in Brest: FM, Aude, Gweltaz, Bruno, Emilie, Hélène, Mari Lena, Lucie, and others.

Thank you to Hélène, Mehdi and Bastien for the great times playing Spirit Island together.

Thank you to my bouldering buddies: FM, Aude, Antoine, Gweltaz, Emilie, Bastien, Estelle, Lorène, Pascal, Hélène, Oscar and others.

Many thanks to my supervisors again for proofreading the present manuscript and spotting so many mistakes and potential improvements. I am obviously the one to blame for the mistakes that still remain.

This whole work was made possible by the Agence Nationale de la Recherche, under grant ANR-21-CE48-0005 LEMONADE.

My final thanks go to the members of my jury, who have kindly accepted to take interest in my work, and to evaluate its relevance. Special thanks go to the rapporteurs, who have accepted to read and evaluate the present manuscript.

Résumé en français

Contexte

Au cours des dernières décennies, et singulièrement au cours des dernières années, d'importantes quantités de données ont été acquises par des satellites d'observation terrestre. Ces satellites ont été lancés dans le cadre de programmes scientifiques tels que le programme Copernicus de l'Union européenne, qui permet un accès libre à toutes les données collectées. Ces données se présentent sous la forme d'images correspondant à une certaine partie de la surface de la Terre.

Dans notre cas, on s'intéressera plus particulièrement aux satellites d'observation optique multispectrale, c'est-à-dire ceux qui acquièrent de l'information optique dans plusieurs bandes spectrales outre les trois bande correspondant aux couleurs perçues par l'œil humain. Ces bandes spectrales additionnelles sont généralement situées dans le domaine de l'infrarouge, qui permet entre autres l'acquisition d'informations utiles à l'analyse de la végétation, par exemple au travers de l'indice de végétation par différences normalisées (NDVI).

Par ailleurs, un aspect important du travail ici présenté est que l'on s'intéressera principalement aux aspects dynamiques des données acquises par les satellites d'observation. Concrètement, on s'intéressera à des séries temporelles d'images satellites (satellite image time series, SITS), que l'on constituera en réunissant de nombreuses images consécutives d'une même zone, avec les dates d'acquisition associées. Dans ce cadre, le temps de revisite du satellite d'observation, c'est-à-dire le temps qui sépare 2 acquisitions d'une même zone sur la surface de la Terre, sera un paramètre important. Dans le cadre des expériences présentées dans le manuscrit, on utilisera les données issues de la constellation Sentinel-2, qui comprend 2 satellites en opposition de phase permettant, à eux 2, un temps de revisite de 5 jours à l'équateur.

Notre approche : apprentissage auto-supervisé

Ainsi, l'objectif principal de cette thèse est de modéliser la dynamique associée aux séries temporelles d'images satellites Sentinel-2, dans un premier temps au niveau d'un pixel seul. Or, à notre connaissance, il n'existe pas de modèle analytique suffisamment riche pour modéliser une multitude de matériaux différents tels que l'herbe, l'eau, les environnements urbains et les différents types d'arbres et de cultures, dont on s'attend à ce qu'ils aient tous un comportement dynamique différent. De plus, selon les hypothèses classiques du problème de démélange spectral, on s'attendra à ce qu'un pixel soit composé en pratique d'un mélange de plusieurs de ces matériaux, dont l'effet induit sur la dynamique pourrait ne pas être linéaire. Pour ces raisons, on a recours dans ce travail de thèse à des méthodes d'apprentissage automatique, moins interprétables mais plus expressives que les méthodes classiques.

En outre, il convient de préciser dans quel paradigme de l'apprentissage on travaille. Le paradigme le plus courant est l'apprentissage supervisé, qui consiste à entraîner un modèle à réaliser une tâche précise, supposément matérialisée par une relation fonctionnelle entre une entrée et la sortie associée, en utilisant un jeu de données composé d'un certain nombres de paires entrée-sortie. Ce paradigme d'apprentissage est très efficace dès lors que le jeu de données, l'architecture choisie pour le modèle et les hyperparamètres de l'entraînement permettent d'obtenir un modèle final qui parvient à généraliser à des exemples hors du jeu de données d'entraînement, représentant de nouveaux exemples de résolution de la tâche choisie. Néanmoins, l'acquisition de données d'entraînement en qualité et quantité suffisantes peuvent être problématiques en termes de difficulté technique ou de coût financier. Par ailleurs, un modèle ainsi entraîné sera généralement spécialisé dans la tâche d'entraînement choisie et pourra difficilement être généralisé à une nouvelle tâche correspondant, généralement, à une nouvelle relation fonctionnelle dont l'entrée est la même que pour la tâche d'entraînement.

Dans le cas des séries temporelles d'images satellites, comme mentionné plus tôt, il existe une très grande quantité de données brutes acquises depuis des années. Néanmoins, l'obtention d'annotations pour résoudre des tâches utiles en pratique telles que la segmentation de cultures peut poser des difficultés. Ainsi, ce cadre semble bien se prêter à un autre paradigme d'apprentissage : l'apprentissage auto-supervisé. L'apprentissage auto-supervisé vise à apprendre un modèle qui comporte une représentation des données suffisamment générale pour résoudre diverses tâches différentes. Ainsi, plutôt que d'entraîner ce modèle à résoudre une tâche spécifique nécessitant des annotations pour avoir des exemples de la relation fonctionnelle correspondante, on entraîne le modèle à résoudre une tâche dite "prétexte", qui ne nécessite pas d'annotations des données mais pour laquelle on peut créer des données d'entraînement directement à partir des données brutes. Concrètement, dans notre cas, la tâche prétexte est la prédiction de long terme des séries temporelles d'images satellites. Ainsi, des jeux de données d'entraînement peuvent être créés simplement en associant des états initiaux du vecteur de réflectance correspondant à un pixel à un temps donné (l'entrée) à la série temporelle qui suit cette condition initiale (la sortie). L'hypothèse sous-jacente à ce choix est qu'un modèle capable de prédire la dynamique d'un pixel de données sur le long terme à partir d'une condition initiale comporte nécessairement une information suffisamment générale sur les données pour résoudre différentes tâches dites "aval" sur ces séries temporelles : interpolation, débruitage, détection d'anomalies, etc.

Prédiction de long terme avec l'opérateur de Koopman

Pour mettre en œuvre cette stratégie, on a réuni des données de séries temporelles d'images satellites correspondant à deux zones : la forêt de Fontainebleau et la forêt d'Orléans, en France. Le fait de travailler avec ces deux zones nous permet d'entraîner nos modèles sur une seule d'entre elles puis de tester la capacité de généralisation de ce modèle sur la deuxième zone, dont les dynamiques peuvent différer du fait notamment de matériaux différents, dont les mélanges

peuvent également différer. Par défaut, les séries temporelles d'images Sentinel-2 sont lacunaires puisqu'une grande quantité d'informations est masquée par les nuages situés entre le satellite et la surface de la Terre. Ainsi, on a proposé d'utiliser la méthode classique d'interpolation de Cressman pour obtenir, à partir de ces données irrégulièrement échantillonnées dans le temps, des données régulièrement échantillonnées mais partiellement synthétiques. Les jeux de données correspondant aux zones de Fontainebleau et Orléans ont donc été tous deux mis à disposition publiquement pour la communauté scientifique, dans une version incomplète mais exacte et dans une version complète mais partiellement corrompue.

En ce qui concerne les architectures utilisées pour répondre au problème d'apprentissage auto-supervisé que l'on a posé, nous nous sommes intéressés à la théorie de l'opérateur de Koopman. Cette théorie stipule que tout système dynamique peut être décrit par un opérateur linéaire, généralement de dimension infinie, qui s'applique à un ensemble de fonctions d'observations de ce système. La théorie de l'opérateur de Koopman, introduite dans les années 1930, a connu un renouveau dans les dernières décennies avec l'introduction de nombreuses méthodes qui s'en inspirent pour modéliser les systèmes dynamiques. La plus connue de ces méthodes est la décomposition en mode dynamiques, qui est décrite en détails dans le manuscrit. Pour notre part, nous travaillons avec une classe de méthode qui consiste à apprendre un auto-encodeur neuronal, dont l'encodeur représente un ensemble fini de fonctions d'observation que l'on suppose (approximativement) invariant par l'opérateur de Koopman du système modélisé, et dont l'encodeur permet de repasser de cet encodage vers l'espace d'état du système. L'idée sous-jacente est que l'espace latent de cet auto-encodeur permet de modéliser la dynamique du système de manière linéaire. Ainsi, entre l'encodeur et le décodeur, on insère une matrice (dont les coefficients sont des paramètres du modèle, appris conjointement avec ceux de l'encodeur et du décodeur) qui représente la restriction de l'opérateur de Koopman à l'ensemble de fonctions d'observation défini par l'encodeur. En d'autres termes, cette matrice représente la dynamique linéaire du système dans l'espace latent de l'auto-encodeur. Nous la désignerons par la suite comme la "matrice de Koopman".

Cette architecture, que l'on nomme l'auto-encodeur de Koopman, a déjà été étudiée dans un certain nombre de publications scientifiques avant le début de la thèse. Néanmoins, nous y apportons des contributions nouvelles, qui se présentent principalement sous deux formes. Premièrement, nous montrons qu'en plus des termes classiquement utilisés dans la fonction de coût pour l'entraînement d'un tel modèle, un terme favorisant l'orthogonalité de la matrice de Koopman permet d'améliorer la stabilité sur le long terme des prévisions réalisées par le modèle, en garantissant notamment la quasi-constance de la norme des vecteurs d'état latents avec le temps de prédiction. Deuxièmement, nous avons montré qu'il était possible d'obtenir un modèle continu sous-jacent à l'auto-encodeur de Koopman en utilisant le logarithme matriciel de la matrice de Koopman. De cette manière, il est possible d'obtenir une prédiction en temps continu,

ce qui permet d'une part de réaliser des prédictions à haute fréquence temporelle même si les données d'entraînement sont échantillonnées à basse fréquence, et d'autre part d'entraîner un modèle à partir de données échantillonnées irrégulièrement dans le temps, y compris avec les paternes d'échantillonnage les plus complexes.

La pertinence de ces deux contributions a été vérifiée au travers d'une série d'expériences réalisées sur des données synthétiques et réelles de différents systèmes. En ce qui concerne les données synthétiques, on a travaillé sur un système dynamique en 3 dimensions issu de la mécanique des fluides, sur le système du pendule simple et sur une simulation de séries temporelles d'images satellites de végétation basées sur le modèle PROSAIL. En ce qui concerne les données réelles, on a travaillé sur les séries temporelles d'images satellites Sentinel-2 des données de Fontainebleau et Orléans mentionnées précédemment.

Pour ce qui est du terme d'orthogonalité dans la fonction de coût, on a notamment vérifié que l'utilisation de ce terme permettait effectivement d'améliorer les performances de long terme sur données réelles, à la fois en extrapolation temporelle sur la zone de Fontainebleau utilisée pour l'entraînement et en extrapolation spatiale sur la zone d'Orléans utilisée pour la validation. On a par ailleurs montré que le modèle d'auto-encodeur de Koopman avec notre fonction de coût réalisait de meilleures prédictions que le très classique modèle d'apprentissage Long Short-Term Memory (LSTM) et qu'un modèle linéaire de décomposition en modes dynamiques entraîné à la prédiction de long terme.

En ce qui concerne le modèle continu sous-jacent de l'auto-encodeur de Koopman, on a entraîné ce modèle sur des données de pendule simple et de mécanique des fluides échantillonnées à basse fréquence et montré que, en utilisant la formulation continue du modèle, on parvenait à obtenir des prédictions à haute fréquence qui correspondent effectivement à la réalité. Par ailleurs, on a montré que l'auto-encodeur de Koopman était en mesure, grâce à cette formulation continue, d'apprendre la dynamique de séries temporelles d'images satellites, à la fois dans un cas où les données sont régulièrement échantillonnées avec des points de données manquants (données réelles) et dans un cas plus général où le paterne d'échantillonage temporel est complètement irrégulier (données synthétiques).

Résolution de tâches aval par l'assimilation de données

Après avoir décrit nos contributions au modèle d'auto-encodeur de Koopman, et par là-même notre tâche prétexte pour l'entraînement de ce modèle dans un cadre d'apprentissage auto-supervisé, nous décrivons différentes méthodes pour utiliser un modèle ainsi pré-entraîné comme un a priori dynamique dans un problème d'assimilation de données variationnelle, afin de résoudre des tâches aval telles que l'interpolation temporelle, le débruitage de séries temporelles et la prédiction de long terme à partir de plusieurs (et non plus une seule) observations. Le modèle d'auto-encodeur de Koopman étant différentiable via la règle de la chaîne (ce qui a permis son entraînement), il est possible de l'inclure dans un coût variationnel que l'on minimise via un outil

de différentiation automatique tel que Pytorch. En ce sens, on a proposé différentes formulations de coûts variationnels, basées sur des formes contraintes et non contraintes d'assimilation de données.

Dans le cas de l'assimilation non contrainte, la variable sur laquelle porte le coût variationnel est directement la série temporelle obtenue par la méthode. Le coût variationnel est alors composé de deux termes : un terme d'attache aux données observées et un terme de fidélité à l'a priori dynamique, c'est-à-dire à l'auto-encodeur de Koopman pré-entraîné pour la prédiction de long terme sur des données de la forêt de Fontainebleau. Ce terme de fidélité au modèle est en l'occurrence calculé sur le court terme : pour chaque paire d'états consécutifs de la série temporelle optimisée, on promeut la proximité entre l'état suivant et la prédiction après un pas de temps réalisée par l'a priori dynamique à partir de l'état précédent.

En ce qui concerne l'assimilation contrainte, la variable sur laquelle porte le coût variationnel n'est plus directement la série temporelle obtenue en sortie mais une variable auxiliaire qui paramètre cette série temporelle. En l'occurrence, le coût variationnel porte sur la condition initiale latente du modèle utilisé comme a priori dynamique. Ainsi, la série temporelle est simplement prédite en réalisant une prédiction de long terme via le modèle à partir de cette condition initiale résultant de l'optimisation. De ce fait, la série temporelle obtenue est directement produite par le modèle dynamique pré-entraîné, d'où le terme d'assimilation "contrainte" qui signifie que la fidélité au modèle est assurée par la contrainte et non plus par un terme "objectif" du coût variationnel.

Comme expliqué qualitativement et quantitativement via une série d'expériences sur les séries temporelles d'images Sentinel-2, les assimilations contrainte et non contraintes comportent des avantages différents. En effet, l'assimilation contrainte, du fait qu'elle résulte nécessairement en un résultat directement produit par le modèle dynamique a priori, est moins expressive que l'assimilation non contrainte, mais on s'attend à ce que le résultat produit corresponde toujours à une trajectoire plausible du système dynamique étudié. Ainsi, l'assimilation contrainte permet d'obtenir de meilleures performances en débruitage lorsque l'amplitude du bruit sur les données observées est relativement forte, tandis que que l'assimilation non contrainte permet de meilleures performance pour les amplitudes de bruit faible. En effet, le modèle dynamique pré-entraîné comporte nécessairement une erreur de reconstruction des données observées. Or, dès lors que cette erreur de reconstruction est d'amplitude plus élevée que celle du bruit associé à ces données observées, le résultat produit par l'assimilation contrainte sera moins fidèle à la réalité que les observations elles-mêmes. L'assimilation non contrainte, plus flexible, permettra de réaliser de légers ajustements des données observées pour se rapprocher légèrement de l'a priori dynamique, et donc supposément de la trajectoire réelle des données.

Les performances de nos méthodes d'assimilation contrainte et non contraintes ont été démontrées pour la réalisation de trois tâches sur les séries temporelles d'images Sentinel-2 : le

débruitage, la prédiction de long terme à partir de plusieurs observations et l'interpolation. Pour ce qui est du débruitage, on a montré que l'assimilation non contrainte était effectivement efficace pour les amplitudes de bruit faibles tandis que l'assimilation contrainte l'est davantage pour les amplitudes de bruit fortes. Dans les deux cas, on a montré la supériorité de ces méthodes vis-à-vis de l'interpolation de Cressman, une méthode relativement efficace pour traiter les bruits gaussiens mais agnostique à la spécificité du système dynamique étudié.

En ce qui concerne la prédiction de long terme à partir de plusieurs observations, on a repris l'ensemble de modèles utilisés pour résoudre la tâche prétexte. On a ainsi montré que le modèle d'auto-encodeur Koopman entraîné avec le terme d'orthogonalité dans sa fonction de coût était un meilleur a priori dynamique que le même modèle entraîné sans terme d'orthogonalité et que le LSTM et la décomposition en modes dynamiques. Dans ce cadre, on a utilisé une méthode d'assimilation contrainte, plus appropriée pour extrapoler au-delà des données observées.

Enfin, pour l'interpolation, on a cette fois comparé notre méthode avec une autre méthode d'interpolation contrainte consistant à trouver le paterne d'évolution périodique qui correspond le mieux aux données observées. On a montré que, bien que cette méthode utilisée comme point de comparaison soit efficace sur les données qui nous intéressent dans la mesure où elles comportent une composante périodique importante, notre méthode qui s'appuie sur un modèle dynamique appris à partir des données permet tout de même d'obtenir de meilleurs performances.

Extensions de nos méthodes pour des modèles stochastiques

Dans la dernière partie de ce manuscrit de thèse, nous proposons des extensions des méthodes présentées précédemment pour créer des modèles stochastiques. En quelques mots, les modèles stochastiques se distinguent par le fait que les prédictions qu'ils réalisent se présentent sous la forme de distributions de probabilité non triviales, tandis que les prédictions d'un modèle classique sont ponctuelles, c'est-à-dire déterministe. Autrement dit, plutôt que de prédire une sortie ponctuelle $\mathbf{y}$ associée à l'entrée $\mathbf{x}$, un modèle stochastique prédira une distribution de probabilité pour $\mathbf{y}$, considérée comme une variable aléatoire, sachant $\mathbf{x}$. Les modèles stochastiques sont particulièrement utiles pour modéliser des relations qui sont elles-mêmes stochastiques, c'est-à-dire des systèmes pour lesquels la connaissance de la variable $\mathbf{x}$ ne permet pas de déterminer avec certitude la valeur de $\mathbf{y}$. Ils permettent par ailleurs de prendre en compte une incertitude portant directement sur la variable d'entrée $\mathbf{x}$. Ainsi, un modèle stochastique permet de réaliser facilement une quantification de l'incertitude associée à une prédiction, puisque la variabilité de la distribution de sortie représente une indication de l'incertitude du modèle. Nous nous intéresserons particulièrement au *continuous ranked probability score*, ou CRPS. En outre, en travaillant par exemple avec des intervalles de confiance issus de la distribution de probabilité prédite par le modèle, on peut s'intéresser à la tâche de détection d'anomalies, c'est-à-dire chercher à prédire des occurrences de phénomènes inhabituels conduisant à des écarts importants

entre certaines observations, qualifiées d'anomalies, et le comportement moyen observé dans les données.

Pour obtenir des modèles stochastiques, nous proposons dans un premier temps de travailler avec des ensembles d'auto-encodeurs de Koopman déterministes. Ainsi, la distribution de probabilités prédite par un tel ensemble de modèles est simplement une somme de Dirac, supposés équiprobables et chacun correspondant à la prédiction de l'un des membres de l'ensemble de modèles. Bien qu'une telle distribution de probabilité reste très simple et ne permette pas, avec un petit nombre de membres, de calculer efficacement des intervalles de confiance, elle permet néanmoins d'estimer l'incertitude associée à une entrée donnée, simplement via l'écart-type des prédictions réalisées. Or, en ce qui concerne la quantification d'incertitudes, il est attendu d'un modèle stochastique efficace que la variance de ses prédictions soit proche de l'erreur quadratique qu'il réalise en moyenne. Cela est du au fait que cette relation entre la variance et l'erreur des prédictions est vérifiée par la distribution de probabilité réelle que suivent les données. Lorsque la variance d'un modèle stochastique est trop faible vis-à-vis de ses erreurs quadratiques, on dit que le modèle est "sur-confiant" (overconfident). Or, il est montré dans le manuscrit que, lorsque les auto-encodeurs de Koopman composant un ensemble de modèles sont entraînés indépendamment les uns des autres, selon la méthode classiquement utilisée pour entraîner des ensembles de réseaux de neurones, l'ensemble résultant est sur-confiant. Ainsi, pour obtenir plus de variance dans les prédictions, on propose d'entraîner les membres de l'ensemble conjointement en encourageant directement la variance inter-membres dans la fonction de coût. Pour ce faire, on introduit différents termes de promotion de la variance, avec un coefficient ajustable dont on montre les valeurs acceptables et l'effet sur la confiance de l'ensemble entraîné. On analyse ainsi précisément les performances de l'ensemble en termes de CRPS, ainsi que via les diagrammes spread-skill, en fonction du coefficient choisi. Les résultats présentés montrent qu'un ensemble dont les membres sont entraînés conjointement et encouragés à présenter une variance élevée permettent une meilleure calibration de l'incertitude.

Comme mentionné précédemment, les ensembles de modèles déterministes permettent d'estimer les incertitudes en fonction de l'entrée observée, mais renvoient une distribution de probabilité très simple qui permet difficilement de calculer, par exemple, des intervalles de confiance. En outre, le temps de calcul de ces ensembles, aussi bien pendant l'entraînement que pendant l'inférence, est conséquent. Pour ces raisons, nous présentons également un nouveau modèle d'auto-encodeur de Koopman qui permet d'obtenir directement une distribution de probabilité en sortie, sans avoir besoin d'entraîner plusieurs modèles en parallèle. Pour ce faire, nous nous inspirons des auto-encodeurs variationnels, qui sont des modèles génératifs populaires dans le domaine de l'apprentissage machine. Le principe est que l'encodage d'une entrée ponctuelle n'est pas lui-même un vecteur ponctuel mais une distribution de probabilité gaussienne, définie par sa

moyenne et sa matrice de covariance (généralement diagonale). On définit ainsi dans un premier temps un auto-encodeur de Koopman variationnel.

En outre, de manière à faciliter les calculs de vraisemblance effectués via le modèle tout en garantissant une reconstruction exacte des états d'entrée, nous remplaçons le réseau de neurones quelconque qui calcule la moyenne de la distribution gaussienne latente par un flot normalisant. Le flot normalisant est un autre modèle génératif classique en apprentissage machine, qui permet d'établir un lien entre une distribution de probabilités simple (généralement gaussienne) et une distribution plus complexe, via une transformation analytiquement inversible, dont le jacobien peut être calculé facilement en pratique. Ainsi, le modèle résultant est un auto-encodeur de Koopman variationnel inversible. Or, pour que ces propriétés soient vérifiées, il est nécessaire que la dimension de l'entrée du flot normalisant soit conservée. Pour un auto-encodeur de Koopman, cette propriété peut apparaître comme une contrainte importante, dans la mesure où la dimension de l'espace d'état peut être insuffisante pour trouver un espace de fonctions d'observation invariant par l'opérateur de Koopman. Pour cette raison, on propose de travailler avec une version augmentée de l'auto-encodeur inversible. Concrètement, on ajoute un second encodeur, qui n'est pas modélisé par un flot normalisant mais par un réseau de neurones quelconque, et qui permet d'augmenter la taille de l'espace latent du modèle. Le décodage est cependant toujours réalisé à l'aide de l'inverse analytique du flot normalisant, et donc uniquement en utilisant les variables correspondant au premier décodeur. Ainsi, on travaille désormais avec un état latent divisé en deux parties : la partie inversible et la partie augmentée, chacune calculée par l'un des deux encodeurs. Ce modèle complet, que nous baptisons l'auto-encodeur de Koopman variationnel inversible augmenté (en anglais, Augmented Variational Inversible Koopman AutoEncoder ou AVIKAE), permet donc de modéliser des distributions de probabilité complexes, avec une expressivité importante notamment permise par l'augmentation de l'état.

Dans les expériences présentées, nous nous en servons notamment pour modéliser la distribution de probabilités pour l'état du vecteur de réflectance après un certain nombre de pas de temps, étant donnée une observation initiale (inférence classique) ou une trajectoire observée (assimilation de données). Ainsi, on reproduit des expériences similaires à celles des chapitres précédents, mais dans un contexte stochastique, ce qui implique encore une fois que les performances des modèles testés sont calculées en terme de CRPS et non d'erreur quadratique moyenne. Dans un premier temps, on compare les performances du modèle AVIKAE pour l'inférence, vis-à-vis de l'ensemble d'auto-encodeurs de Koopman présenté plus haut ainsi que de variantes partielles du modèle AVIKAE (auto-encodeurs non inversibles et inversibles mais non augmentés). Les résultats montrent que le modèle AVIKAE obtient de meilleurs résultats que ces méthodes concurrentes. Pour les expériences suivantes, on étudie différentes manières de travailler avec un modèle AVIKAE pré-entraîné. Par exemple, on montre qu'il est possible de partir d'une condition initiale considérée comme parfaitement fiable et de néanmoins obtenir de la variabilité

dans les prédictions subséquentes, simplement en fixant la partie inversible de l'état latent initial pour conserver de la variabilité uniquement dans la partie augmentée. Par ailleurs, on propose une manière d'adapter directement les méthodes d'assimilation de données variationnelle proposées pour les auto-encodeurs de Koopman non déterministes à ce modèle stochastique. Il s'agit simplement de définir le coût variationnel uniquement sur la moyenne latente initiale du modèle AVIKAE pré-entraîné, ce qui se fait de manière analogue à celle du modèle déterministe. Une variance initiale latente associée peut alors être calculée en décodant puis réencodant la partie inversible de cette moyenne initiale latente résultant de l'assimilation. Concrètement, on montre que cette méthode permet de bien meilleurs résultats en termes de CRPS que la méthode d'inférence naïve à partir d'une seule observation. L'une des perspectives de ce travail de thèse est de travailler avec des méthodes d'assimilation plus avancées, notamment en réalisant une optimisation conjointe sur la moyenne et la variance initiale latentes de la trajectoire. On pourrait dans ce cas utiliser le CRPS comme coût variationnel, ou bien travailler directement avec un calcul de vraisemblance, sur la trajectoire entière ou sur les marginales aux différents indices de temps.

Conclusion

En résumé, dans ce manuscrit de thèse, nous présentons l'apprentissage de dynamiques dans les séries temporelles d'images multispectrales comme un problème d'apprentissage auto-supervisé. Il s'agit donc d'apprendre une représentation générale des données, sans utiliser de labels associés à une tâche spécifique comme la segmentation d'images, d'une manière qui permette d'utiliser cette représentation pour résoudre un certain nombre de tâches spécifiques, toujours avec peu voire pas de données labellisées. Plus précisément, nous nous intéressons à des modèles basés sur la théorie de l'opérateur de Koopman, c'est-à-dire qu'il s'agit de modéliser l'évolution temporelle de l'état via un modèle latent d'évolution linéaire. Notre tâche prétexte pour l'entraînement du modèle est la prédiction de long terme de la dynamique des données à partir d'un seul état observé. En ce qui concerne les tâches aval, on s'intéresse notamment au débruitage, à l'interpolation et à la prédiction de long terme à partir de plusieurs observations à différents pas de temps. Ces tâches sont résolues via l'assimilation de données variationnelle, en utilisant un modèle auto-supervisé comme a priori dynamique de l'assimilation. En outre, nous nous intéressons à des modèles stochastiques, qui ouvrent la voie à de nouvelles possibilités telles que la quantification d'incertitudes et la détection d'anomalies.

TABLE OF CONTENTS

List of acronyms

ANN	Artificial Neural Network
AVIKAE	Augmented Variational Invertible Koopman AutoEncoder
CNN	Convolutional Neural Network
CRPS	Continuous Ranked Probability Score
DMD	Dynamic Mode Decomposition
EDMD	Extended Dynamic Mode Decomposition
FNO	Fourier Neural Operator
GAN	Generative Adversarial Network
GPU	Graphical Processing Unit
HNKO	Hamiltonian Neural Koopman Operator
IKAE	Invertible Koopman AutoEncoder
KIS	Koopman Invariant Subspace
LSTM	Long Short Term Memory
MLP	Multi Layer Perceptron
MSE	Mean Squared Error
NLL	Negative Log-Likelihood
PDF	Probability Density Function
PSNR	Peak Signal-to-Noise Ratio
RGB	Red Green Blue
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SITS	Satellite Image Time Series
SNR	Signal-to-Noise Ratio
SSL	Self-Supervised Learning
SSRAT	Spread-Skill RATio
SSREL	Spread-Skill RELiability
VAE	Variational AutoEncoder
VIKAE	Variational Invertible Koopman AutoEncoder
VKAE	Variational Koopman AutoEncoder

INTRODUCTION

Remote sensing means acquiring information about an object without making direct contact with this object. It typically designates information acquired on the surface of the Earth through the means of aircraft, drones or observation satellites, in the framework of monitoring missions such as the Copernicus programme carried by the European Union.

The main objective of the present thesis is to develop methods for processing remote sensing data with a temporal structure. The study of a single remote sensing image can already bring precious information on a corresponding area, allowing to solve problems like object detection [1] or image segmentation [2]. When studying several images of a same area taken at different times, one can obtain new possibilities for these tasks, but also solve new tasks such as change detection [3] and spatio-temporal interpolation [4]. In this work, we introduce methods that learn useful representations of the spatio-temporal data, enabling to solve such tasks.

In practice, we work on images of the surface of the Earth taken by satellites with a periodic orbit. Hence, we select areas of interest and collect series of pictures of these areas taken with a fixed periodic cycle. This results in the main objects of study for this thesis: satellite image time series (SITS).

There are several technical challenges associated to SITS. To begin with, unlike classical unidimensional time series, SITS have 4 dimensions: a spectral dimension, a temporal dimension and 2 spatial dimensions. The spectral dimension, in particular, accounts for the different spectral frequencies for which the satellites acquire information. Indeed, the pictures taken by the satellites are not usual "red-green-blue" images but generally contain a finer spectral resolution and/or information outside of the visible spectrum, notably in the infrared domain. Depending on the satellite, the number of spectral bands might be in the order of tens (multispectral images) or hundreds (hyperspectral images). Thus, the SITS data come in an unusually complex format, for which there are few existing methods. Indeed, there are existing methods for processing images, which contain two spatial dimensions and possibly a spectral dimension. Some of these methods are specifically designed for multi/hyperspectral images. Besides, there also exist many methods for processing time series, possibly with several variables. However, simultaneously taking into account the spatial, temporal and spectral dimensions is a conceptually more difficult challenge, which comparatively fewer methods have addressed so far.

Another important technical challenge associated to SITS is the scarcity of the data. Indeed, while we are only interested in observing the surface of the Earth, this information is often hidden by the presence of clouds or by the shades that these clouds project on the surface. For this

reason, the SITS have to be interpolated as a pre-processing step, or processed as incomplete data with a suitable method.

Some of the most common tasks involving SITS with different formats are classification (either at the pixel or at the image level), forecasting, interpolation and anomaly detection. Our general goal will be to learn representations of the data that are expressive enough to facilitate the resolution of these tasks. In order to tackle these challenges, we will propose methods that rely on artificial neural networks (ANNs), from which we shall now give a high-level description.

ANNs are a class of methods which falls under the framework of machine learning, which means that they are used to learn information from observed data in order to perform tasks without being given specific instructions. In particular, one way to understand ANNs is as extremely powerful regressors, with thousands, millions or even billions of adjustable parameters. Let us now explicit what we mean by this through a high-level description of a regression problem. Suppose that we are dealing with a set of data in the form of couples $(\mathbf{x}_i, \mathbf{y}_i)_{1\leq i\leq N}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in \mathbb{R}^m$. Then, a regression task typically consists in finding a *model*, i.e. a function $f : \mathbb{R}^n \to \mathbb{R}^m$ that successfully explains the dataset, in the sense that $f(\mathbf{x}_i) \approx \mathbf{y}_i$ for any $1 \leq i \leq N$. In addition, we would like f to also have good generalization properties, which means that for new inputs $\mathbf{x} \in \mathbb{R}^n$ which are not included in the (training) dataset, the associated prediction $f(\mathbf{x})$ should still be reasonable. In practice, the ability of the model f to generalize is often assessed by using a second (test) dataset. Perhaps the most simple model for regression is the linear regression with no bias term. Here, the idea is that each of the m entries of the output $\mathbf{y}$ is obtained as an affine function of the n entries of the input $\mathbf{x}$. Thus, a linear model contains $n \times m$ parameters, which are contained in a matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$. The regression function f simply writes $f(\mathbf{x}) = \mathbf{W}\mathbf{x}$. The optimal parameters are obtained by solving

$$\arg\min_{\mathbf{W}} \sum_{i=1}^{n} ||\mathbf{W}\mathbf{x}_i - \mathbf{y}_i||^2. \tag{1}$$

This equation can be solved analytically, making linear regression a simple and interpretable method. However, it will yield unsatisfying results when the true relationship between $\mathbf{x}$ and $\mathbf{y}$ is not linear. This motivates the introduction of more complex regressors like neural networks. One simple neural network model may be described with two weight matrices $\mathbf{W}_1 \in \mathbb{R}^{h\times n}$ and $\mathbf{W}_2 \in \mathbb{R}^{m\times h}$, separated by a nonlinear function σ. This function may be a rectified linear unit (ReLU), i.e. $\sigma(x) = \max(x, 0)$, or a sigmoid like $\sigma(x) = \frac{1}{1+e^{-x}}$ (applied entrywise on vectors). $\mathbf{W}_1$ and $\mathbf{W}_2$ contain adjustable parameters while σ is fixed and enables the global model to be nonlinear. Then, the model function f is written as $f(\mathbf{x}) = \mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x})$. The intermediary representation of the model, $\sigma(\mathbf{W}_1\mathbf{x})$, which is of size h, is often called the "hidden" or "latent"

representation. The regression problem writes:

$$\arg\min_{\mathbf{W}_1,\mathbf{W}_2} \sum_{i=1}^{n} ||\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}_i) - \mathbf{y}_i||^2. \tag{2}$$

Qualitatively, this minimization problem is very similar to the one of equation (1). In practice, if the hidden dimension h is large enough, the minimal value that is attained should be lower in most cases since this regression model is more expressive thanks to its nonlinear component. However, the problem of equation (2) cannot be solved analytically. Hence, it is solved by *gradient descent*, where the gradient is computed through *automatic differentiation*, which is a tool for computing the gradients of arbitrarily complex functions through the chain rule.

The vast majority of neural networks methods can be understood as variations of this simple regression example. In particular, classification problems can be seen as regression problems in which the output variable $\mathbf{y}$ takes discrete, rather than continuous, values. Many of the complex practical aspects of neural network training can be summarized in two fundamental questions:

— What is the architecture of the neural network? In other words, what is the structure of the function that is used in order to fit the data? In the above example, we have mentioned linear regression and regression with a two-layer neural network, but the architectures may be significantly more complex according to the problem that one is trying to solve.

— What is the actual loss function? In the above example of equation (2), the loss function is the mean squared error between the datapoints $\mathbf{y}_i$ and the corresponding predictions $f(\mathbf{x}_i)$ performed by the model. One might choose another function, such as the mean average error (i.e. substitute the squared L2 norm by an L1 norm), or introduce auxiliary terms, which enable to make the problem more convex or to improve the capacity of the model to generalize to unseen data.

When talking about deep learning, it is a commonplace statement that neural networks are always hungry for more data in order to reach better performance on the task for which they are designed, and that data is a valuable resource. While this statement may be correct in many cases, it does not explicitly mention which kind of data they are hungry for. In the machine learning field, a rough distinction is made between labelled and unlabelled data. As a simple example, suppose that one wants to train a neural network model to classify images, in categories such as "dog", "cat", "car"... Currently, the preferred way to do so is to train the model on a set of labelled data, i.e. a set of images that are each associated to a label among the categories "dog", "cat", "car"... This approach is referred to as *supervised learning*. In a few words, it consists in looking for a nonlinear embedding from the space of images to a new space, called the *latent space*, in which the classes can be easily discriminated, e.g. using a linear classifier.

Alternatively, one can wonder how to train a model when having mainly access to unlabelled data, i.e. images from which the category is unknown. In this case, one cannot directly train

a neural network to separate the data samples according to their labels. However, the images themselves still hold a lot of information, which one can try to leverage in order to learn generally useful representations of the data. Thus, instead of training a neural network on a label-dependent task, one can train it on a task that does not require any labels, in order to learn some sort of knowledge about the data at hand. For example, one could randomly mask some parts of the images and then ask the model to reconstruct the masked parts using the unmasked parts. This training task is then called a *pretext task*. It is defined in contrast to the *downstream tasks* which are the actual tasks that we want the model to be able to solve in the end, e.g. image classification. In order to solve such a downstream task, one can use a model that was pre-trained on a pretext task and then use a small amount of labelled data to slightly adapt the pre-trained model on this new task, which is called *fine-tuning*. This approach falls under *self-supervised learning*, in the sense that one finds a way to supervise the training of the model even when there is no specific task to solve. The success of self-supervised learning relies on the assumption that a model which is able to solve the chosen pretext task necessarily holds enough information to solve the more specific downstream task. For example, if a self-supervised model is able to successfully infer the masked shape of a cat's ears, then the model is somehow able to recognize from the remaining of the image that this is the image of a cat, rather than of a dog. Therefore, it should be able to classify cats and dogs.

To sum up, the purpose of self-supervised learning is to leverage a big amount of raw data with no labels in order to learn a general representation of the data that eventually enables to solve various useful tasks. This is a very interesting perspective since, in a lot of domains, unlabelled data are abundantly available while labelled data are more difficult to acquire. Indeed, constituting a labelled dataset generally requires the intervention of one or several human annotators, which makes it a costly and cumbersome process. One of the recent trends of self-supervised learning is to obtain *foundation models* [5], which are very large models trained on huge amounts of data, which are indeed able to solve many problems related to e.g. natural language processing [6] or image segmentation [7].

Remote sensing is an excellent example of a field where the unlabelled data are extremely abundant while the labels are scarce and difficult to obtain. Indeed, since the first Landsat mission was launched in 1972 [8], dozens of satellites have been acquiring images over the Earth's surface, enabling the accumulation of tremendous amounts of data. For example, the Sentinel satellites alone had already acquired 25 petabytes of data by the end of 2020 [9]. Yet, the vast majority of these data have not been labelled by any expert human annotators, and have only been processed through various automatic pipelines. These data offer opportunities for various applications, among which forest fire prevention [10], agriculture [11], biodiversity monitoring [12] and many others. However, each new task related to satellite images requires a comprehensive analysis, including the choice of the satellite that best suits the particular needs of the task

and possibly the annotation of a relatively large dataset to train a data-driven model with a supervised learning approach.

In order to circumvent this need to label large amounts of data for each new task involving satellite images, there are currently extensive discussions on how to obtain foundation models for remote sensing: see e.g. [13, 14], and [15] for an attempt that also takes into account the temporal dimension. However, compared to natural RGB images for which successful foundation models have already been proposed, remote sensing data are conceptually much more difficult to handle. The additional difficulties come in a large part from the heterogeneity of the data: as mentioned earlier, the satellite images are not acquired with the three usual ranges of wavelengths of the electromagnetic spectrum to which the human eye is sensitive. Instead, the information spans tens or hundreds of spectral bands, which differ according to the technical specificities of the satellite that took the picture. When additionally considering SITS instead of isolated satellite images, new difficulties appear. In addition to the intuitive degree of complexity brought by the addition of the temporal dimension to the data, the question of how to handle the (numerous) missing data in the time series is a challenge that is difficult to address with most existing time series processing frameworks.

To sum up, the question of designing foundation models for SITS is an interesting one, to which it is difficult to find an answer yet. In this thesis, we will be training neural networks with a self-supervised approach, but without trying to build a foundation model. Our ambition will be to design data-driven dynamical models that are able to provide a good understanding of a relatively restricted category of data, i.e. forested areas observed by the Sentinel-2 satellites. These models will heavily rely on the Koopman operator theory which, in a few words, states that any dynamical system can be described in a linear yet infinite-dimensional representation. Our main tool for solving downstream tasks using these pre-trained models will be variational data assimilation, which consists in solving an optimization problem involving both the fidelity to the observed data and to an a priori dynamical model. More precisely, we will use our models as dynamical priors in variational data assimilation costs, in order to solve tasks such as forecasting, interpolation and denoising.

An important aspect of the tasks that we consider is the quantification of the uncertainty. The majority of the methods based on neural networks only enables to get a pointwise prediction according to a given input. Typically, in a regression task such as the one that we used before as an example, it is desirable that a model produces an uncertainty estimate for every prediction that it performs, in order to have an idea of how confident the model is about its predictions. Ideally, instead of a single prediction, a model returns a complex probability distribution for the possible outputs given the considered input. Such models can be qualified as conditional stochastic models, and will be extensively discussed in the thesis.

Objectives and organisation of the thesis

The 4 chapters of this thesis are organised as follows:

Chapter 1 extensively describes the data that will be of interest throughout the thesis. It contains a general introduction of dynamical systems, with a particular focus on linear dynamical systems which are of interest for later understanding the Koopman operator. It also describes the different tasks that we will seek to solve on data organised as time series, with a probabilistic state space formulation. Finally, it describes remote sensing data with different visualisation approaches. In particular, a description of SITS is provided and the SITS datasets collected for the purpose of the thesis are presented.

Chapter 2 discusses the Koopman operator theory, along with practical data-driven models that leverage this theory for time series processing. A particularly detailed review is provided for Koopman autoencoders, which is a class of neural networks based on Koopman operator theory, under which our introduced models fall. We describe the improvements that we brought to the Koopman autoencoder framework, notably involving a new stability-promoting loss function term and the ability to resort to a continuous formulation of the model, either at training or testing time. The added value of our new loss function term, as well as the capacity of our model to leverage scarcely or irregularly sampled data, is demonstrated through a series of experiments on both synthetic dynamical systems and real SITS.

Chapter 3 reports the way in which our pre-trained Koopman autoencoder models can be used for data assimilation. In particular, we show that our models can be successfuly used as dynamical priors in a variational data assimilation cost. With a self-supervised learning perspective, using a model pre-trained on a forecasting task as a dynamical prior for assimilation can be seen as a downstream task. This prior may come as a part of the assimilation cost along with the data fidelity term: a classical data assimilation approach which we refer to as unconstrained assimilation. Alternatively, the prior may be used in a more constrained way, in the sense that the output time series may only be directly produced by the dynamical prior, rather than simply being prompted to roughly respect this prior. We refer to this approach as constrained variational data assimilation. We show through experiments on our SITS datasets that these assimilation approaches enable to greatly boost the forecasting performance by taking into account multiple observations instead of a single one to perform a prediction. We also show that they enable to solve new tasks such as interpolation and denoising, with far better performance than a classical data-agnostic approach such as the Cressman interpolation.

Chapter 4 deals with uncertainty quantification, and more generally stochastic predictions. It includes a comprehensive introduction to neural-network-based stochastic models, especially conditional ones. Then, it proposes several approaches that extend the deterministic Koopman autoencoder model of chapter 2 in order to make it able to produce a whole probability distribution

for its predictions, instead of a pointwise prediction. In particular, we present a method based on ensembles of Koopman autoencoders. We introduce variability-promoting loss functions derived from mathematical analysis in order to jointly train these ensembles of models. Another method that we introduce instead relies on variational autoencoders and normalizing flows. We show that this new model is able to produce complex prediction distributions, as well as to estimate the likelihood of a time series taking a particular value given an input observation.

Finally, in the conclusion, we summarize all of the contributions of the thesis, and then discuss some limitations and perspectives of this work. Some of these perspectives are explored in appendix C. Other appendices include brief reviews of the use of orthogonal matrices in deep learning (appendix A) and of previous works on self-supervised learning (appendix B).

Chapter 1

An overview of dynamical systems and satellite image time series

In this chapter, we will give a high-level overview on the problems that will be solved in the following chapters of the manuscript.

In section 1.1, we give a definition of dynamical systems, and introduce the Lorenz-63 system as an illustrative example. Then, in section 1.2, we describe some specific properties of linear dynamical systems. In particular, we discuss the possibility to switch from a discrete to a continuous formulation in subsection 1.2.1, and the long-term stability conditions in subsection 1.2.2. In section 1.3, we describe some usual problems that one can seek to solve when having only access to data supposedly generated by a dynamical system but without knowledge of the underlying equations that govern the data. In particular, we introduce probabilistic formulations for forecasting, interpolation and denoising, which are problems that we will be tackling throughout the remaining of the manuscript, particularly in chapter 3. To conclude this chapter, section 1.4 focuses on satellite image time series (SITS), which are the class of data on which our subsequent experiments will be focused. In particular, after introducing general concepts about these data in subsection 1.4.1, we describe datasets that were collected and pre-processed for the purpose of the thesis work in subsection 1.4.2.

1.1 Dynamical systems

Here, we describe dynamical systems from which the behaviour can be analysed in an exact, deterministic way. Roughly speaking, these are the systems from which the evolution from a given initial state can be described in an exact manner as long as one has enough knowledge of it, with no particular perturbations making the system noisy. They are defined in opposition to the stochastic dynamical systems, which are such that even with a complete knowledge of the state at a given time one cannot be certain about the subsequent evolution of the state.

A dynamical system is primarily characterised by the space $\mathcal{X}$ in which its state can be described. $\mathcal{X}$ is most often included in $\mathbb{R}^n$, where the positive integer n is the dimension of the dynamical system. Another important characteristic of a dynamical system is the function characterising the evolution of its state through time, which we will denote by $F : \mathcal{X} \to \mathcal{X}$. The

action of F is to advance an input state vector $\mathbf{x} \in \mathcal{X}$ by a step in time. Note that, since F is a function of the state only, we have that a dynamical system always behaves in the same way when fed with a given input state. In particular, the dynamics does not depend on a time of measurement.

In general, we will study the behaviour of a dynamical system when assuming that an initial condition is observed. For instance, one can assume that the state of the system is $\mathbf{x}_0$ at time 0. We then denote by $\mathbf{x}_t$ the state at time t. For any integer time $t > 0$, we obtain $\mathbf{x}_t$ by iterating the function F t times from $\mathbf{x}_0$:

$$\mathbf{x}_t = \underbrace{F \circ F \circ ... \circ F}_{\text{t iterations}}(\mathbf{x}_0) = F^t(\mathbf{x}_0), \quad t \in \mathbb{N}. \tag{1.1}$$

This description corresponds to discrete dynamical systems with a fixed time step. While equation (1.1) can always be written, one might wonder whether there might be an arbitrary discretization of a system that actually evolves continuously in time. If the system is indeed continuous, then one can generalize equation (1.1) to non-integer times $t > 0$, as follows:

$$\mathbf{x}_t = F^t(\mathbf{x}_0), \quad t \in \mathbb{R}_+. \tag{1.2}$$

We assume that, for any initial state $\mathbf{x}_0$, $\mathbf{x}_t = F^t(\mathbf{x}_0)$ defines a continuous trajectory. In particular, this means that, in case of existence, the time derivative of the state $\mathbf{x}$ at a given point may be computed as

$$f(\mathbf{x}) = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \lim_{t \to 0} \frac{F^t(\mathbf{x}) - \mathbf{x}}{t}. \tag{1.3}$$

Remind that a discrete formulation as in equation (1.1) can be always be derived from the continuous formulation, as explained for example in the definition 2 of [16]. If equation (1.1) cannot be generalized to non-integer time advancements t, then it characterizes a *discrete* dynamical system, also called a *map*, as referenced e.g. in [17]. If the equation can be generalized for any $t > 0$ as indicated in equation (1.2), then it characterizes a *continuous* dynamical system or *flow*[1]: see e.g. [18].

Some dynamical systems are defined primarily through the time derivative f of their state $\mathbf{x}$, as defined in equation (1.3). This choice comes from the fact that their discrete time advancement F can only be defined through a differential equation involving f, for which there is no analytical solution. In this case, the only way to represent a trajectory of the dynamical system in practice is through numerical integration, for example with the Euler integration scheme, which is the most simple known method for solving ordinary differential equations. However, such schemes

1. In our definitions, we have only considered non-negative time advancements t, i.e. systems that cannot necessarily be reversed. According to some authors, dynamical systems may be enabled to go backward in time, and the ones that cannot do so (like ours) are called semi-maps or semi-flows.

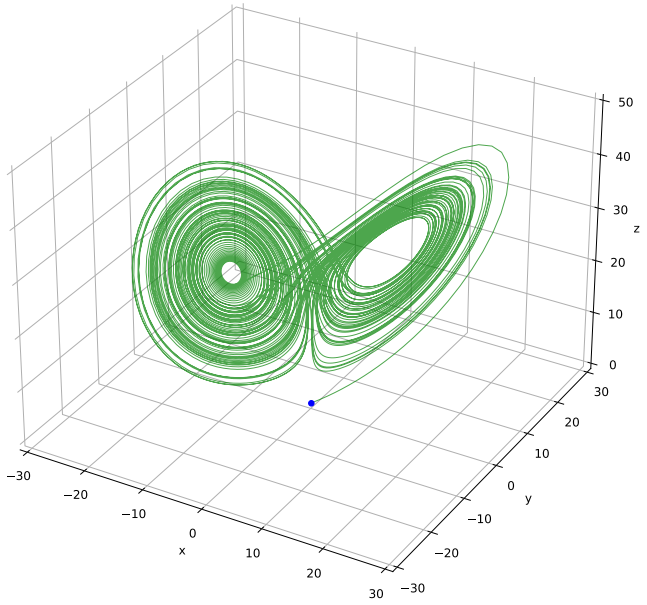

Figure 1.1 – Trajectory for the Lorenz attractor, obtained by numerically integrating the system of equations (1.4) for a duration of 100 seconds. The numerical integration scheme that we use is the LSODA solver [20], with a time step of 0.01 s. The initial state, $(x, y, z) = (0.1, 0, 0)$ is marked by a blue dot.

necessarily come with some integration errors. The best known example of a dynamical system which can only be represented by numerical integration is the Lorenz-63 dynamical system [19], introduced by Edward Lorenz in 1963. It is characterized by a 3-dimensional state $(x, y, z) \in \mathbb{R}^3$ evolving through the following set of equations:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z, \end{aligned} \tag{1.4}$$

where the parameters σ, ρ and β are assumed to be positive, with the most common configuration being $\alpha = 10$, $\beta = \frac{8}{3}$, $\rho = 28$.

On figure 1.1 we graphically represent the evolution of the state through equations (1.4). The Lorenz-63 dynamical system is known for its chaotic behaviour. This means that, considering two sets of initial conditions (x_1, y_1, z_1) and (x_2, y_2, z_2) separated by a very small distance ϵ, if we let the state evolve from these 2 starting points, the distance between the resulting states will grow exponentially as a function of time, until the two evolved states have no more in

common than two randomly selected points in the attractor of the system. The chaoticity of the Lorenz-63 system, coupled with the fact that its discrete evolution operator F cannot be expressed analytically, means that there exists no way to accurately predict the state of the system from an initial condition in the long term. Indeed, one can only resort to numerical integration schemes, which will inevitably accumulate errors as the prediction unrolls, and hence diverge from the true state of the system as a consequence of chaoticity.

Thus, the Lorenz-63 example shows that the behaviour of dynamical systems can be extremely difficult to describe in practice, even in cases where their mathematical formulation might look simple at first. In the next section, we will discuss linear dynamical systems, which is a specific class of dynamical systems for which many tools are available to perform an analysis, making them relatively simple systems.

1.2 Linear dynamical systems

An interesting concern about dynamical system is whether they are linear or nonlinear. This property simply depends on the form of the advancement function F. Indeed, if a dynamical system is linear, its advancement through time can be simply described by a matrix-vector product

$$F(\mathbf{x}_t) = \mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t, \tag{1.5}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$. Such dynamical systems can leverage linear algebra tools, making their description much more convenient than for general nonlinear systems. In particular, we will study the interplay between the discrete and continuous formulations of linear dynamical systems in subsection 1.2.1, and the asymptotic properties in subsection 1.2.2.

1.2.1 The discrete and continuous formulations

The vast majority of discrete linear dynamical systems described in terms of equation (1.5) admit a continuous formulation which can be computed by leveraging the matrix exponential and matrix logarithm. We shall therefore explain the interplay between the continuous and discrete formulations for linear dynamical systems after describing these important tools. As an important note, the results that we state about the matrix exponential and logarithm can be found in multiple sources, e.g. [21, 22].

The matrix exponential is a function defined for any matrix with real coefficients $\mathbf{A} \in \mathbb{R}^{n\times n}$, by the same power series as for real numbers:

$$\exp(\mathbf{A}) = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{A}^k. \tag{1.6}$$

A convenient way to compute the matrix exponential of a real matrix $\mathbf{A}$ is through diagonalization. Indeed, almost [2] any matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ can be diagonalized in $\mathbb{C}$, thus there exist a complex (invertible) matrix $\mathbf{U} \in \mathbb{C}^{n\times n}$ and a diagonal matrix $\mathbf{D} \in \mathbb{C}^{n\times n}$ such that

$$\mathbf{A} = \mathbf{U}^{-1}\mathbf{D}\mathbf{U}, \quad \mathbf{D} = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}. \tag{1.7}$$

We emphasize that the diagonal coefficients $d_1, ..., d_n$ are complex in the general case, since many matrices with real coefficients cannot be diagonalized in $\mathbb{R}$. Then, for any integer power k of $\mathbf{A}$,

$$\mathbf{A}^k = \mathbf{U}^{-1}\mathbf{D}^k\mathbf{U}, \quad \mathbf{D}^k = \begin{bmatrix} d_1^k & & \\ & \ddots & \\ & & d_n^k \end{bmatrix}. \tag{1.8}$$

This leads us to the fact that the matrix exponential of $\mathbf{A}$ can be very simply described through the complex exponential of the coefficients $d_1, ..., d_n$ of $\mathbf{D}$ as

$$\exp(\mathbf{A}) = \mathbf{U}^{-1}\exp(\mathbf{D})\mathbf{U}, \quad \exp(\mathbf{D}) = \begin{bmatrix} \exp(d_1) & & \\ & \ddots & \\ & & \exp(d_n) \end{bmatrix}. \tag{1.9}$$

We now move on to the description of the matrix logarithm. By definition, a matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$ is said to be *a* matrix logarithm of $\mathbf{A}$ if

$$\exp(\mathbf{L}) = \mathbf{A}. \tag{1.10}$$

Note that we do not talk about *the* matrix logarithm of $\mathbf{A}$ since the uniqueness of a matrix logarithm is not guaranteed: there might be several matrices from which the exponential equals $\mathbf{A}$. However, the existence of such a matrix is not guaranteed either. One can assess the existence of a matrix logarithm of $\mathbf{A}$ by looking at its diagonal form from equation (1.7). Indeed, whenever the complex logarithm of the diagonal coefficients $d_1, ..., d_n$ is defined, one simply has that the matrix

$$\mathbf{L} = \mathbf{U}^{-1}\mathbf{D}_\mathbf{L}\mathbf{U}, \quad \mathbf{D}_\mathbf{L} = \begin{bmatrix} \log(d_1) & & \\ & \ddots & \\ & & \log(d_n) \end{bmatrix}. \tag{1.11}$$

2. A real matrix which is not diagonalizable in $\mathbb{C}$ necessarily has roots with a multiplicity greater than 1 in its characteristic polynomial. Thus, any matrix from which the characteristic polynomial has no multiple roots (i.e. the vast majority of matrices) can be diagonalized.

verifies equation (1.10). Here, $\log : \mathbb{C}\backslash\mathbb{R}_- \to \mathbb{C}$ denotes the principal value of the univariate complex logarithm function, i.e. for a nonzero complex number in its geometric form $z = r\exp(i\theta)$ (with $r > 0$ and $\theta \in]-\pi, \pi[$), we have $\log(z) = \ln(r) + i\theta$. The interest of the principal value lies in the fact that, actually, any value $\ln(r) + i(\theta + 2\pi k)$ with an integer k is also a complex logarithm of z. This non-uniqueness of the complex logarithm justifies the non-uniqueness of the matrix logarithm. However, the principal logarithm is a natural choice since it takes the same value as the real logarithm when z is a strictly positive real number. Since the principal logarithm exists for any non-negative complex number, we have that the expression (1.11) exists as long as the diagonal coefficients $d_1, ..., d_n$ are all included in $\mathbb{C}\backslash\mathbb{R}_-$. Conversely, remind that all matrices $\mathbf{L}$ admit a matrix exponential.

Suppose now that we are studying a dynamical system defined by $F(\mathbf{x}) = \mathbf{A}\mathbf{x}$ for $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$. We further assume that $\mathbf{A}$ admits a (principal) matrix logarithm $\mathbf{L} \in \mathbb{R}^{n\times n}$, following equation (1.10). Then, for a positive integer time t, we immediately obtain that

$$F^t(\mathbf{x}) = \mathbf{A}^t\mathbf{x}, \quad t \in \mathbb{N} \tag{1.12}$$

and it is therefore easy to compute the state of the system many time steps ahead. In order to obtain a continuous formulation to this system, one needs to define non-integer powers of the matrix $\mathbf{A}$. This can be done by using the exact same definition as for real numbers, which writes:

$$\mathbf{A}^t = \exp(t\mathbf{L}), \tag{1.13}$$

To be clear, this definition of non-integer matrix powers is not commonplace in mathematics. However, it is obviously coherent with the definition of integer powers. In addition, by defining $F^t(\mathbf{x}) = \mathbf{A}^t\mathbf{x}$ for any $t > 0$, we have that $F^{t_1} \circ F^{t_2} = F^{t_1+t_2}$. Hence, we are able to describe an underlying continuous evolution corresponding to the discrete evolution of equation (1.12), which can be summarized as

$$F^t(\mathbf{x}) = \exp(t\mathbf{L})\mathbf{x}, \quad t \in \mathbb{R}_+. \tag{1.14}$$

In particular, we have that

$$f(\mathbf{x}) = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \lim_{t\to 0}\frac{F^t(\mathbf{x}) - \mathbf{x}}{t} = \lim_{t\to 0}\frac{(\exp(t\mathbf{L}) - \mathbf{I}_n)\mathbf{x}}{t} = \mathbf{L}\mathbf{x}, \tag{1.15}$$

which is the definition of a continuous linear dynamical system. We emphasize that $\mathbf{L}$ is the principal matrix logarithm, from which all eigenvalues have imaginary parts that lie in the interval $]-\pi, \pi[$.

Using these equations, time advancements in linear dynamical system are very easy to compute in practice. Suppose for example that we seek to compute $\mathbf{x}_{t+\tau}$ from $\mathbf{x}_t$, where τ is a large positive integer. If we only have a nonlinear function F at our disposal, then it has to be

iterated τ times, potentially with a large computational cost, to get a result. If we only have knowledge of the derivative f with no analytical expression of F (as is the case for the Lorenz-63 system), then the computation is even more difficult since we have to resort to a numerical integration scheme, which will be very costly and inevitably accumulate errors over time. In contrast, if the dynamical system is known to be linear and continuous, then $\mathbf{x}_{t+\tau}$ can simply be computed through equation (1.14), for any real positive advancement τ. Although this requires a numerical approximation of the matrix exponential, it does not imply any accumulation of the error over time. In fact, since $\mathbf{L}$ is assumed to be diagonalizable, computing the corresponding advancement matrix $\mathbf{A}^\tau = \exp(\tau\mathbf{L})$ essentially means computing the appropriate scalar complex exponentials of the n nonzero coefficients of a diagonal matrix.

1.2.2 Asymptotic properties of linear dynamical systems

The exponential function appearing in the time advancement of a dynamical system through equation (1.14) makes it obvious that, for the vast majority of advancement matrices $\mathbf{A}$ (or $\mathbf{L}$ in a continuous formulation), $\mathbf{x}_{t+\tau}$ will either converge to zero or diverge to infinity as τ goes to infinity. We will now characterize the stability and information preservation properties more explicitly. Let us suppose that we are studying a linear dynamical system characterized by the matrix $\mathbf{A}$ with the discrete time increments described by equation (1.5). Let us further assume that $\mathbf{A}$ can be diagonalized in $\mathbb{C}$, leading to its egeindecomposition in equation (1.8). Then, for a given initial condition $\mathbf{x}_0$ of the dynamical system, the state after k iterations will be

$$\mathbf{x}_k = \mathbf{A}^k\mathbf{x}_0 = \mathbf{U}^{-1}\mathbf{D}^k\mathbf{U}\mathbf{x}_0. \tag{1.16}$$

From this, one can see that the contribution of the j^{th} component of $\mathbf{x}_0$ in the eigenvectors basis $\mathbf{U}$ of $\mathbf{A}$ will grow as an exponential of base d_j. Reminding that d_j is a complex number in the general case, one should then see it in its geometric form $d_j = r_j e^{i\theta_j}$, where $r_j \in \mathbb{R}$ is the modulus of d_j and $\theta_j \in \mathbb{R}$ is the phase of d_j. In this form, the power k of d_j is written as $d_j^k = r_j^k e^{ik\theta_j}$. Thus, one can see that the phase of the eigenvalue characterizes the oscillatory behaviour of the corresponding component through time, while the modulus characterizes the evolution of its amplitude. When the modulus of d_j is strictly higher than 1, the associated component will exponentially tend to infinity, which is a physically unrealistic behaviour in the long term. In contrast, when the modulus is strictly lower than 1, the amplitude of the associated contribution will converge exponentially fast to 0. Although this is a physically sound behaviour, it still means that some information is lost, i.e. the dimension of the state is reduced in practice since, from any initial condition, the component corresponding to this eigenvector will inevitably converge to zero. In particular, when the state is encoded numerically, one cannot practically quantify how low the component has become after some amount of time.

Following the above analysis, it appears that the matrix $\mathbf{A}$ characterising a linear dynamical system has to have all of its eigenvalues inside of the complex unit disc in order to yield a realistic long-term evolution. Moreover, only the eigenvalues located exactly on the circle of modulus 1 enable the conservation of the information through time in practice. Hence, the matrices with all eigenvalues of modulus equal to 1 are of particular interest when modelling dynamical systems. In particular, the orthogonal matrices, characterized by

$$\mathbf{A}\mathbf{A}^\intercal = \mathbf{A}^\intercal\mathbf{A} = \mathbf{I}_n, \tag{1.17}$$

with $\mathbf{A}^\intercal$ the transpose of $\mathbf{A}$ and $\mathbf{I}_n$ the identity matrix of size n, indeed have complex eigenvalues of modulus 1, and they will be studied at length in chapter 2.

Moreover, if the matrix logarithm of $\mathbf{A}$ is defined, then the dynamical system may also be characterized in a discrete formulation by the (principal) matrix logarithm $\mathbf{L}$ of $\mathbf{A}$, which follows equation (1.10). The eigendecomposition of $\mathbf{L}$ is then given by equation (1.11). Let us then denote $l_j = \log(d_j)$. If we again use the geometric description of d_j as $d_j = r_j e^{i\theta_j}$, we have that the complex logarithm of d_j writes:

$$l_j = \log(d_j) = \log(r_j) + \log(e^{i\theta_j}) = \log(r_j) + i\theta_j. \tag{1.18}$$

Thus, we can see that the real part of l_j is related to the modulus of d_j, while the imaginary part of l_j is related to the phase of d_j. Hence, we obtain that

$$\begin{aligned} |d_j| \leq 1 &\iff \mathrm{Re}(l_j) \leq 0, \\ |d_j| = 1 &\iff \mathrm{Re}(l_j) = 0, \end{aligned} \tag{1.19}$$

where Re denotes the real part of a complex number. Hence, when working with the continuous formulation of a dynamical system, we will often want the real part of the eigenvalues of the matrix to be negative (or zero). For example, a result in Lie group theory states that the matrix exponential of an antisymmetric matrix is a special orthogonal matrix, and every special orthogonal matrix can be expressed as the matrix exponential of a real antisymmetric matrix. This property indeed agrees with equation (1.19) since the eigenvalues of an orthogonal matrix always have a modulus of 1 while the eigenvalues of a real antisymmetric matrix are always zero or strictly imaginary. The link between these classes of matrices will be further discussed in chapter 2. Since the articles in the literature often choose either the discrete or the continuous formulation of linear dynamical systems without expliciting the equivalences of equation (1.19), it is important to be able to recognize the stability and information conservation properties in both contexts.

1.3 Data-driven modelling of dynamical systems

In many cases, one can access neither a discrete nor a continuous operator for a given dynamical system. Instead, one only has at disposal some sets of data snapshots with associated timestamps. The satellite image time series, which are discussed in section 1.4, are a typical example of this case: one has a lot of successive satellite images at disposal but no direct information on the underlying equations from which these data originate. Therefore, one typically looks for a model that has some kind of predictive power for the associated dynamical system.

Typically, we can assume that we are working with a time series $(\mathbf{x}_{t_0}, ..., \mathbf{x}_{t_T})$, where each $\mathbf{x}_i$ corresponds to a vector of measurements of the state of a system. Although the generalization of the following concepts to continuously evolving dynamical systems is relatively straightforward, we suppose here that $t_0, ..., t_T$ are positive integers which come in an increasing order. In many cases, we have that the time series is regularly sampled in time, i.e. $t_k = k$ for every index k. However, the data may be irregularly sampled in some contexts, e.g. satellite image time series. Additionally, we will often consider that several time series are observed to train a machine learning model, but we put these considerations aside for now.

1.3.1 Some frequently encountered time series processing tasks

There are a variety of different tasks that one would like to be able to solve when handling such data from an unknown dynamical system. The simplest example of these tasks is forecasting. In a basic setup, it consists in building a function $\hat{F}$ such that, given an observed initial condition $\mathbf{x}_t$, one can get accurate predictions for the evolution of the system through:

$$\hat{F}^\tau(\mathbf{x}_t) \approx \mathbf{x}_{t+\tau}. \tag{1.20}$$

In this regard, it is important to note that the time advancement τ for which we want equation (1.20) to hold largely depends on the context. Typically, a given model $\hat{F}$ might excel at predicting the state of a system after one time step (i.e. $\tau = 1$) but diverge in the long term (i.e. $\tau \gg 1$). Therefore, there is an important yet fuzzy distinction between short-term and long-term forecasting tasks. Depending on whether the main objective is to predict in the short-term or in the long-term, the most appropriate model might not always be the same.

In some cases, one wants to perform forecasting based on the knowledge of the state of a system at several different times. Thus, one can take several previous values $(\mathbf{x}_{t_0}, ..., \mathbf{x}_{t_k})$ up to time $t = t_k$ as an input to infer predictions for the state at time $t + \tau$. This can be reduced to the previous case by simply ignoring the knowledge of the values before $\mathbf{x}_t$ and computing equation (1.20), which consists in a autoregressive model of order 1. Yet, more elaborate methods can take into account the remaining of the input data. Those can be best described as computing

the probability distribution of the subsequent states given all of the know states, i.e.

$$p(\mathbf{x}_{t_k+\tau}|\mathbf{x}_{t_0},...,\mathbf{x}_{t_k};\theta). \tag{1.21}$$

In this formulation, θ denotes the set of parameters of the used model. This formulation in fact encompasses even the deterministic models such as (1.20), where θ are then the parameters of $\hat{F}$ and the corresponding probability distribution is a Dirac distribution located on $\hat{F}^\tau(\mathbf{x}_{t_k})$. Here, it should be noted that the probability distribution of equation (1.21) should be compared to the probability distribution

$$p(\mathbf{x}_{t_k+\tau}|\mathbf{x}_{t_k};\theta) \tag{1.22}$$

corresponding to equation (1.20), where we let $t = t_k$. If the time series $\mathbf{x}$ is assumed to be Markovian, then the knowledge of a set of several past values is equivalent to the knowledge of the most recent of these past values, which leads to the equality of the probability distributions of equations (1.21) and (1.22). In this case, the probabilistic inference is considerably simplified. In particular, if the data $(\mathbf{x}_{t_0},...,\mathbf{x}_{t_T})$ has been generated by a groundtruth deterministic dynamical system with a function F, then the Markovian property holds, with a deterministic evolution of the system from the knowledge of its last state. However, as we will demonstrate in chapter 3, the knowledge of other past values often improves the forecasting performance in practice, either because the observation of these additional datapoints partially compensates for our incomplete understanding of F or because the data are actually intrinsically generated by a stochastic, rather than deterministic, dynamical system.

In fact, equation (1.21) may describe a more general class of problems when inferring values of the time series for time indexes that are not posterior to all datapoints. In particular, if the observed points $\mathbf{x}_{t_0},...,\mathbf{x}_{t_k}$ are irregularly sampled, then one might be interested in inferring the unknown values of the interval $[\![t_0, t_k]\!]$. Concretely, for a given time $t_0 \leq \tau \leq t_k$ for which x_τ is not observed, one can seek the probability distribution

$$p(\mathbf{x}_\tau|\mathbf{x}_{t_0},...,\mathbf{x}_{t_k};\theta) \tag{1.23}$$

which is an interpolation problem.

Besides, we are so far only interested in the probability distribution of the state at a single time given a set of available values at other times, yet one could be interested in the joint probability distribution of the values of $\mathbf{x}$ at different times given its values at other times, i.e.

$$p(\mathbf{x}_{\tau_1},...,\mathbf{x}_{\tau_m}|\mathbf{x}_{t_0},...,\mathbf{x}_{t_k};\theta). \tag{1.24}$$

For some models, the probability distributions of the variables may be independent from each

other knowing the observed data, which leads to the following simplification:

$$p(\mathbf{x}_{\tau_1}, ..., \mathbf{x}_{\tau_m} | \mathbf{x}_{t_0}, ..., \mathbf{x}_{t_k}; \theta) = \prod_{i=1}^{m} p(\mathbf{x}_{\tau_i} | \mathbf{x}_{t_0}, ..., \mathbf{x}_{t_k}; \theta). \tag{1.25}$$

However, in the general case, the independence property is not verified, which means that it is much more difficult in practice to compute the joint probability distribution of equation (1.24) than to compute the collection of the associated marginal probabilities of equation (1.23).

1.3.2 State-space models

So far, we have always assumed that we were computing probability distributions conditioned on completely trustworthy information at some set of times. However, in practical cases, the available data $\mathbf{x}_t$ cannot be fully trusted and should be considered as a possibly noisy observation of an underlying true state vector $\mathbf{y}_t$. In practice, this opens new applications to dynamical systems modelling, notably denoising and anomaly detection. One then has to consider so-called state-space models, from which we give a simple yet quite general form as:

$$\mathbf{y}_{t+1} = F(\mathbf{y}_t) + \epsilon_t \tag{1.26}$$

$$\mathbf{x}_t = H(\mathbf{y}_t) + \eta_t. \tag{1.27}$$

In these equations,

- $\mathbf{y}_t$ is the state variable, which enables to describe entirely the state of the dynamical system at time t.
- F is a deterministic function of discrete evolution.
- ϵ_t is a random variable, following a given probability distribution which represents the stochasticity of the dynamical system. It is often assumed to follow a Gaussian distribution which is independent on time, i.e. $\epsilon_t \sim \mathcal{N}(\mu_\epsilon, \mathbf{\Sigma}_\epsilon)$. It is also often assumed that ϵ is unbiased, i.e. $\mu_\epsilon = 0$.
- $\mathbf{x}_t$ is the observation variable, i.e. a variable containing possibly noisy and incomplete information on the state variable $\mathbf{y}_t$.
- H is the operator linking $\mathbf{y}_t$ to $\mathbf{x}_t$. In simple cases, it is a linear operator or simply the identity function.
- η_t is the observation noise, characterizing the stochasticity of the observed variable given the state variable. Like ϵ_t, it is often assumed to be Gaussian and time-independent, i.e. $\eta_t \sim \mathcal{N}(\mu_\eta, \mathbf{\Sigma}_\eta)$. Again, this noise is generally considered to be unbiased, which means that $\mu_\eta = 0$.

Note that, in most descriptions of the state-space models than can be found in the literature, the state variable is denoted with the letter $\mathbf{x}$ while the observed variable is denoted with the

letter $\mathbf{y}$. However, we chose to invert these usual notations here in order to keep them consistent with the previous notations of the chapter.

In the deterministic case considered so far, we simply have that $\eta_t = \epsilon_t = 0$ and H is the identity. This is a particular case of the state-space formulation, in which the observed variable is exactly the state variable, with no projection or noise. By introducing the more general framework, one can consider more complex cases, in which it is important to separate the imperfect observations $\mathbf{x}$ from the actual state of the system $\mathbf{y}$. In particular, a denoising problem might be formulated as finding the probability distribution

$$p(\mathbf{y}_1, ..., \mathbf{y}_T | \mathbf{x}_1, ..., \mathbf{x}_T). \tag{1.28}$$

Besides from the pure denoising problems, it is conceptually useful to consider the observed states as potentially untrustworthy. This means that, for example, many interpolation schemes will also provide suggestions for denoising the available datapoints. In addition, the general state-space formulation enables to have a theoretical understanding of outlier datapoints, which might lead to ignoring some points when performing predictions in practice.

1.3.3 A data-driven linear model: Dynamic Mode Decomposition

We now describe a simple but well-known technique to concretely model a dynamical system when having only access to observed data $\mathbf{x}_t$ with their corresponding timestamps.

We begin by assuming that the available data comes in the form of pairs of data snapshots $(\mathbf{x}_i, \mathbf{x}'_i)_{1 \leq i \leq T}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{x}'_i \in \mathbb{R}^n$ is the observation that comes one time step after the observation $\mathbf{x}_i$. From these pairs, one can constitute matrices $\mathbf{X} \in \mathbb{R}^{n \times T}$ and $\mathbf{Y} \in \mathbb{R}^{n \times T}$ such that, for $1 \leq i \leq T$, $\mathbf{X}_{.,i} = \mathbf{x}_i$ and $\mathbf{Y}_{.,i} = \mathbf{x}'_i$. One simple assumption, which would bring us all of the advantages of linear dynamical systems described in section 1.2 if it were to be true, is that there might exist a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ such that, for any data pair, $\mathbf{x}'_i \approx \mathbf{A}\mathbf{x}_i$. This assumption motivates the introduction of the following least-squares problem:

$$\mathbf{A}^* = \underset{\mathbf{A} \in \mathbb{R}^{n \times n}}{\arg\min} ||\mathbf{A}\mathbf{X} - \mathbf{Y}||_F. \tag{1.29}$$

This is a very well-known optimization problem, admitting an analytical solution which can be written as

$$\mathbf{A}^* = \mathbf{Y}\mathbf{X}^+, \tag{1.30}$$

where $\mathbf{X}^+$ denotes the Moore-Penrose pseudo-inverse of $\mathbf{X}$. Computing $\mathbf{A}^*$ is a method known as Dynamic Mode Decomposition [23] [3], which heavily relies on Koopman operator theory. We

3. More precisely, it corresponds to a variant known as exact dynamic mode decomposition [24], which is one of the most popular variants of the original method.

shall discuss this theory and its other applications for dynamical systems modelling at length in chapter 2.

1.4 Satellite image time series

In this section, we first introduce the general stakes of satellites images, and in particular of satellite image time series (SITS), in subsection 1.4.1. Then, in subsection 1.4.2, we describe in detail our process for collecting time series that will be used in a variety of experiments throughout the manuscript.

1.4.1 An introduction to satellite image time series

Over the last few decades, there has been an important increase of the number of satellites launched in orbit around the Earth, equipped with different kinds of sensors in order to enable the observation of the Earth's surface, mostly for scientific purpose. In this regard, the Copernicus program of the European Union, along with others, has taken a major role and enabled scientists to access tremendous amounts of image data continuously acquired by a number of satellites.

Observation satellites have various modalities of acquisition, among which optical and radar acquisitions. In the case of optical acquisition, satellites are equipped with different kinds of optical sensors, each reacting to a specific band of wavelengths according to a nontrivial function of acquisition, just like the three kinds of cone cells present in a human eye. Unlike the human eye however, optical observation satellites can acquire from one to thousands of spectral bands. Thus, they enable to catch a lot more information than a human eye would. Multispectral images typically feature 3 to 15 different spectral bands, while hyperspectral images can feature up to thousands of bands. Therefore, a multi/hyperspectral image can be described as a tensor $\mathbf{X} \in \mathbb{R}^{L \times W \times H}$ with three dimensions: a spectral dimension of size L, and two spatial dimensions of sizes W and H. We give a schematic representation of a multispectral image in figure 1.2. The term "spectral resolution" is often used to characterize the sizes of the intervals between the considered wavelenghts and the number of spectral bands that are considered.

With such amounts of pixelwise information, multi/hyperspectral images offer interesting possibilities to discriminate the different materials that could compose an image, among which various kinds of trees, crops, roads and buildings. Indeed, different categories of materials are often assumed to be characterized by specific kinds of reflectance spectra, which are often referred to as the "spectral signatures" of these materials. The combination of this spectral information with the spatial layout of the satellite images enables to solve specific computer vision tasks such as semantic segmentation [25] and object detection [26]. However, the wealth of the pixelwise spectral information generally comes at the cost of a reduced spatial resolution, since it is a major technical challenge to acquire data with both a high spectral and a high spatial resolution,

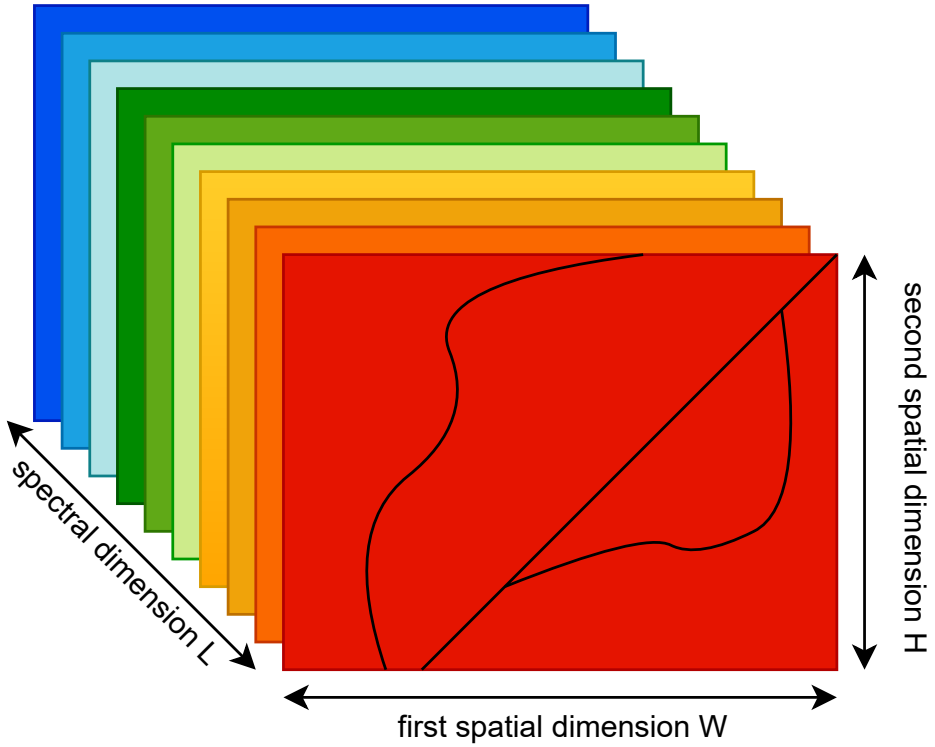

Figure 1.2 – Graphical representation of a multispectral image with $L = 10$ spectral bands. Each spectral band corresponds to a matrix of size $W \times H$, which can itself be seen as a black-and-white image containing the reflectance in the chosen spectral band for the considered spatial area.

often requiring some sort of compromise. For some satellites, such as the Sentinel-2 constellation, from which the data are extensively exploited in this manuscript, not all spectral bands are acquired at the same resolution. A simple solution to bring all spectral bands to the highest resolution is to interpolate the lower-resolution ones using a classical interpolation scheme such as bicubic interpolation [27]. However, such an interpolation scheme does not share the information between the spectral bands, and in particular does not use the information of the high-spatial-resolution bands to help obtaining a sharper interpolation for the interpolated spectral bands. Therefore, extensive work has been performed on pansharpening [28], which aims at combining the information of an image with a high spatial resolution and an image with a high spectral resolution in order to get the better of both worlds.

The rich spectral information acquired by satellites means that one cannot simply visualise it through a 2-dimensional image, as we usually do for a black-and-white or RGB picture. Therefore, instead of seeing a satellite image as a square, we see it as a cube, where the depth dimension

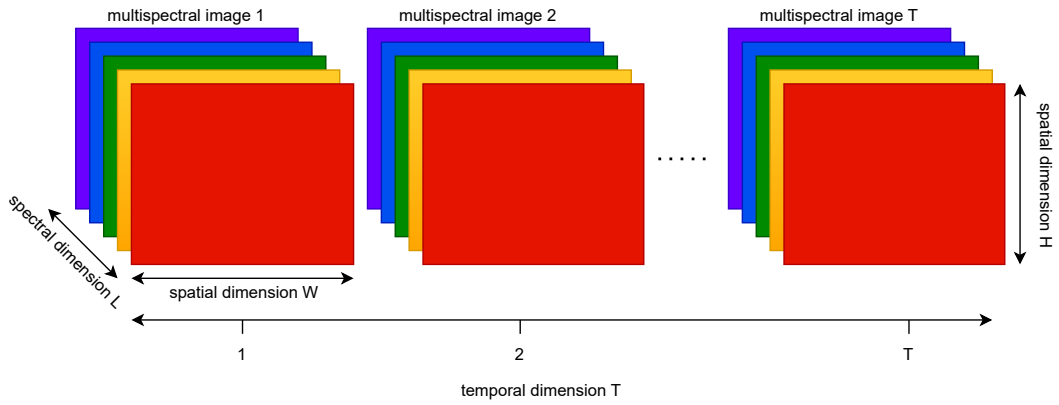

Figure 1.3 – A graphical representation of a SITS with T images, each with L spectral bands, a width W and a height H.

represents the spectral information. Again, we refer to figure 1.2 for a graphical representation of such a cube. Two possibilities for decomposing the information contained in this cube are:

— Given a particular pixel of a hyperspectral or multispectral image, one can extract all information related to this pixel. In this way, we obtain a 1-dimensional vector in which each entry is the reflectance of this pixel for a corresponding spectral band.

— Given a chosen spectral band, one can have a look at the reflectance of several pixels in this particular band. To do so, one can visualise a black-and-white image in which the color for each pixel indicates how much it reflects in the chosen spectral band. Thus, in this case, the data comes in the form of a matrix of size $W \times H$.

Of particular interest for our work are the satellites with a periodic orbit. Indeed, Earth observation satellites have a polar orbit, enabling them to take successive images of a given place in the world with a relatively high frequency. This is the case of the Sentinel 2-A and Sentinel 2-B, respectively launched in 2015 and 2017, which revolve around exactly the same orbit at a period of 10 days. Since their phases are opposed, together they take a picture of a given position every 5 days. One can therefore gather a succession of these regularly-sampled images, in order to obtain a satellite image time series (SITS). In this case, we add a temporal dimension to the 3 dimensions that compose a single multispectral image. Thus, a SITS is a 4-dimensional tensor $\mathbf{X} \in \mathbb{R}^{T \times L \times W \times H}$, where T, L, W and H respectively denote the length of the time series, the number of spectral bands and the width and height of the images. In figure 1.3, we show a schematic representation of such a tensor.

If we take interest in the projection of a SITS on one particular pixel of the considered spatial area, then we find ourselves with a matrix of size $T \times L$. This matrix can be directly visualised as an image, or as a collection of L univariate time series of length T, each corresponding to the

evolution of the reflectance value of one of the L spectral bands of the SITS over time. These 2 alternative visualisation approaches are graphically represented in figure 1.4.

Throughout the remainder of this manuscript, we will often assume that the spectral and temporal dimensions have more structure than the two spatial dimensions. For example, let us imagine that we seek to predict the future states that a particular pixel might take in a SITS $\mathbf{X} \in \mathbb{R}^{T \times L \times W \times H}$. Then, we argue that most of the useful information for solving this task is contained in the past reflectance vectors for this particular pixel, i.e. the aforementioned $T \times L$ matrix. Although having access to the complete time series might be helpful, this additional gain is limited. Besides, having access to all data from only one of the L spectral bands of the SITS, even for all pixels and past times, is likely to be an insufficient information since a single spectral band only gives a partial information of the content of a pixel. Finally, having access to a complete multispectral image, say the last one of the time series, enables to predict the future evolution of the system relatively accurately, yet much less accurately than when having access to the past time series for the single pixel of interest. These claims will be experimentally validated throughout the manuscript, particularly in chapter 3 and appendix C.

1.4.2 Description of our SITS datasets

For the purpose of this thesis work, Sentinel-2 image time series spanning durations of several years have been collected. Among the many existing platforms that enable to download satellite images, we used Theia [4], which is a platform aiming to provide public and easy access to environmental data. The images obtained from Theia are already pre-processed to correct the atmospheric effects using the MAJA software [29]. Hence, they correspond to a L2A (bottom of atmosphere) product.

The regions of interest that we consider are two forest areas in France: the forest of Fontainebleau and the forest of Orléans. In particular, we use two crops of 500×500 pixels. Since the highest spatial resolution for the Sentinel-2 data is 10 m, these crops correspond to squares of 25 square kilometers. We show their locations in figure 1.5. The distance between the two areas is approximately 60 km, which means that they should be subject to similar climatic conditions. Although they are both mostly composed of forests, they also contain some quantity of other land occupations, such as grass, roads, water and buildings. In addition, one can qualitatively see (even when only considering the visible wavelengths) that even the actual forest areas of both images do not necessarily have the same colors (i.e. spectra). In other words, the respective distributions of the data in the two spatial areas largely differ, probably because the proportions of the different species of trees differ. We will leverage this fact throughout our experiments using data-driven models. Concretely, we will train models on the Fontainebleau area only and then assess the capacity of these models to transfer their knowledge to the Orléans

4. `https://www.theia-land.fr/en/homepage-en/`

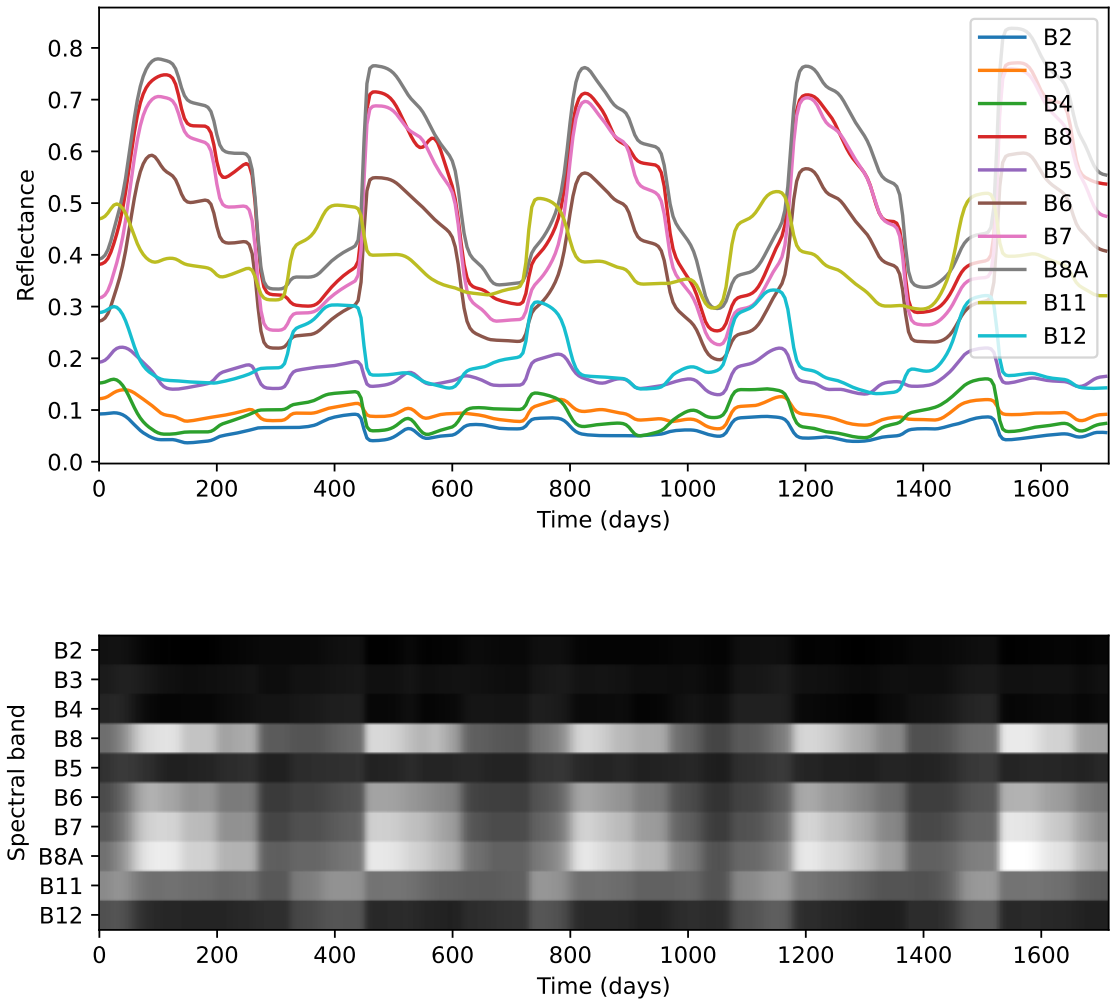

Figure 1.4 – Two alternative ways of projecting a SITS onto one pixel of the considered spatial area. On the top, we plot the L unidimensional time series of length T each corresponding to one of the spectral bands, on a same graph with a single scale. On the bottom, we directly represent the data matrix as a grayscale image of size $T \times L$, where each row corresponds to a spectral band while the horizontal axis corresponds to the time. One can check that the information contained in the two visualisations is indeed the same, with the visible pseudo-periodicity of one year in both cases. The notations "B2, B3..." correspond to standard notations for the high-resolution spectral bands of the Sentinel-2 data, see e.g. `https://en.wikipedia.org/wiki/Sentinel-2` for more information.

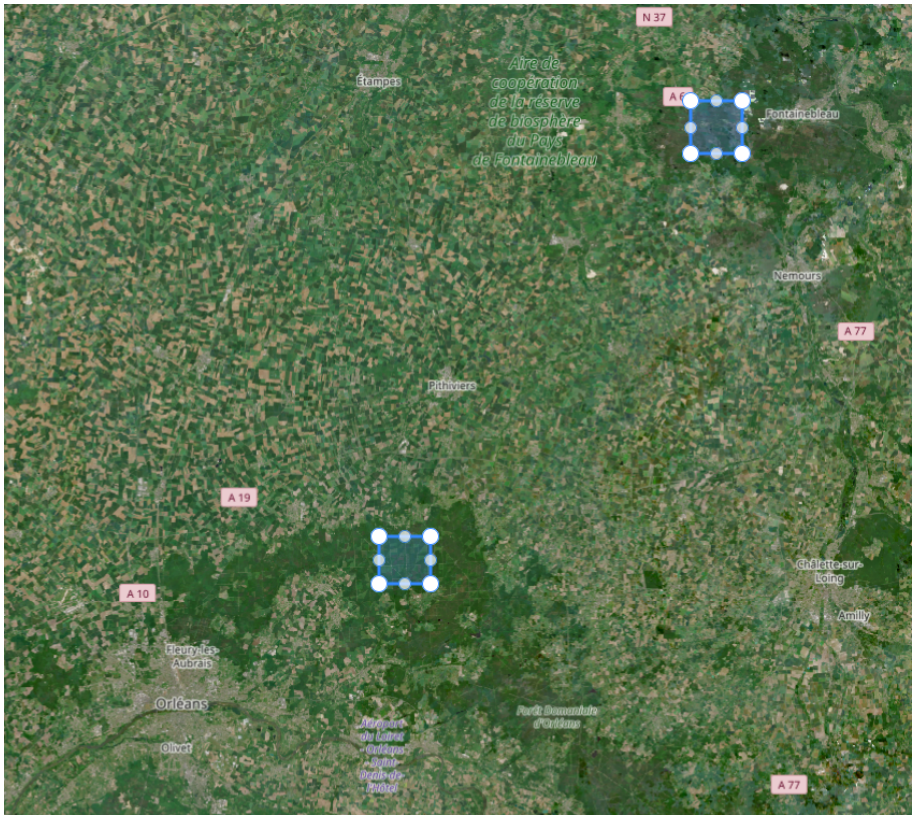

Figure 1.5 – Representation of the two areas where data are collected, in the forest of Fontainebleau (top right) and the forest of Orléans (bottom left).

area. Thus, we will have an idea of the capacity of different models to generalize to different data distributions.

Both SITS span a duration of nearly 5 years, from February 2018 to November 2022. Thus, in contrast to many datasets which only contain time series over a duration of five years, the datasets that we use enable to visualise the dynamics of the areas over several years. Since the revisit time, i.e. the time separating two consecutive observations of the Sentinel-2 satellites, is of 5 days, the considered time window contains approximately 340 discrete time steps.

The Sentinel-2 images are composed of 13 spectral bands with 3 different spatial resolutions: 10 m, 20 m and 60 m. In our case, we select only the 10 spectral bands that have a 10 or 20 m resolution, and we bring back all of the data to the 10 m resolution grid by resorting to a bicubic interpolation of the spectral bands that originally had a lower resolution.

One difficulty that we have not mentioned yet concerning optical satellite images is that a significant part of the data are corrupted by clouds or by the clouds' shades, which obstruct the view of the surface. The preview of a Sentinel-2 image corrupted by clouds is shown in figure 1.6. Thus, when constituting a SITS, one typically excludes the images from which a significant

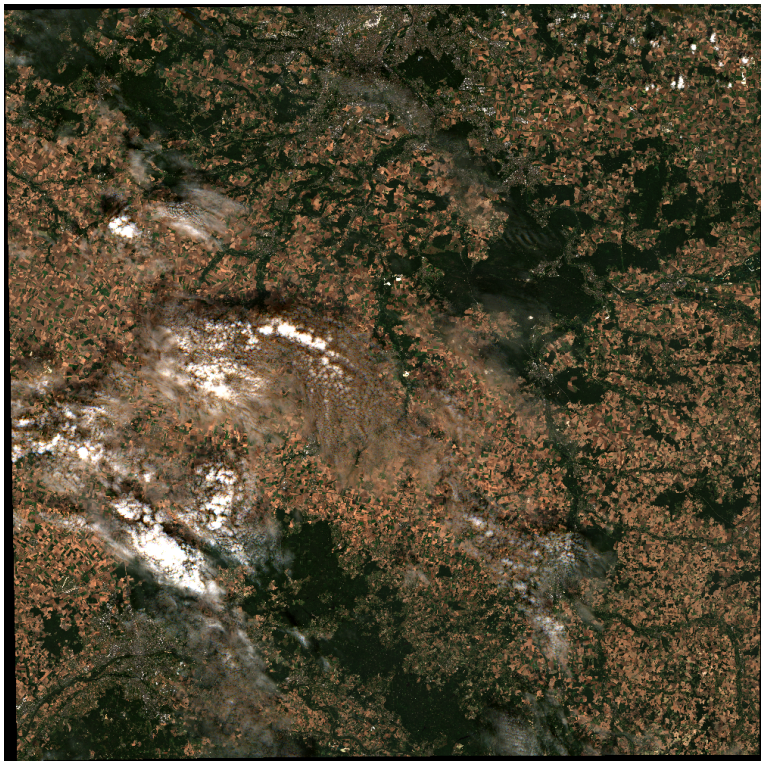

Figure 1.6 – The preview of a Sentinel-2 tile corrupted by the presence of clouds. The preview has been obtained from the Theia platform.

portion is covered by clouds, with an appropriate level of tolerance to the cloud coverage. From there, several choices are possible. One possibility is to accept working with an incomplete time series which is irregularly-sampled in time. The data availability mask may also be chosen to be spatio-temporal in order to account for the exact part of the corrupted data in each image, yet this choice is not considered in the thesis, and thus our data availability masks will only be temporal. A second possibility is to use an interpolation method in order to reconstruct the images for which the exact state is not available. Each of these two approaches have their own flaws. When working with incomplete time series, the processing is more difficult since most of the existing dynamical models are sequential and designed to process regularly-sampled data. In contrast, when working with interpolated data, it is easier to train a data-driven model but, since the training data is partly synthetic, the model will learn to reproduce the interpolation scheme along with the true distribution of the data, which is an undesirable effect. In order to always use the most relevant approach depending on the method that we use to model the data, we will keep both an irregularly-sampled and an interpolated version of the SITS for both of the spatial areas.

In figure 1.7, we show a graphical representation of an irregularly-sampled time series for 4 spectral bands of a pixel from the Fontainebleau area. One can see from this example that

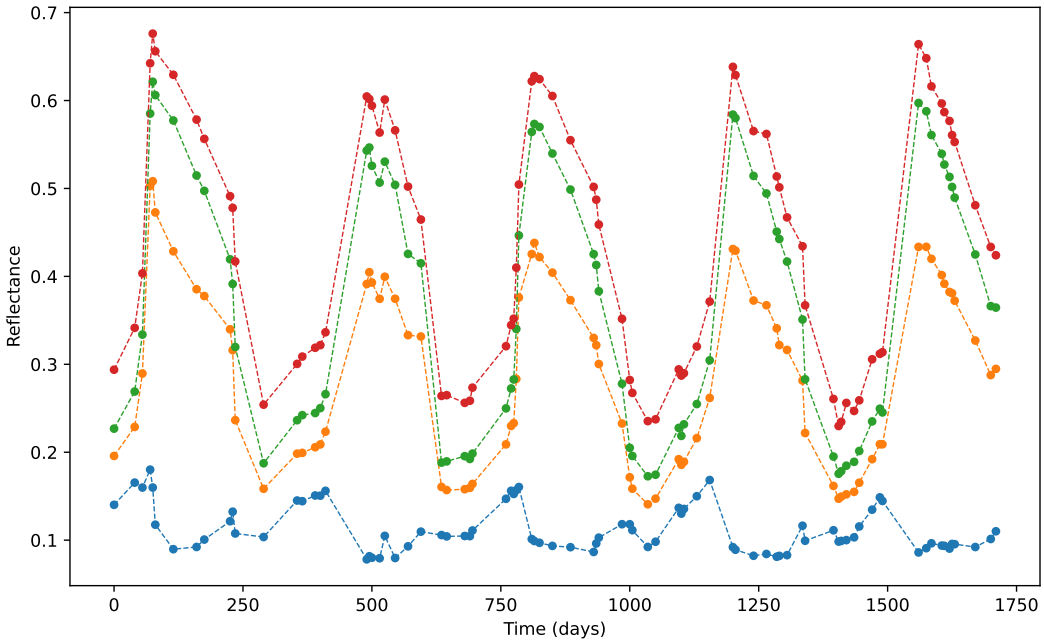

Figure 1.7 – Graphical representation of 4 spectral bands for one pixel of an incomplete SITS. The dots represent the actual datapoints while the dashed lines are only used to link the consecutive datapoints.

the sampling pattern is highly irregular, with some temporal regions being sampled at a high frequency while some other ones suffer from a lack of data. This irregular sampling pattern results from the fact that the probability of observing clouds over time is nontrivial. For example, there is typically a higher rate of retained data samples in the summer (local maxima for the reflectance in most bands) than in the winter (local minima for the reflectance in most bands). Nonetheless, this data is sufficient to have a good view of the pseudo-periodic pattern of oscillation that the dynamics follows with a period of one year, which is simply due to the seasonal dynamics of the forest.

Concerning the interpolated version of the data, we resort to a Cressman interpolation, as introduced in [30]. In a few words, this method is performed independently on each pixel and spectral band, and its main idea is to compute a complete time series in which each value is a weighted sum of the values of all available datapoints. The weights, which are normalized in order to sum to 1, are proportional to a Gaussian radial basis function of the temporal distance, with a radius R corresponding to 15 days, i.e. 3 time steps of 5 days. Formally, for every pixel and spectral band, we interpolate a set of scalar points $(x_t)_{t\in H}$ with its set of associated timestamps $H \subset [\![0, T]\!]$. The resulting regularly-sampled time series can be denoted $(y_t)_{t\in[\![0,T]\!]}$. Then, the (unnormalized) contribution of each time $t' \in H$ of the original data on the value y_t associated

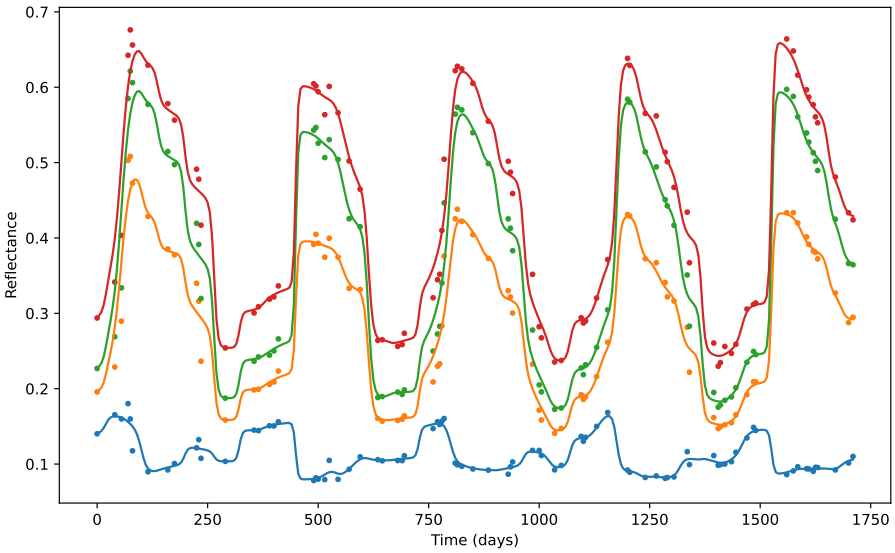

Figure 1.8 – Graphical representation of 4 spectral bands for one pixel of a SITS interpolated with the Cressman interpolation scheme. The dots represent the original datapoints that are used to perform the interpolation, while the plots represent the interpolated trajectories. On time indexes for which there is an original datapoint, the interpolated value does not necessarily correspond to this point, which is part of the expected behavior of the Cressman interpolation.

to a time $t \in [\![0, T]\!]$ is:

$$w_{t,t'} = \exp \frac{-(t-t')^2}{2R^2}. \tag{1.31}$$

Then, the value for y_t is obtained as a normalized weighted sum of the original values using the collection of weights $w_{t,t'}$ for all times $t' \in H$, as follows:

$$y_t = \frac{1}{\sum_{t' \in H} w_{t,t'}} \sum_{t' \in H} w_{t,t'} x_{t'}. \tag{1.32}$$

In particular, one can note that, for a given $t \in H$, in the general case we have that $y_t \neq x_t$.

The result of the Cressman interpolation for the same pixel and spectral bands as in figure 1.7 is shown in figure 1.8. The original datapoints used to perform the interpolation are also scattered on this image in order to facilitate the visualisation. One can see that, as previously mentioned, the interpolation scheme does not only infer the missing values of the original data, but also changes the values that were already available. Thus, it also acts as a smoothing or denoising technique, which makes the data easier to process but also brings slight changes to its original distribution.

On figure 1.9, we show sample images from the interpolated version of the time series. At a first glance, there are no visible interpolation artifacts. One can visually assess the seasonal

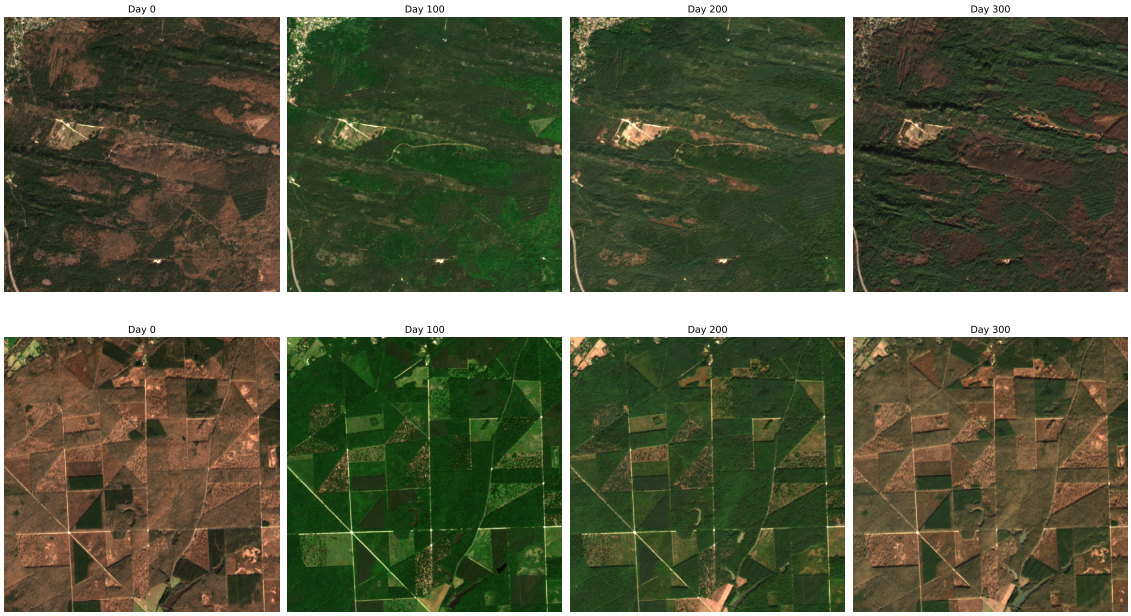

Figure 1.9 – Data samples from the interpolated time series of Fontainebleau (top row) and Orléans (bottom row). The samples are RGB compositions using only the three bands of the visible spectrum in the data, instead of the 10 spectral bands of the pre-processed time series. The date for day 0 is the 25/02/2018.

dynamics of both areas, with the images being globally "greener" in the summer and "browner" in the winter.

1.5 Conclusion

In this first chapter of the thesis, we have extensively discussed the data that we will be processing in the following chapters, along with the tasks that we will seek to solve. In particular, we have introduced dynamical systems, with a particular focus on linear dynamical systems. These notions are key to the Koopman operator theory, which we discuss in the following chapter and on which most of the methods that we introduce are based. We also introduce the forecasting, interpolation and denoising tasks, which are typically carried on observed time series in a data-driven way, assuming that the observed data originate from a complex dynamical system in the framework of a state-space model. We have given probabilistic interpretations of these tasks, which will be useful for understanding the data assimilation methods from chapter 3 and the stochastic models from chapter 4. Finally, we have given an overview of satellite image time series (SITS), and described the Sentinel-2 datasets that will be used in most of the experiments of chapters 2, 3 and 4.

In the following chapter, we will introduce methods that enable to perform forecasting by

means of a latent linear dynamical model, based on the Koopman operator theory. Notably, we will use the developments of subsection 1.2.2 to make sure that this latent linear model yields stable long term predictions. We will also leverage the continuous formulation of dynamical systems, outlined in subsection 1.2.1, to handle irregularly or scarcely sampled data. The performance of the introduced methods for data-driven forecasting will be compared against classical dynamical models such as the long short-term memory neural network as well as other Koopman-based models. The comparisons will be performed on well-known nonlinear dynamical systems such as the simple pendulum, as well as on our SITS introduced in subsection 1.4.2.

Chapter 2

The Koopman operator and its numerical implementations

This chapter is divided in 2 sections. In section 2.1, we provide the necessary background knowledge on Koopman operator theory before describing numerical implementations of this operator - particularly the Koopman autoencoder framework. Some of these techniques were developed before the beginning of the thesis (e.g. [31, 32, 33, 34]) while others are concomitant to our work, e.g. [35, 36, 37]. In section 2.2, we describe our own implementations, which are part of the Koopman autoencoder framework. These include a new orthogonality-promoting loss term and a set of techniques that leverage the ability of a Koopman autoencoder to produce continuous trajectories, either in the training or the testing stage.

2.1 Existing work on the Koopman operator

2.1.1 Background on the Koopman operator theory

As a reminder, we are working on n-dimensional dynamical systems evolving in a space $\mathcal{X} \in \mathbb{R}^n$, with a discrete dynamical function $F : \mathcal{X} \to \mathcal{X}$ such that, for any given state $\mathbf{x}_t$ in $\mathcal{X}$ to which we associate a time $t \in \mathbb{R}_+$, we have that $F(\mathbf{x}_t) = \mathbf{x}_{t+1}$.

An observation function of $\mathcal{X}$ is simply a function $g : \mathcal{X} \to \mathbb{R}$. It could be something as simple as the projection of $\mathcal{X}$ to one of its n variables: these observation functions are called the canonical observation functions, and together they form the identity function of $\mathcal{X}$. The observation functions could also be much more complex as they contain all possible functions with a real scalar output and a state $\mathbf{x} \in \mathcal{X}$ as its input. The set of all observation functions g with some regularity [1] can be denoted as $\mathcal{G}(\mathcal{X})$, or simply $\mathcal{G}$.

Koopman operator theory, introduced by B.O. Koopman in 1931 [39], states that, for any dynamical system, no matter its dimensionality or overall complexity, the discrete dynamics F can be described by an operator acting on the set $\mathcal{G}$ of its observation functions: the so-called Koopman operator $\mathcal{K} : \mathcal{G} \to \mathcal{G}$. For any observation function $g \in \mathcal{G}$, the action of the Koopman

1. Continuity or integrability are often imposed, among other choices: see e.g. section 2.1 of [38] for a discussion.

operator on g is defined as

$$\mathcal{K}g = g \circ F \tag{2.1}$$

In other words, for any observation function g and state $\mathbf{x}_t$ of $\mathcal{X}$, we have that

$$\mathcal{K}g(\mathbf{x}_t) = g(\mathbf{x}_{t+1}). \tag{2.2}$$

As a consequence from this definition, the Koopman operator has two important properties:

1) $\mathcal{K}$ is linear, as a trivial consequence of the linearity of the function space $\mathcal{G}$. Indeed, we have that, for any observation functions g and h in $\mathcal{G}$, and for any state $\mathbf{x}_t$,

$$\mathcal{K}(g+h)(\mathbf{x}_t) = (g+h) \circ F(\mathbf{x}_t) = (g+h)(\mathbf{x}_{t+1}) = g(\mathbf{x}_{t+1}) + h(\mathbf{x}_{t+1}) = \mathcal{K}g(\mathbf{x}_t) + \mathcal{K}h(\mathbf{x}_t). \tag{2.3}$$

2) $\mathcal{K}$ is also an infinite-dimensional operator since the space $\mathcal{G}$ on which it acts is itself of infinite dimension.

To sum up, the Koopman operator can be seen as an infinite-dimensional matrix, characterizing which observation functions of the state intervene when advancing any observation function in time. In practice, we are most interested in knowing its expression for the n canonical observation functions constituting the state vector, since they give a complete knowledge of the state of the system. For a given observation function $g \in \mathcal{G}$, the Koopman operator perspective tells us that there exist sets $H_g = (h_1, h_2, ...)$ and $\Lambda_g = (\lambda_1, \lambda_2, ...)$ of observation functions and weights such that, for any $\mathbf{x}_t \in \mathcal{X}$, $g(\mathbf{x}_{t+1}) = \sum_i \lambda_i h_i(\mathbf{x}_t)$. Although there is no guarantee that H_g is finite for a given g, one can focus on observation functions that have the most convenient expressions. In particular, like in classical linear algebra, one can find interest in studying the eigenfunctions of the Koopman operator, which are observation functions $g \in \mathcal{G}$ verifying $\mathcal{K}g = \lambda g$ (and consequently $H_g = \{g\}$). Trivial examples of such eigenfunctions are all constant observation functions of $\mathcal{G}$, i.e. functions g_c such that $g_c(\mathbf{x}) = c$ for any $\mathbf{x} \in \mathcal{X}$, with a constant value $c \in \mathbb{R}$. Those observation functions are eigenfunctions of the Koopman operator with an eigenvalue of 1.

Although the Koopman operator is infinite-dimensional in the general case, finite approximations might be sound for some systems. In this regard, most of existing numerical methods inspired from the Koopman operator theory rely on so-called Koopman invariant subspaces (KIS). A KIS is a set of observation functions for which the action of the Koopman operator on one of its members always remains inside of the set. More formally, it is a set $G \subset \mathcal{G}$ such that

$$\forall g \in G, \mathcal{K}g \in G. \tag{2.4}$$

Let us suppose that we know a KIS G of dimension d, spanned by linearly independent observation functions $(g_1, g_2, ..., g_d) \in \mathcal{G}^d$. We denote by $[h]_\mathcal{B} \in \mathbb{R}^d$ the vector containing the coordinates of any observation function $h \in G$ in the basis $\mathcal{B} = (g_1, g_2, ..., g_d)$. By basic linear algebra, $[.]_\mathcal{B} : G \to \mathbb{R}^d$

is a linear operator which defines a bijection between G and $\mathbb{R}^d$. Then, for a function $g \in G$, we have that $[g]_{\mathcal{B}}$ is a vector $\mathbf{a} \in \mathbb{R}^d$ such that $g = \sum_{i=1}^{d} a_i g_i$. Moreover, since G is a KIS, $[\mathcal{K}g]_{\mathcal{B}}$ is also defined and, by linearity of $\mathcal{K}$ and $[.]_{\mathcal{B}}$, we have that

$$[\mathcal{K}g]_{\mathcal{B}} = [\sum_{i=1}^{d} a_i \mathcal{K}g_i]_{\mathcal{B}} = \sum_{i=1}^{d} a_i [\mathcal{K}g_i]_{\mathcal{B}}. \tag{2.5}$$

From there, it is easy to see that the restriction of the infinite-dimensional operator $\mathcal{K}$ on the finite set G is simply a matrix $\mathbf{K}_G \in \mathbf{R}^{d \times d}$, whose coefficients are contained in the vectors $[\mathcal{K}g_i]_{\mathcal{B}}$. This enables to represent the time evolution of any observation function $g \in G$ very easily in practice. Although Koopman invariant subspaces are rather common, not all of them enable to describe the whole dynamical system accurately. Indeed, since the state of the system is described entirely by its canonical observation functions, one would like these functions to be either included in a KIS or accurately reconstructed using the observation functions from a KIS.

2.1.2 Practical implementations of the Koopman operator

We have seen that the key to many numerical methods related to the Koopman operator theory is to find an (approximate) finite-dimensional KIS. In order to do so, several approaches exist. Early methods consist in first designing a dictionary of observable functions through manual choices and then trying to find the best linear match for a one-time-step transition of these observables. Many of them are derived from a reference approach called Extended Dynamic Mode Decomposition [40] (EDMD), which we shall now describe. Like many related methods, EDMD assumes in its classical setup that a dynamical system is known only through a set of state vectors organized in pairs $(\mathbf{x}^j, \mathbf{y}^j)_{1 \leq j \leq N}$, such that $\mathbf{y}^j = F(\mathbf{x}^j)$. Thus, the goal is to approximate F through a data-driven model. To do so, using one's knowledge and assumptions about the system at hand, one can design a dictionary $\Phi = (\phi_1, ..., \phi_d)$ of d observation functions of the state. Common choices of such observables are:

— The n canonical observation functions of the state. This choice corresponds to Dynamic Mode Decomposition, which was discussed in section 1.3.3.

— The set of all Hermite polynomials constituted from the state variables up to a certain degree.

— A chosen set of radial basis functions.

Denoting $\mathbf{X} = (\mathbf{x}^1, ..., \mathbf{x}^N)$ and $\mathbf{Y} = (\mathbf{y}^1, ..., \mathbf{y}^N)$, we then seek the best linear match between $\Phi(\mathbf{X})$ and $\Phi(\mathbf{Y})$, i.e.

$$\min_{\mathbf{K} \in \mathbb{R}^{d \times d}} \mathcal{L}(\mathbf{K}) = ||\mathbf{K}\Phi(\mathbf{X}) - \Phi(\mathbf{Y})||^2 \tag{2.6}$$

From the form of this optimization objective, one can see that if the functions of Φ constitute a KIS of the dynamical system, then there exists a matrix $\mathbf{K}^*$ such that $\mathcal{L}(\mathbf{K}^*) = 0$. Although this

is rarely attainable in practice, one hopes that the space Φ is approximately Koopman invariant so that its advancement through time can be described in a reasonably accurate way by a matrix. The optimization problem from equation (2.6) is simply solved using the least squares equation, like in the more specific case of DMD [23] presented in section 1.3.3.

Suppose now that a dictionary of observation functions Φ and a Koopman matrix $\mathbf{K}$ have been obtained for the dynamical system under study, so that for a given input state $\mathbf{x}_t \in \mathcal{X}$, we have $\mathbf{K}\Phi(\mathbf{x}_t) \approx \Phi(\mathbf{x}_{t+1})$. In this form, this model is not so satisfying since it enables to keep track of $\Phi(\mathbf{x})$ rather than $\mathbf{x}$, which is the true quantity of interest. Fortunately, for most of the common choices of dictionaries (including the 3 choices mentioned above), the n canonical observation functions are directly included in Φ. Therefore, one can simply obtain a prediction for $F(\mathbf{x})$ by projecting $\mathbf{K}\Phi(\mathbf{x})$ onto the appropriate set of variables. Although the least-squares solution for $\mathbf{K}$ in equation (2.6) only accounts for one-time-step prediction, one can autoregressively obtain a prediction for a time advancement of τ from a given initial state $\mathbf{x}$ by computing $\mathbf{K}^\tau\Phi(\mathbf{x})$ and then projecting it to the state space. Moreover, assuming that $\mathbf{K}$ admits a matrix logarithm, one can even compute its "non-integer" powers in order to obtain a continuous model for the advancement of the state, as described at length in section 1.2.

Typically, Dynamic Mode Decomposition (DMD) [23] is used for high-dimensional dynamical systems for which a sound linear approximation exists, often with a dimension reduction of the state through singular value decomposition. In contrast, other variants of EDMD are used for lower-dimensional dynamical systems for which a good choice for the dictionary of observables can lead to a good linear model. However, this dictionary often has to contain a high number of functions. For example, the number d of polynomial observation functions scales exponentially with the chosen maximal order of the polynomials. Therefore, EDMD typically requires manipulating a high-dimensional matrix $\mathbf{K} \in \mathbb{R}^{d\times d}$. In order to alleviate this issue, several solutions have been proposed.

One solution is to define the set of observable functions more implicitly, through the use of a low-dimensional kernel which may encompass a large number of observation functions. This is the key idea of Kernel Dynamic Mode Decomposition (KDMD) [41]. There are a number of works leveraging this principle [42, 43, 44, 45], which we will not discuss further in order to focus on the other class of methods addressing the same issue: neural network-based Koopman methods.

The basic idea of neural network-based Koopman implementations is to substitute the manual design of a dictionary of observation functions for a dynamical system by an automatic learning of this dictionary. In order to do so, the observation functions are parameterized by a trainable neural network. The objective of this substitution is to obtain a set of observation functions for which the linear approximation is good even with a small dimension. Therefore, it amounts to finding a d-dimensional KIS, where d is typically of much lower dimension than the one of a hand-chosen dictionary for EDMD. Using classic machine learning terminology, we may hereafter

refer to the output of this learnt set of observation functions as the encoded state or latent state. The neural network that learns the relation from the state space to the KIS will be referred to as an encoder.

Again, the question of how to reconstruct the input state from the observation functions is of great importance. Some works [31, 46] use a similar strategy to EDMD by constraining the inclusion of the (non-trainable) canonical observation functions of the dynamical system in the dictionary of observation functions, along with the actually trainable functions. This enables an exact reconstruction of the input state from the encoded state consisting in the (partly learnt) KIS, just like for EDMD. The inconvenient of this strategy is that it means assuming that there exists a low-dimensional KIS containing the canonical observation functions of the dynamical system.

In order not to rely on this assumption anymore, and therefore obtain a less constrained learning problem where the encoded state may even be of much lower dimension than the input state, most of subsequent works have chosen to remove the constraint of including the input state in the latent state, hence opting for a fully learnt Koopman invariant subspace. In this case, it is important to ensure that the learnt KIS still includes enough information to be useful in inference. In practice, this means that one should be able to (approximately) reconstruct an input vector by passing its corresponding latent vector into a decoder ψ. In other words, one jointly learns an encoder ϕ and a decoder ψ such that, for any input state $\mathbf{x} \in \mathbb{R}^n$, $\psi(\phi(\mathbf{x})) \approx \mathbf{x}$. Finally, such a model also learns the latent evolution of the state through a matrix $\mathbf{K}$, which might be trained by automatic differentiation along with the parameters of ϕ and ψ or obtained by solving the least squares problem at every optimization step. The expected behavior of this model is that, for any initial state $\mathbf{x}_t$ and time increment τ,

$$\psi(\mathbf{K}^\tau \phi(\mathbf{x}_t)) \approx \mathbf{x}_{t+\tau}. \tag{2.7}$$

Again, τ might be any real number if we compute the corresponding non-integer power of $\mathbf{K}$ through

$$\mathbf{K}^\tau = \exp(\tau \mathbf{L}), \tag{2.8}$$

where $\mathbf{L}$ is the matrix logarithm of $\mathbf{K}$, as defined in equation (1.11). Note that the decoder ψ could take multiple forms. In most of the works falling under the framework of equation (2.7), ϕ and ψ are both neural networks forming a neural autoencoder. It is worth noting here that a neural autoencoder is simply a composition of two neural networks ϕ and ψ for which we expect that $\psi \circ \phi(\mathbf{x}) \approx \mathbf{x}$, where the encoded state $\mathbf{z} = \phi(\mathbf{x})$ is usually of lower dimension than $\mathbf{x}$, justifying an interpretation of the autoencoder as a nonlinear version of the well-known principal component analysis. However, in most of the so-called Koopman autoencoder methods, the encoded state $\mathbf{z} = \phi(\mathbf{x})$ is actually of higher dimension that $\mathbf{x}$. Thus, these particular autoencoders seek to

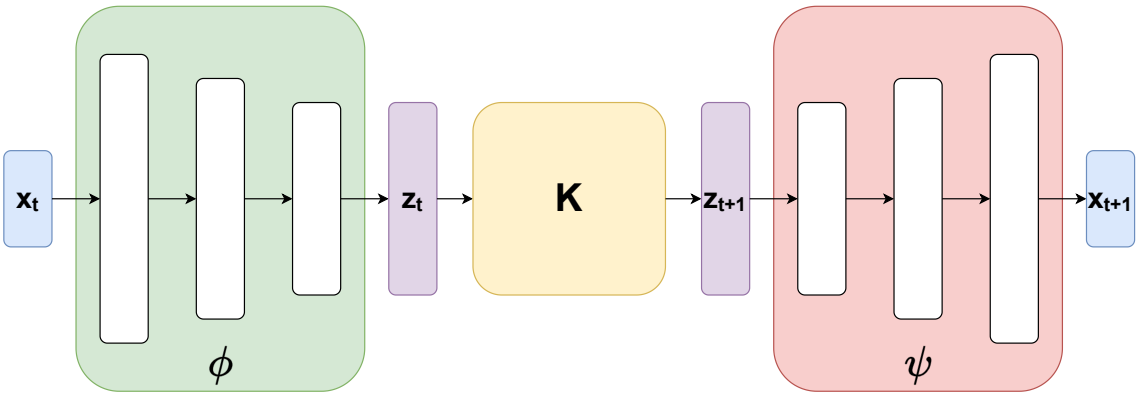

Figure 2.1 – A schematic representation of the most common Koopman autoencoder architecture, composed of an encoder ϕ and a decoder ψ forming an autoencoder, with a matrix of advancement $\mathbf{K}$. As indicated, the encoded state $\phi(\mathbf{x}_t)$ will hereafter be denoted $\mathbf{z}(t)$.

find a higher-dimensional representation of the state from which the evolution can be described linearly, which is quite different from the basic setup for autoencoders. We show in figure 2.1 a schematic view of the Koopman autoencoder architecture.

Some other frameworks have been proposed for the definition of ψ. A simple possibility is to define ψ as a second learnt linear operator rather than as a nonlinear neural network. Although this choice means a reduced expressivity, it matches common classical (i.e. not based on neural networks) implementations of the Koopman operator, e.g. [47]. Although few works have made this design choice, a recent study [48] suggests that using a linear decoder might actually perform better than the more commonly used nonlinear decoder. It should be noted that such a choice brings the general architecture closer to the EDMD framework, since the projection operator used when the canonical observation functions are included in the encoder is itself a linear operator.

Another recently explored design choice ([49, 50]) is to use an analytically invertible encoder ϕ, so that one can simply replace the jointly learnt decoder ψ by the exact inverse ϕ^{-1} of ϕ. The most commonly used invertible neural networks take inspiration of the literature on normalizing flows [51, 52] and rely on successions of invertible coupling layers. Like the more classical multilayer perceptron, neural networks based on invertible coupling layers benefit from universal approximation theorems [53], so that, in theory, their invertibility does not come at the cost of a reduced expressivity. However, the invertibility of ϕ requires the conservation of the dimension in the latent space, which is a limitation for Koopman models since there might not exist a sufficiently low-dimensional KIS.

Coupling layer normalizing flows, and Koopman autoencoder models that use these flows, will be extensively discussed in chapter 4.

2.1.3 Training a Koopman autoencoder model

For now, we will stay focused on the most common Koopman autoencoder (KAE) framework, where ϕ and ψ are two separate neural networks (generally multi-layer perceptrons, sometimes convolutional neural networks). Once this design is chosen, the question of how to obtain the desired behaviour of equation (2.7) is nontrivial. As is usually the case for neural networks, it largely resides on the design of the loss function used for training the model. We denote by θ the set of all trainable parameters of a KAE. θ then contains:

— The trainable parameters θ_ϕ of ϕ,

— The trainable parameters θ_ψ of ψ,

— The d^2 coefficients of the matrix $\mathbf{K}$.

Let us assume that we have at disposal a set of N observed time series of length $T+1$ to train on, denoted as $(\mathbf{x}_{i,t})_{1\leq i\leq N, 0\leq t\leq T}$. Note that, for any $1\leq i\leq N$ and $0\leq t\leq T$, $\mathbf{x}_{i,t}$ is a vector of size n. The main goal of the model training is to find a set of parameters θ that is good at fitting the training data but also generalizes well to unseen data. This is generally assessed by using a second set of trajectories, which is called the test dataset and is not used during training but only to evaluate the generalization performance of a trained model. It is common in machine learning to use regularisation terms in the loss function of a neural network, i.e. terms that do not improve the ability to fit the training data but improve the ability of the model to generalize. We shall first present the most naive loss function for training a KAE and then introduce common regularisation terms.

The most straightforward choice of loss function is one that directly reflects the quality of the predictions obtained through equation (2.7), i.e. the so-called prediction loss:

$$L_{pred}(\theta) = \sum_{1\leq i\leq N} \sum_{1\leq \tau\leq T} ||\mathbf{x}_{i,\tau} - \psi(\mathbf{K}^\tau \phi(\mathbf{x}_{i,0}))||^2. \tag{2.9}$$

Note that, similarly to the objective of DMD and EDMD, we are mostly interested in the squared L^2 norm of the prediction errors, i.e. the mean squared error (MSE). Though this is not an obvious choice, the squared L^2 norm is by far the most commonly used for training neural networks for regression tasks.

In theory, should we have an infinite amount of data, this loss function would be enough to obtain a model that generalizes well. However, there are several issues in practice. Imagine, for example, that the training trajectories only have length $T+1=2$, which corresponds to the classical setup for DMD and EDMD. Then, equation (2.9) reduces to

$$\sum_{1\leq i\leq N} ||\mathbf{x}_{i,1} - \psi(\mathbf{K}\phi(\mathbf{x}_{i,0}))||^2. \tag{2.10}$$

Assuming that this loss is brought to an arbitrarily low level, we have actually no guarantee that

ϕ, ψ and $\mathbf{K}$ truly keep their respective roles. In particular, there might be situations in which $\mathbf{K}$ converges to an identity matrix, so that ϕ and ψ learn the one-time-step advancement of an input state through the dynamical system rather than an invertible encoding of this state. In other words, this would imply

$$\psi \circ \phi(\mathbf{x}_{i,0}) \approx \mathbf{x}_{i,1} \tag{2.11}$$

In this case, predicting several time steps ahead through equation (2.7) would lead to $\psi(\mathbf{K}^\tau \phi(\mathbf{x})) = \psi \circ \phi(\mathbf{x})$, which corresponds to a constant prediction through time. To avoid this, one would need to go through ϕ and ψ for each time step rather than encode the input state through ϕ once, multiply the encoding by the suitable power of $\mathbf{K}$ and then decode the result through ψ (which is the approach of equation (2.7)). This is not fundamentally wrong, as some works introduce no matrix $\mathbf{K}$ and explicitly expect ϕ and ψ to learn the one-time-step dynamics. Such a model is called a time-lagged autoencoder [54]. It is also similar to the periodic re-encoding strategy from [48], although this work recommends re-encoding the state every $p > 1$ time steps rather than every time step. However, equation (2.11) obviously contradicts equation (2.7), which has several implications in practice, among which the impossibility to obtain a continuous formulation of the model and the increased computation time for long-term predictions due to the fact that computing $\phi \circ \psi$ is a lot more computationally intensive than performing a matrix-vector product by $\mathbf{K}$. More generally speaking, optimizing for the loss function (2.10) will most certainly lead to a situation where $\mathbf{K}$ and (ϕ, ψ) jointly learn the one-time-step dynamics when we actually want (ϕ, ψ) to be an autoencoder and $\mathbf{K}$ to represent the latent dynamics. This will inevitably lead to large prediction errors when predicting several time steps in the future through equation (2.7). Although training the model to predicting several time steps ahead partially alleviates this issue, one way to make sure that this does not happen is to explicitly make sure that (ϕ, ψ) behave as an autoencoder. This is done through a first regularisation loss term which we call the auto-encoding loss:

$$L_{ae}(\theta) = \sum_{1 \leq i \leq N} \sum_{0 \leq t \leq T} ||\mathbf{x}_{i,t} - \psi(\phi(\mathbf{x}_{i,t}))||^2. \tag{2.12}$$

Once this second loss term is introduced, the predictions should be more stable, yet problems still remain. Indeed, as previously mentioned, the dimension d of the latent or encoded state $\phi(\mathbf{x})$ is typically higher than the dimension n of the input state $\mathbf{x}$. Intuitively, this means that ψ is very unlikely to be injective since its input space is higher dimensional than its output space. More generally, ψ is a "black box" neural network model from which the injectivity cannot be guaranteed. Thus, even though the auto-encoding relationship is guaranteed by the loss term (2.12), the latent state $\hat{\mathbf{z}}_{i,t} = \mathbf{K}^t \phi(\mathbf{x}_{i,0})$ might quickly diverge from the encoding $\mathbf{z}_{i,t} = \phi(\mathbf{x}_{i,t})$ of the true state as the time t increases, even when the prediction $\hat{\mathbf{x}}_{i,t} = \psi(\mathbf{K}^t \phi(\mathbf{x}_{i,0}))$ remains close to the true state $\mathbf{x}_{i,t}$. In order to see why this is problematic, let us again assume that $T = 1$. Then, if we train a model that is good for 1-time-step prediction but for which the predicted latent state

is not close to the encoding of the following state, there is no guarantee that even the 2-time-step prediction will still be good. In contrast, if we have that $\hat{\mathbf{z}}_{i,1} \approx \mathbf{z}_{i,1} = \phi(\mathbf{x}_{i,1})$, then we also have that $\hat{\mathbf{z}}_{i,2} \approx \mathbf{K}\phi(\mathbf{x}_{i,1})$, which should also reflect in the input space. In other words, if the encodings of the states are well fitted by predicting 1 time step ahead in the latent space, then one can obtain good predictions in the input space several time steps ahead even when training only for one-time-step predictions. To ensure that this is the case, a second regularisation loss term is used:

$$L_{lin}(\theta) = \sum_{1 \leq i \leq N} \sum_{1 \leq \tau \leq T} ||\phi(\mathbf{x}_{i,\tau}) - \mathbf{K}^{\tau}\phi(\mathbf{x}_{i,0})||^2. \tag{2.13}$$

We now have three loss function terms to minimize: the prediction loss, the auto-encoding (or reconstruction) loss and the linearity loss, respectively in equations (2.9), (2.12) and (2.13). In practice, one can design a loss function that takes all three into account by setting weights α and β to quantify their relative weights. This gives:

$$L_{total}(\theta) = L_{pred}(\theta) + \alpha L_{ae}(\theta) + \beta L_{lin}(\theta). \tag{2.14}$$

In many Koopman autoencoder works, e.g. [33, 55, 56], the model is trained through a similar combination of loss terms, although sometimes $\alpha = 0$ or $\beta = 0$. The paper that first proposed to use all of the 3 loss terms in conjunction was [32], which inspired many of the subsequent works. In particular, our own approach, detailed in section 2.2, is built on this basis. We now describe some approaches that fall under the Koopman autoencoder framework but use a different loss function.

In [57], the authors seek matrices $\mathbf{K}$ which exhibit (through equation (2.7)) asymptotically stable dynamics in the sense of Lyapunov [58]. This property informally means that, if the initial latent state is close enough to an equilibrium point of the latent dynamics, then the latent state will eventually converge to this equilibrium point. Thus, it acts as a regularization for a Koopman-inspired model, enabling to avoid diverging predictions. In order to attain it, the authors of [57] note in their Proposition 1 that the latent dynamics are asymptotically stable in the sense of Lyapunov if and only if, for any (symmetric) definite positive matrix $\mathbf{Q} \in \mathbb{R}^{d \times d}$, there exists a (symmetric) definite positive matrix $\mathbf{P} \in \mathbb{R}^{d \times d}$ such that

$$\mathbf{K}^{\intercal}\mathbf{P}\mathbf{K} - \mathbf{P} = -\mathbf{Q}. \tag{2.15}$$

This property makes use of the well-known discrete-time Lyapunov equation [59], which is commonly used in control theory. In practice, the authors propose to approximate the matrix $\mathbf{P}$ such that

$$\mathbf{K}^{\intercal}\mathbf{P}\mathbf{K} - \mathbf{P} = -\mathbf{I}_d, \tag{2.16}$$

and to penalize its small negative eigenvalues. Thus, the general idea is to choose a particular

definite positive matrix $\mathbf{Q} = \mathbf{I}_d$ and to enforce the corresponding matrix $\mathbf{P}$ in equation 2.15 to be positive definite. One hopes that it enables to obtain - or at least favor - the Lyapunov asymptotic stability of the latent dynamics. Concretely, the penalty that the authors use is set to

$$a = \begin{cases} \rho(p) = \exp(-\frac{|p-1|}{\gamma}) & \text{if } p < 0 \\ 0 & \text{otherwise,} \end{cases} \tag{2.17}$$

where p is an eigenvalue of $\mathbf{P}$ and γ is a tunable parameter. Finally, the loss function proposed in [57] can be written as

$$L(\theta) = L_{pred}(\theta) + \alpha L_{ae}(\theta) + \kappa \sum_{i=1}^{d} \rho(p_i), \tag{2.18}$$

where p_i denote the eigenvalues of $\mathbf{P}$. Note that this approach necessitates to compute an approximation of $\mathbf{P}$, deduced from $\mathbf{K}$ through equation (2.16), for each optimization step. We do not describe here how this approximation is performed, and refer to [57] for the complete algorithm. Importantly, one can note that the loss function from equation (2.18) does not feature a linearity loss term as in the reference loss function from equation (2.14).

A more recent article [34] proposes to model the backward latent dynamics through a second trainable matrix. In theory, one would simply have to invert the forward prediction matrix to make backward predictions through a KAE model, yet the authors make the observation that this process leads to unstable predictions in practice. Thus, in comparison to the classical KAE architecture, the authors add a component in the form of a second matrix $\mathbf{D} \in \mathbb{R}^{d \times d}$ which is trained to give the backward latent dynamics of the model. Thus, for this specific architecture, the set of trainable parameters θ now includes the coefficients of $\mathbf{D}$ along with the coefficients of $\mathbf{K}$ and the parameters of ϕ and ψ. The matrix $\mathbf{D}$ is used along with the unchanged encoder ϕ and ψ to obtain the following behavior:

$$\psi(\mathbf{D}^\tau \phi(\mathbf{x}_t)) \approx \mathbf{x}_{t-\tau}. \tag{2.19}$$

This equation is analogous to equation (2.7), yet the time advancement is backward in this case. Thus, this behaviour is obtained through a backward prediction loss term, which is analogous to the (forward) prediction loss term from equation (2.9):

$$L_{back}(\theta) = \sum_{1 \leq i \leq N} \sum_{1 \leq \tau \leq T} ||\mathbf{x}_{i,T-\tau} - \psi(\mathbf{D}^\tau \phi(\mathbf{x}_{i,T}))||^2. \tag{2.20}$$

In addition, the authors seek to promote the consistency between the forward prediction matrix

$\mathbf{K}$ and its backward counterpart $\mathbf{D}$. This is done through a consistency loss, which is expressed as

$$L_{con}(\theta) = \sum_{k=1}^{d} \frac{1}{2k} ||\mathbf{K}_{k*}\mathbf{D}_{*k} - \mathbf{I}_k||_F^2 + \frac{1}{2k} ||\mathbf{D}_{k*}\mathbf{K}_{*k} - \mathbf{I}_k||_F^2, \tag{2.21}$$

where $\mathbf{M}_{k*}$ and $\mathbf{M}_{*k}$ respectively denote the upper k rows and leftmost k columns of a matrix $\mathbf{M}$. This loss term roughly means that $\mathbf{K}$ and $\mathbf{D}$ are encouraged to be close to each other's inverse. The computational cost of computing it scales in $\mathcal{O}(d^4)$, so that it can only be used with a reasonably small latent dimension d.

We can now write the full loss function for the model of [34] as

$$L_{cKAE}(\theta) = \alpha_{pred} L_{pred(\theta)} + \alpha_{back} L_{back}(\theta) + \alpha_{ae} L_{ae}(\theta) + \alpha_{con} L_{con}(\theta), \tag{2.22}$$

where L_{pred} and L_{ae} are referenced in equations (2.9) and (2.12). Again, one can note that this loss function does not feature the linearity loss term from equation (2.13). As we shall mention in the experiments of section 2.2.2.a, using a linearity loss term for this model can enhance its generalization ability. The main goal of adding the loss terms from equations (2.20) and (2.21) is actually to obtain a matrix $\mathbf{K}$ which features more stable predictions through equation (2.7). This approach is quite similar to our approach presented in section 2.2.1, in the sense that they also use a criterion on the matrix $\mathbf{K}$ in order to promote its stability through the means of a loss term (rather than to constrain the stability through a direct parameterization of $\mathbf{K}$, a class of methods presented in section 2.1.4). In practice, leveraging the learnt backward dynamics from $\mathbf{D}$ might also be useful to perform more accurate backward predictions, yet the authors of [34] do not present any experiment in this sense.

Another method that will be of interest for later comparisons is a variation of the base architecture of [32], introduced in the same article. The idea of this variation is to obtain the matrix $\mathbf{K}$ through an auxiliary network which takes the encoded state as an input, rather than learning the coefficients of a matrix $\mathbf{K}$ which will be used whatever the encoded state is. More formally, the learnt matrix $\mathbf{K} \in \mathbb{R}^{d \times d}$ is replaced by a neural network $K : \mathbb{R}^d \to \mathbb{R}^{2d}$. For an input encoding $\mathbf{z} \in \mathbb{R}^d$, the output of $K(\mathbf{z})$, denoted $\lambda(\mathbf{z})$, are the real and imaginary parts of complex conjugate eigenvalues. For this purpose, d is assumed to be an even integer, and $\lambda(\mathbf{z}) = (a_1, ..., a_{d/2}, b_1, ..., b_{d/2})$, where a_k and b_k are scalars. Finally, the matrix $\mathbf{K}(\mathbf{z})$ is built as a block diagonal matrix, from which the blocks are the 2×2 Jordan blocks with parameters (a_k, b_k):

$$\mathbf{K}(\mathbf{z}) = \begin{pmatrix} \mathbf{B}_1 & & \\ & \ddots & \\ & & \mathbf{B}_{d/2} \end{pmatrix}, \quad \mathbf{B}_k = \exp(a_k \Delta t) \begin{pmatrix} \cos(b_k \Delta t) & -\sin(b_k \Delta t) \\ \sin(b_k \Delta t) & cos(b_k \Delta t) \end{pmatrix}, \tag{2.23}$$

where Δt is the discrete time step, usually set to 1. Each diagonal block $\mathbf{B}_k$ parameterizes a pair of complex conjugate eigenvalues $a_k \pm ib_k$ in the continuous formulation of the Koopman operator. This approach is especially well suited for modeling dynamical systems from which the spectrum is known to be continuous, such as the Duffing oscillator and the simple pendulum. In order to illustrate what a dynamical system with a continuous spectrum is, one can briefly discuss the case of the simple pendulum. This dynamical system, from which the state is described by its angle θ and its angular velocity $\dot{\theta}$, is always periodic. However, its period of oscillation continuously depends on the initialisation of the system [60]. Thus, depending on the amplitude of the oscillations, the simple pendulum can oscillate at any frequency in a continuous range, hence its continuous spectrum. Some dynamical systems with a continuous spectrum can be successfuly modelled by using the approach of [32] with an auxiliary network having a latent dimension $d = 2$ only, i.e. a single Jordan block. In contrast, the classical KAE approach might require more dimensions and simply represent a harmonic series expansion of the true continuous spectrum (as conjectured by [32]).

The model with an auxiliary network, however, still comes with some disadvantages, which are linked to an overall lack of flexibility. For example, when predicting T steps ahead for a given initial condition, one cannot simply multiply the latent state by $\mathbf{K}^T$ as done by leveraging equation (2.7). Instead, one has to naively go through the auxiliary network T times. In addition, using the continuous formulation for a Koopman autoencoder with an auxiliary network is much less convenient than with a single matrix $\mathbf{K}$ since one cannot simply compute the matrix logarithm of $\mathbf{K}$ and bring it to the appropriate power anymore: instead one must compute the matrix logarithm of every matrix $\mathbf{K}(\mathbf{z})$ that is returned by the auxiliary network at each time step. Apart from these practical considerations, a more fondamental issue arises: when the model computes a matrix $\mathbf{K}(\mathbf{z})$ for each input rather than a single matrix $\mathbf{K}$, it is more prone to overfitting to its training data. This means that it can overfit to a certain distribution of the possible initial conditions of a dynamical system, but also to a given sampling frequency when training on regularly-sampled data (which corresponds to the vast majority of training setups for dynamical systems modelling). In section 2.2.1, we will show that, although a KAE with an auxiliary network is very good at modelling a simple dynamical system in a given sampling frequency, it does not generalize as well to higher frequencies as a properly trained classical model with a single matrix $\mathbf{K}$.

2.1.4 Constrained parameterizations of the Koopman model for guaranteed stability

In the previous section, we have described typical unconstrained means to guide the parameters of the KAE model towards a setting that hopefully enables reasonable predictions. However, it should be noted that the learnt matrix $\mathbf{K}$ has not been constrained at all since its coefficents were

theoretically free to take any possible values. In practice, some classes of matrices may be more interesting than others for building a KAE model. Notably, there is a line of work that studies such constrained parameterizations of the matrix $\mathbf{K}$ in order to obtain more stable predictions on the long term. Typically, these methods consist in restraining the matrix $\mathbf{K}$ to belong to a chosen set of matrices, on which there are theoretical guarantees, notably for the long-term stability of the model. For example, by leveraging the eigendecomposition of $\mathbf{K}$ as in equation (1.7), one can see that the (complex) eigenvalues of $\mathbf{K}$ should stay as close as possible to the complex unit circle in order to avoid an exponentially growing or decaying contribution of their corresponding eigenvectors. This advocates for restraining the search of a suitable $\mathbf{K}$ to the set of orthogonal matrices, i.e. matrices $\mathbf{K}$ such that $\mathbf{K}\mathbf{K}^T = \mathbf{K}^T\mathbf{K} = \mathbf{I}_d$.

Classically, an optimization problem with a constraint on the optimized variable can be solved via some variation of projected gradient descent [61]. This means that after each gradient step one obtains a value of the variable that does not belong to the restrained set (in our case, it is simply a matrix) and then projects this value to the restrained set (e.g. orthogonal matrices) before performing the next gradient step. However, there often exists a simple parameterization of the desired set of matrices which enables to avoid performing projections by ensuring that the optimized matrix always remains in this set anyway. For example, it is a well known fact that every orthogonal matrix with determinant 1 (i.e. special orthogonal matrix) can be expressed as the matrix exponential of a skew-symmetric matrix, and that reciprocally the matrix exponential of a skew-symmetric matrix is a special orthogonal matrix. This comes from the fact that:

— The group of orthogonal matrices of size d, $O(d)$, is a Lie group whose Lie algebra $\mathfrak{o}(d)$ is the set of skew-symmetric matrices of size d.
— The image of the exponential map of a Lie algebra always lies in the connected component of its Lie group containing the identity element, which in this case is the special orthogonal group $SO(d)$, i.e. the group of orthogonal matrices with determinant 1.
— since $SO(d)$ is connected and compact, the exponential map is moreover surjective on $SO(d)$.

We refer to Theorem 1 for a more detailed proof and to [62] for background on the Lie group theory. This property can be leveraged to learn an orthogonal matrix in an unconstrained optimization problem where the optimized variable is an antisymmetric matrix. To do this, one can define an isomorphism α from $\mathbb{R}^{d(d-1)/2}$ to $\mathfrak{o}(d)$ as

$$\alpha(\mathbf{A}) = \mathbf{A} - \mathbf{A}^T, \tag{2.24}$$

where $\mathbf{A}$ is identified as a strictly upper (or lower) triangular matrix. Then, one can redefine the set of trainable parameters of the model by replacing the d^2 coefficients of $\mathbf{K}$ by the $d(d-1)/2$

nonzero coefficients of $\mathbf{A}$, which implicitly parameterize an orthogonal matrix $\mathbf{K}$ through

$$\mathbf{K} = \exp(\alpha(\mathbf{A})). \tag{2.25}$$

In early stages of the thesis, we considered using the parameterization from equation (2.25) in order to obtain stable dynamics for our trained KAE model. This was never presented in a scientific publication since we finally decided to use an approximately (rather than exactly) orthogonal matrix $\mathbf{K}$, as explained in section 2.2.1. However, a recent work [37], published in February 2024, uses this parameterization of special orthogonal matrices through antisymmetric matrices. We will refer to this model as the Hamiltonian neural Koopman operator (HNKO). Their loss function is composed of 5 terms:

— The auto-encoding loss L_{ae} from equation (2.12)

— The linearity loss L_{lin} from equation (2.13), which is only computed for 1-time-step predictions, i.e. $T = 1$.

— They additionally introduce a learnable scalar radius parameter r, along with the loss term $L_{sphere}(\theta_\phi, r)$, which ensures that all encodings are located on a d-dimensional sphere of radius r:

$$L_{sphere}(\theta_\phi, r) = \sum_{1\leq i\leq N} \sum_{0\leq t\leq T} (||\phi(\mathbf{x}_{i,t}))||^2 - r^2)^2 \tag{2.26}$$

— The authors prove that the trajectory generated by predicting multiple time steps through (2.7) should lie in a manifold of dimension at most $d/2$. This motivates the introduction of a new set of learnable parameters $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_q)$, with $q \leq d/2$ and $\mathbf{v}_k \in \mathbb{R}^d$. Those parameters are used in order to reduce the degrees of freedom of the encoded vectors through the following loss term:

$$L_{deg}(\theta_\phi, \mathbf{V}) = \sum_{1\leq i\leq N} \sum_{0\leq t\leq T} \sum_{k=0}^{q} \langle \frac{\mathbf{v}_k}{||\mathbf{v}_k||}, \phi(\mathbf{x}_{i,t})\rangle. \tag{2.27}$$

— Additionally, in order to ensure that the parameters $\mathbf{V}$ are linearly independent and therefore able to restrict the dimension of the encodings properly, the following loss function term is introduced:

$$L_{ind}(\mathbf{V}) = \sum_{k\neq j} \langle \mathbf{v}_k, \mathbf{v}_j\rangle^2. \tag{2.28}$$

Using these 5 loss function terms, one can now define the global loss function of the model:

$$L_{HNKO}(\theta_\phi, \theta_\psi, \mathbf{K}, r, \mathbf{V}) = L_{ae}(\theta_\phi, \theta_\psi) + L_{lin}(\theta_\phi, \mathbf{K}) + L_{sphere}(\theta_\phi, r) + L_{deg}(\theta_\phi, \mathbf{V}) + L_{ind}(\mathbf{V}) \tag{2.29}$$

It is noteworthy that the global L_{HNKO} loss function contains equivalents of the linearity and

auto-encoding loss terms of the general framework of equation (2.14) but no term corresponding to the prediction loss from equation (2.9), although this is the most intuitive term. Our experiments presented in section 2.2.2.b suggest that using the prediction loss is of critical importance, yet this might not be true for every model and data. One can note that, although the prediction loss is not included in the HNKO loss function, it is still a relatively complex loss which might exhibit a complex landscape.

There are numerous other works featuring similar ways to parameterize $\mathbf{K}$ as belonging to a desired class of matrices. We will refer to these methods more briefly than to HNKO since they are less similar to our approach from section 2.2.1.

In [63], the authors propose to search for a suitable matrix $\mathbf{L}$ (in the continuous formulation, linked to the discrete advancement matrix $\mathbf{K}$ by equation (2.8)) in the set of tridiagonal matrices with negative diagonal coefficients and opposite upper and lower diagonals. This set is parameterized by

$$\mathbf{L} = \begin{pmatrix} -\sigma_1^2 & \zeta_1 & & \\ -\zeta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \zeta_{d-1} \\ & & -\zeta_{d-1} & -\sigma_d^2 \end{pmatrix} \tag{2.30}$$

where the parameters $\sigma_1, \cdots, \sigma_d, \zeta_1, \cdots, \zeta_{d-1}$ are all positive real numbers. The authors show that:

1) For any parameters $\sigma_1, \cdots, \sigma_d, \zeta_1, \cdots, \zeta_{d-1} \in \mathbb{R}_+$, the matrix $\mathbf{L}$ defined by (2.30) always corresponds to a stable evolution, as it is negative semidefinite and hence the real parts of its eigenvalues are all negative ([63] appendix A and theorem 2.1). Remind that, for a linear dynamical system in which the continuous evolution is given by a diagonalizable (in $\mathbb{C}$) matrix $\mathbf{L}$, the system is stable if and only if the real parts of the eigenvalues of $\mathbf{L}$ are nonpositive. As summarized in equation (1.19), this amounts to saying that the eigenvalues of a corresponding discrete evolution matrix $\mathbf{K}$ have modulus smaller than 1.

2) For any real square matrix $\mathbf{L}' \in \mathbb{R}^{d\times d}$ from which all eigenvalues have negative real parts, there exists a set of parameters $\sigma_1, \cdots, \sigma_d, \zeta_1, \cdots, \zeta_{d-1}$ such that the matrix $\mathbf{L}$ defined by equation (2.30) is similar to $\mathbf{L}'$ over $\mathbb{R}^{d\times d}$ (see [63] theorem 2.1). Hence, the authors prove that the parameterization from (2.30) can actually model any stable continuous evolution matrix.

Leveraging similar ideas, the authors of [36] opt for a continuous formulation of the Koopman matrix, restrained to the set of Hurwitz matrices, i.e. matrices for which all complex eigenvalues have strictly negative real parts. The authors of [36] prove in their lemma 2 that a matrix $\mathbf{K} \in \mathbb{R}^{d\times d}$ is Hurwitz if and only if there exist $\mathbf{N}, \mathbf{Q}, \mathbf{R} \in \mathbb{R}^{d\times d}$ and $\epsilon \in \mathbb{R}_+$ such that

$$\mathbf{K} = (\mathbf{N}\mathbf{N}^\intercal + \epsilon \mathbf{I}_d)^{-1}(-\mathbf{Q}\mathbf{Q}^\intercal - \epsilon \mathbf{I}_d + \frac{1}{2}(\mathbf{R} - \mathbf{R}^\intercal)). \tag{2.31}$$

Again, we refer to [36] for a proof of this result. In fact, when fixing the value of ϵ, an increasing portion of the space of Hurwitz matrices is covered as ϵ tends to 0. In practice, the authors propose to simply set ϵ to a very small value to be able to reconstruct almost any Hurwitz matrix. Thus, simlarly to [63], the authors of [36] design a loss function in which the optimized variables are the matrices $\mathbf{N}, \mathbf{Q}, \mathbf{R}$ parameterizing the Koopman matrix $\mathbf{K}$, along with the encoder and decoder that link the input space to the learnt Koopman invariant subspace. Note that, contrarily to the majority of KAE architectures, they define the encoder ϕ as a normalizing flow based on coupling layers [52] and the decoder ψ as a linear decoder, i.e. a matrix $\mathbf{C} \in \mathbb{R}^{d \times n}$.

In another recent article [35], the authors similarly seek to obtain a stable matrix, yet they express this constraint in a discrete formulation. Thus, they propose to learn a Schur stable Koopman matrix $\mathbf{K}$, i.e. a matrix from which all the complex eigenvalues lie in the unit disk. In a context where the dimension n of the dynamical system is lower than the dimension d of the learnt KIS, they jointly learn a Schur stable matrix $\mathbf{K} \in \mathbb{R}^{d \times d}$ with a set $\phi : \mathbb{R}^n \to \mathbb{R}^d$ of d observation functions, in order to respect the following 2 properties:

— for any pair $(\mathbf{x}_t, \mathbf{x}_{t+1})$ of consecutive states, $\mathbf{K}\phi(\mathbf{x}_t) = \phi(\mathbf{x}_{t+1})$. This simply corresponds to the expected behaviour of a Koopman model.

— for any state $\mathbf{x} \in \mathbb{R}^n$, $\Phi(\mathbf{x}) = \nabla\phi(\mathbf{x})^\intercal$ has full column rank, and $\nabla\phi(\mathbf{x})^\intercal\nabla\phi(\mathbf{x})$ is uniformly bounded.

In their theorem 1, the authors of [35] show that these two properties imply that the underlying dynamical system in the input space of $\mathbf{x}$ is contracting [64] [2]. Conversely, they show that if the studied dynamical system is contractive, then there exist a Schur stable Koopman matrix $\mathbf{K}$ and a Koopman mapping ϕ verifying the above two properties. We refer to [35] for the proof of this theorem.

Motivated by this result, the authors of [35] suggest to parameterize a Schur stable matrix $\mathbf{K} \in \mathbb{R}^{d \times d}$ through two matrices $\mathbf{N} \in \mathbb{R}^{2d \times 2d}, \mathbf{R} \in \mathbb{R}^{d \times d}$ such that

$$\mathbf{K} = K(\mathbf{N}, \mathbf{R}) = 2(\mathbf{M}_{11} + \mathbf{M}_{22} + \mathbf{R} - \mathbf{R}^\intercal)^{-1}\mathbf{M}_{21}, \tag{2.32}$$

with

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{pmatrix} = \mathbf{N}\mathbf{N}^\intercal + \epsilon\mathbf{I}_{2d}. \tag{2.33}$$

They show that the existence of $\mathbf{N}$ and $\mathbf{R}$ such that $\mathbf{K} = K(\mathbf{N}, \mathbf{R})$ is a necessary and sufficient condition for $\mathbf{K}$ to be Schur stable. Note that, in the original formulation of [35], the name $\mathbf{L}$ was used instead of $\mathbf{N}$, yet we changed it in order not to confuse it with the matrix logarithm of $\mathbf{K}$.

Finally, the model proposed by [35] consists in a set of 4 trainable components: $\phi, \psi, \mathbf{N}, \mathbf{R}$.

2. Informally, contracting systems have the property that all trajectories, regardless of the initial condition, converge exponentially to a single trajectory. Thus, the contraction property represents a form of stability that differs from the previously mentioned Lyapunov (asymptotic) stability.

Compared to the classical KAE setup, the direct parameterization of $\mathbf{K}$ by its d^2 parameters has simply been substituted by an indirect parameterization through equation (2.33), allowing to search among the Schur stable matrices without resorting to a constrained optimization problem.

2.2 Our contributions to the Koopman autoencoder framework

In this section, we introduce our own contributions to the Koopman autoencoder framework which was presented earlier in the chapter. Those contributions have been presented in our paper [65] published at the IEEE ICASSP 2023 conference, and further refined in a IEEE Transactions on Signal Processing article [66].

2.2.1 The orthogonality loss

2.2.1.a Description of the method

In section 2.1.4, we have motivated the reasons why using an orthogonal matrix is a good choice for modelling a dynamical system because of the long-term stability guarantees that come with this choice.

Besides from the stability argument, it has been noted by the wider machine learning community that leveraging (approximately) orthogonal matrices is a successful remedy for the well-known exploding and vanishing gradient problems [67, 68]. Therefore, it was found that favoring the orthogonality of the optimized weight matrices improves the performance of neural networks in which the computational graph is particularly deep, e.g. recurrent neural networks and residual networks [69]. We review this research branch in more detail in appendix A.

As a brief reminder, the recently published [37] restrains the search of a matrix $\mathbf{K}$ for a KAE model to the set of special orthogonal matrices. This restriction is exact, and guaranteed by a parameterization of special orthogonal matrices as matrix exponentials of antisymmetric matrices. While we have conducted experiments using this parameterization in early stages of the present thesis, we have soon opted for an alternative option in which the matrix $\mathbf{K}$ is encouraged, rather than constrained, to be (approxmately) orthogonal. Indeed, we retain the more common approach consisting of optimizing directly on the d^2 coefficients of $\mathbf{K} \in \mathbb{R}^{d\times d}$ along with the parameters of the autoencoder (ϕ, ψ). The promotion of the orthogonality for $\mathbf{K}$ is performed through an additional loss term, which is written as

$$L_{orth}(\mathbf{K}) = ||\mathbf{K}\mathbf{K}^T - \mathbf{I}_d||_F^2, \tag{2.34}$$

where $||.||_F$ denotes the Frobenius norm. Since the set of orthogonal matrices is characterized by $\mathbf{K}\mathbf{K}^\intercal = \mathbf{I}_d$, minimizing (2.34) along with some other terms should intuitively lead to a resulting matrix that is closer to orthogonality. In order to further justify this assumption, we remind

that the main reason why orthogonality is desirable is that, in many cases, we would like the eigenvalues of $\mathbf{K}$ to lie close to the unit circle. In case of exact orthogonality, the eigenvalues of $\mathbf{K}$ will indeed have a modulus of 1. Moreover, when the orthogonality is not exact but approximated in the sense of equation (2.34), the eigenvalues of $\mathbf{K}$ will indeed be close to the unit circle. Formally, since the eigenvalues of $\mathbf{K}$ and the orthogonality loss L_{orth} are all continuous functions of the matrix $\mathbf{K}$, we have that the modulus of every eigenvalue (in case of existence) converges to 1 as $L_{orth}(\mathbf{K})$ converges to zero. As announced earlier, our method builds upon the baseline loss function from equation (2.14). We propose to simply add the orthogonality loss term to this function with a weighting coefficient λ, which results in:

$$L_{total}(\theta) = L_{pred}(\theta) + \alpha L_{ae}(\theta) + \beta L_{lin}(\theta) + \lambda L_{orth}(\theta). \tag{2.35}$$

Interestingly, we did not observe the need to tune the parameters α and β, so they are set to 1 in our experiments. This means that only the parameter λ has to be tuned. It does not require a fine tuning: our usual procedure is simply to test different powers of 10 and to check, qualitatively, whether the resulting matrix $\mathbf{K}$ is close to orthogonal and enables to obtain stable predictions.

Promoting the orthogonality of $\mathbf{K}$ instead of constraining it comes with several qualitative differences. To begin with, since the matrix $\mathbf{K}$ is not constrained to be exactly orthogonal, it can model a wider class of dynamical systems. Indeed, as we show in theorem 1 below, a KAE model with a special orthogonal matrix will exhibit periodic latent dynamics and, as a direct consequence, periodic dynamics in the input space as well.

Theorem 1. *[Discrete linear systems with special orthogonal matrices lead to periodic dynamics] Let $\mathbf{K} \in \mathcal{SO}(d)$, the special orthogonal group of real invertible matrices satisfying $\mathbf{K}\mathbf{K}^T = \mathbf{K}^T\mathbf{K} = \mathbf{I}$ and with determinant equal to $+1$, and define a discrete-time dynamical system by*

$$\mathbf{z}_{t+1} = \mathbf{K}\mathbf{z}_t \tag{2.36}$$

with any initial condition $\mathbf{z}_0 \in \mathbb{R}^d$. Then there exists a continuous-time dynamical system

$$\frac{d\mathbf{z}}{dt} = \mathbf{L}\mathbf{z} \tag{2.37}$$

with $\mathbf{z}(0) = \mathbf{z}_0$, and $\mathbf{L}$ a skew-symmetric matrix such that $\exp(\mathbf{L}) = \mathbf{K}$. Besides, the dynamics are periodic, i.e. $\exists \tau \in \mathbb{R}^+, \forall t \in \mathbb{R}^+, \mathbf{z}(t+\tau) = \mathbf{z}(t)$.

Proof. Let us consider linear dynamics in a latent space given by $\mathbf{z}_t \in \mathbb{R}^d$, and

$$\mathbf{z}_{t+1} = \mathbf{K}\mathbf{z}_t \tag{2.38}$$

where $\mathbf{K} \in \mathcal{SO}(d)$. $\mathbf{K}$ belongs to the special orthogonal group, i.e. the group of invertible matrices satisfying $\mathbf{K}\mathbf{K}^T = \mathbf{K}^T\mathbf{K} = \mathbf{I}$, and with determinant equal to $+1$. First we note that the norm of the iterates $\mathbf{z}_t$ remain equal to that of the initial condition $\mathbf{z}_0$. Indeed:

$$||\mathbf{z}_{t+1}||^2 = ||\mathbf{K}\mathbf{z}_t||^2 = \mathbf{z}_t^T\mathbf{K}^T\mathbf{K}\mathbf{z}_t = \mathbf{z}_t^T\mathbf{z}_t = ||\mathbf{z}_t||^2 \tag{2.39}$$

and it is easy to see by induction that every iterate's norm is equal to $||\mathbf{z}_0||$. So the dynamics remain on a sphere of radius $||\mathbf{z}_0||$. This is equivalent to saying that the matrix group $\mathcal{SO}(d)$ acts on the sphere via matrix-vector multiplication.

Besides, as explained in section 2.1.4, any special orthogonal matrix can be written as the matrix exponential of a skew-symmetric matrix $\mathbf{L}$: $\exp(\mathbf{L}) = \mathbf{K}$. Equivalently, a skew-symmetric matrix logarithm of a special orthogonal matrix always exists. In these conditions, $\mathbf{z}_{t+1}$ is the solution to the following ODE, representing the same dynamics in continuous time:

$$\frac{d\mathbf{z}}{dt} = \mathbf{L}\mathbf{z} \tag{2.40}$$

with $\mathbf{z}(0) = \mathbf{z}_t$. We proceed to show that the dynamics generated by this ODE must be periodic. $\mathbf{L}$ is a skew-symmetric matrix, to which the spectral theorem applies: it can be diagonalized in a unitary basis, and its eigenvalues must be purely imaginary. Denoting $\mathcal{U}(d)$ the set of unitary matrices of size d, there exists $\mathbf{U} \in \mathcal{U}(d)$ such that:

$$\mathbf{L} = \mathbf{U}^*\mathbf{D}\mathbf{U} \tag{2.41}$$

with $\mathbf{D} = \mathrm{diag}(i\alpha_1, i\alpha_2, ..., i\alpha_d), \alpha_k \in \mathbb{R}$. By denoting $\mathbf{K}^\tau = \exp(\tau\mathbf{L})$ (giving $\mathbf{z}_\tau$ by matrix multiplication with $\mathbf{z}_0$), we can write:

$$\mathbf{K}^\tau = \exp(\tau\mathbf{L}) = \mathbf{U}^* \exp(\tau\mathbf{D})\mathbf{U} \tag{2.42}$$

If we write out $\mathbf{K}^\tau_{rs}$, denoting as $\mathbf{u}_r$ the r^{th} column of $\mathbf{U}$, we get

$$\mathbf{K}^\tau_{rs} = \mathbf{u}_r^* \exp(\tau\mathbf{D})\mathbf{u}_s = \sum_{k=1}^{d} u_{kr}^* \exp(i\tau\alpha_k)u_{sk}. \tag{2.43}$$

The exponential factors are periodic with periods $\frac{2\pi}{\alpha_k}$. Hence each entry of $\mathbf{K}^\tau$ is a linear combination of periodic functions. Mathematically, such a linear combination is only periodic when all the ratios between pairs of periods of the summands are rational. For all practical purposes, however, when numbers are represented with finite precision in a computer, such a linear combination can be itself seen as periodic. Finally, the same argument applies for all entries, with a common period, so the whole matrix $\mathbf{K}^\tau$ is periodic. □

Thus, constraining the (special) orthogonality of $\mathbf{K}$ (as done in [37]) appears to be especially appropriate when modelling a dynamical system which is known to be exactly periodic, while encouraging its approximate orthogonality as we did in [65, 66] seems to be more appropriate when the system is only approximately periodic. Besides, neither of these choices appears to be beneficial when modelling dynamical systems in which the energy is not conserved. Thus, in our case, the orthogonality loss term should only be included in the loss function for learning appropriate dynamical systems, otherwise it might actually detriment the accuracy of the model.

Apart from these theoretic modelling capacities, it should be noticed that the resolution of the optimization problem is also impacted by the choice of constraining the orthogonality of $\mathbf{K}$. Indeed, when restraining the search of $\mathbf{K}$ on the set of special orthogonal matrices, one might conjecture that the optimization could be more subject to falling in local minima. Indeed, with a less constrained optimization, the chosen optimization scheme might benefit from exploring matrices with eigenvalues outside of the unit circle to escape some of these local minima. Besides, even if the global minimum is attained, the best performing orthogonal matrix might not be as good as an easily attainable solution that would not be exactly orthogonal. Conversely, when $\mathbf{K}$ is unconstrained, the training procedure might be less stable since the optimization scheme might lead $\mathbf{K}$ to have relatively high eigenvalues, which will exponentially explode with the prediction steps, thus leading to diverging latent (and observed) trajectories. This is not an issue when constraining the orthogonality of $\mathbf{K}$ since the predictions through equation (2.7) will then preserve the norm of the initial encoding.

In addition, the method proposed in [37] actually consists in optimizing the continuous rather than the discrete formulation of the Koopman autoencoder. Indeed, the matrix $\alpha(\mathbf{A})$ defined in equation (2.24) is a discrete logarithm of the discrete Koopman matrix $\mathbf{K}$, as can be seen from equation (2.25). Thus, optimizing on $\mathbf{A}$ means that the optimization is performed in a continuous formulation. As will be discussed in section 2.2.2.c, this enables to handle irregularly-sampled data, but it also means that the gradient of the loss function is more complicated, as the matrix exponential function has to be differentiated through.

It should be noted that our works and the one of [37] have been performed concomitantly and that, as a consequence, there has been no quantitative comparison between the constrained and unconstrained approaches for learning a (nearly) orthogonal Koopman matrix $\mathbf{K}$ yet. We only noted that the unconstrained approach yielded higher forecasting performance on a synthetic quasi-periodic dynamical system in a preliminary experiment, which was not reported in our published articles. A more extensive comparison, on theoretical as well as practical aspects, might be an interesting direction for future work.

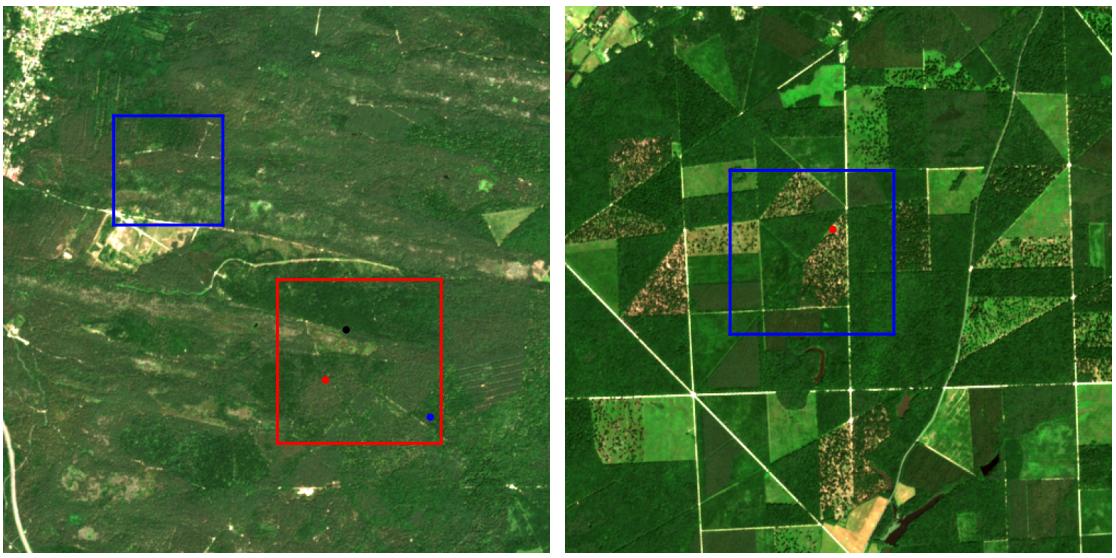

Figure 2.2 – Left: a temporally interpolated Fontainebleau image. Right: a non-interpolated Orléans image. The date for both images is 20/06/2018. Those are RGB compositions with saturated colors. The red square is the 150×150 pixel training area and the blue squares are test areas. The red, blue and black dots mark pixels that will intervene in subsequent figures.

2.2.1.b Experiments with the orthogonality loss

In order to support our claim that using an orthogonality loss term in the cost function can improve the stability of the model, we now present an experiment on Sentinel-2 image time series. We have at disposal times series of two separate areas: the forest of Fontainebleau, on which we train our models, and the forest of Orléans, which is used at test time in order to assess the capacity of the models to generalize to different initial conditions. We refer to section 1.4 for a detailed description of these datasets. In figure 2.2, we show sample images from these datasets, where the spatial areas used for training and testing the models are marked by squares as detailed in the caption. These images result from a simple interpolation that has been performed on the original data as a pre-processing step in order to work with complete time series.

In this experiment, we solve a forecasting task at the scale of a pixel. The input to the models is a reflectance vector $\mathbf{x}$, corresponding to the initial state of the pixel. The reflectance vectors are of size $d = 20$, and are composed of the reflectance values of $L = 10$ spectral bands along with the corresponding discrete temporal derivatives of these reflectance values. Intuitively, including these derivatives in the state vector facilitates the short-term forecasting task. In addition, this intuition is justified by Takens' theorem [70], which, informally, states that the evolution of a dynamical system gets more and more predictable when we know more time lags from an observed variable of the system. A well-known illustration of this theorem is the simple pendulum, for which the evolution of the state cannot be predicted from the angle only. A state composed

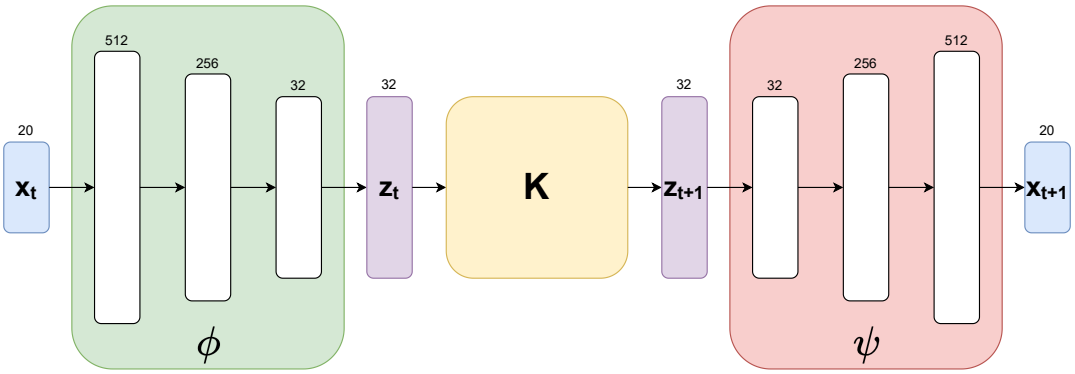

Figure 2.3 – A graphical representation of our Koopman autoencoder for the Sentinel-2 image time series, with the sizes of each vector and linear layer explicited. The white rectangles represent linear layers, while the blue and purple rectangles represent vectors.

vector of a concatenation of the angle of the pendulum and its velocity, however, is a complete description of the state of the pendulum from which one can analytically predict the evolution. Thus, such delayed embeddings appear to be especially useful when the observed vector $\mathbf{x}$ does not fully describe the state of the dynamical system at hand. From this input, the tested models are able to generate predictions for an arbitrary number of subsequent states, in a sequential fashion. We will thus study their long-term forecasting performance on times unseen during training.

For our Koopman autoencoder model, the encoder ϕ is a multi-layer perceptron with two hidden layers of size 512 and 256, and the decoder ψ symmetrically has two hidden layers of size 256 and 512. The latent space and matrix $\mathbf{K}$ have a size of $d = 32$. These architectural elements are graphically represented in figure 2.3, which is an instantiation of figure 2.1 where the sizes of the linear layers are fixed. We train models with this architecture with and without the orthogonality loss term from equation (2.34) in order to assess its influence on the trained models. These models are compared with a long short-term memory [71] (LSTM) model with hidden size 256, which, like ours, has around 3×10^5 parameters. In addition, in order to assess the importance of our model's nonlinear embedding, we compare against a baseline linear model which assumes that ϕ and ψ are simply identity functions in equation (2.7), thus consisting of only a matrix $\mathbf{K} \in \mathbb{R}^{2L \times 2L}$. This baseline is similar to dynamic mode decomposition [23], yet we observed that computing $\mathbf{K}$ as the closed-form least-squares solution for 1-step prediction leads to poor long-term predictions, and we found that an accurate long-term reconstruction was only possible with a high order delayed embedding [72]. To circumvent this issue, we trained this model to long-term prediction with automatic differentiation, like the other models of this benchmark. This model will be referred to hereafter as long term DMD.

The models are trained on the interpolated version of the Fontainebleau dataset, containing

$T_{test} = 342$ images, of which we use the first $T_{train} = 242$ for training. Remember that the time step between two consecutive images is of 5 days, so that the complete time series has a duration of 1710 days while the training time series has a duration of 1210 days. The spatial area that is used is contained in the red square from figure 2.2. Note that we do not train the models on T_{train}-time-steps pixel reflectance time series but on time series of length $T = 100$ extracted from the data. In this way, the models learn to predict from any initial condition rather than just from the initial time of the dataset. The set of 100-time-steps time series that are used for training is fixed and organised in 512 batches of size 512, which are always passed through the models in a deterministic order. This enables to retrain the models and to obtain the exact same results as ours with the available code [3]. We use as validation loss the mean-squared error of a T_{train}-step prediction from the initial state. Thus, the validation loss is computed for a forecast with a longer time span than in the training loss, but still on the same set of data. Since the models tend to overfit to the T-step prediction task, we use the validation loss as an early stopping criterion, and always save the model that minimizes it. All of the studied models are trained with this procedure on the exact same data.

All models are evaluated on the task of predicting the state from times T_{train} to T_{test} using only one delayed embedding at time 1. Their performance are measured through the mean squared error (MSE) averaged over all times, pixels and spectral bands in two areas: the Fontainebleau training area and a test area in the forest of Orléans, marked by a blue square in figure 2.2. Thus, the Fontainebleau forecasting performance only reflects the ability of the models to extrapolate in time while the Orléans testing area also tests the ability of the model to transfer to different initial conditions. For the Orléans area, we use the original irregular time series, which means that the models are tested only on true data. Since it is not possible to use a delayed embedding in this context, we make a rough estimate of the finite difference derivative from the first two available snapshots. All results are displayed in table 2.1. In figure 2.4, we show a particular example for a given pixel, where we plot the predictions of the best instance of each model along with the groundtruth reflectance over time. For all models but the long term DMD, we train 5 different instances with different random parameterizations, and we report the mean and standard deviation of the MSE over these 5 instances. We also show the MSE of the best instance from each model in parenthesis for both datasets. For long term DMD, we noticed that initialising $\mathbf{K}$ with the identity matrix gave better performance than any random initialisation, which is why we only report this result, with no standard deviation.

From the reported results, one can see that the impact of the orthogonality loss is higher on the test Orléans area, which shows that this loss term is especially useful for transferring to new data. Our model with the orthogonality loss is the best on the Fontainebleau area and second best in the Orléans area. The LSTM model performs almost on par with ours on the training area, yet

3. `github.com/anthony-frion/Sentinel2TS`

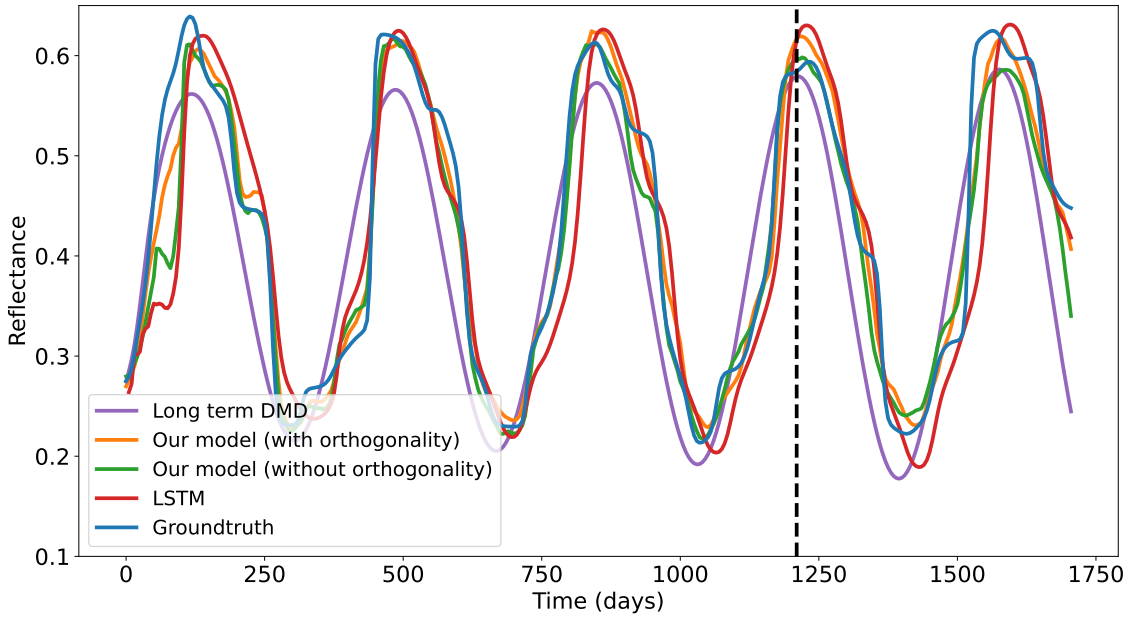

Figure 2.4 – Forecasting results with different dynamical priors for the reflectance of the B7 band (in near infrared). The vertical line marks the limit between the training and the test data. The pixel of interest is marked by a black dot on the Fontainebleau image from figure 2.2.

Table 2.1 – Forecasting MSE ($\times 10^{-3}$) for different areas and methods. For all models except long-term DMD, the uncertainty is computed over 5 instances with different initialisations of the parameters, and the MSE of the best instance is indicated in parentheses.

	Fontainebleau	Orléans (irregular data)
Our model (with orthogonality)	1.98 ± 0.20 (**1.76**)	10.35 ± 0.97 (**8.93**)
Our model (no orthogonality)	2.15 ± 0.28 (1.85)	12.06 ± 0.76 (10.66)
LSTM	2.08 ± 0.11 (1.99)	12.78 ± 1.62 (10.95)
Long term DMD	4.32	9.79

its performance is much worse on the test Orléans area. This can be explained by the fact that the underlying evolution of the LSTM is nonlinear, which makes it a more complex model which is more prone to overfitting on its training data. Long term DMD is able to roughly approximate the periodic pattern of the data, yet its limited capacity makes it unable to capture the precise relationship between the initial condition and the long-term behavior or to fit a complex periodic pattern. However, its simplicity also makes it a very good choice when transferring to a test area since it is not sensitive to overfitting (relatively high bias, but also relatively low variance). It even outperforms the mean error of our model with orthogonality loss in this case: in detail, only 2 of the 5 instances of our model trained with an orthogonality regularisation outperform long term DMD on the test Orléans area in this setting. Indeed, long term DMD suffers less from the shift of the distribution of the data than the other models. However, in chapter 3, we will prolong this experiment by employing the trained models as dynamical priors in a data assimilation scheme. We will show that our Koopman autoencoder performs far better as a dynamical prior than any of the other benchmarked methods.

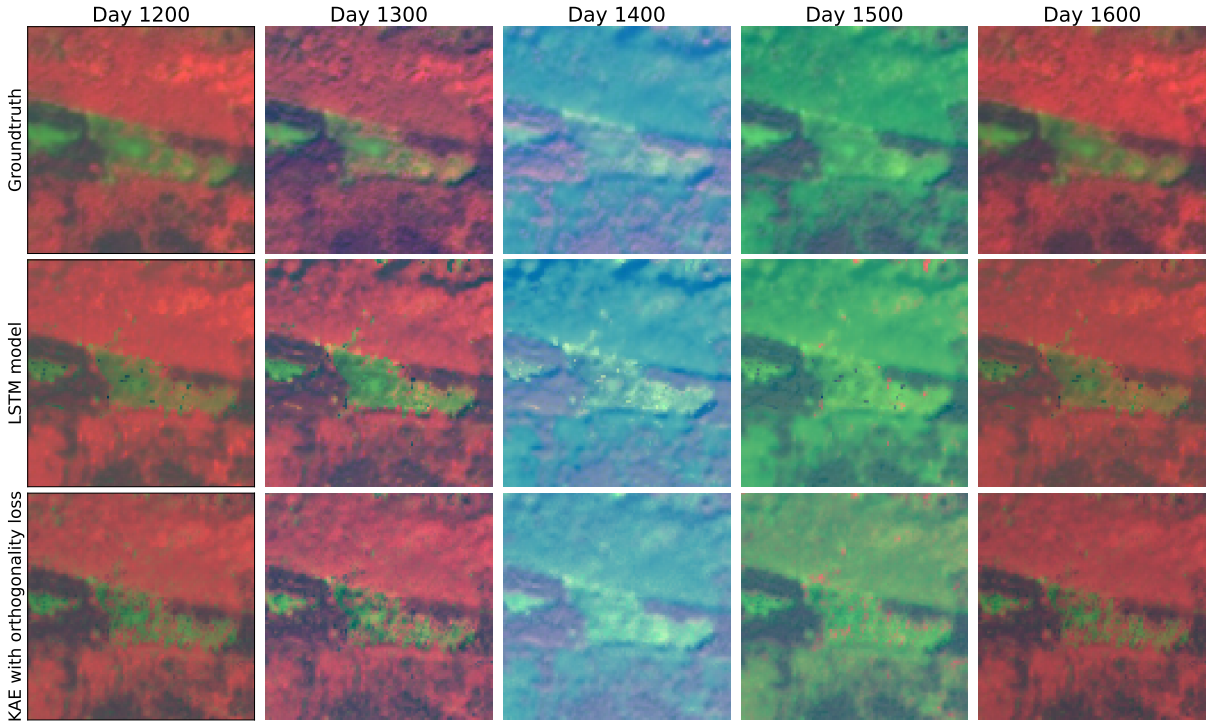

Figure 2.5 – Top: groundtruth images of Fontainebleau, corresponding to test times. Middle: predictions from day 0 made by the best trained LSTM model. Bottom: Predictions from day 0 made by the best trained KAE model with orthogonality loss. The colors result from a 3-dimensional principal component analysis (PCA) of the 10 spectral bands performed globally on all the Fontainebleau data. This is much more informative than an RGB composition, mainly because vegetation is very reflective in the near-infrared domain.

Finally, in figure 2.5, we show forecasting results on the top-left 100x100 sub-square included in the red square of the Fontainebleau image in 2.2. One can see that the groundtruth data are far more spatially smooth than the predictions are, which is due to the fact that both models are performing pixelwise predictions. Although the spatial artifacts are more severe for the LSTM model, the KAE model with orthogonality loss would also clearly benefit from spatial information in order to perform more spacially coherent forecasts. We will experiment methods to include spatial information in the forecasts in chapter 3.

2.2.2 Training a model on scarce and irregular data

A second important contribution that we bring to the KAE framework is that we leverage the continuous formulation of linear dynamical systems, outlined in section 1.2, to be able to train a KAE model on scarce or irregular data and then to use it for obtaining a continuous model.

Here, it is important to note that the continuous formulation of linear dynamical systems, and more specifically of the Koopman operator, is a very well-known property. In particular, many articles use the continuous formulation to directly learn the derivative of a dynamical system rather than a discrete advancement operator [63, 36]. This is done either by leveraging data snapshots of this known derivative or by resorting to finite differences when having couples of datapoints sufficiently close in time. Yet, those choices are quite restrictive, and rely on the fact that these methods are tested in an ideal setting in which the data is regularly sampled at a sufficiently high frequency to get a very good view of the dynamics. [33] contains comments on how the continuous formulation of an approximated Koopman operator could be used to learn a model from irregularly-sampled data, yet it does not present any experiment to illustrate its discussion. To the best of our knowledge, our work [65] was the first to present experiments on learning a discrete neural Koopman model from training data sampled at a low frequency and then resorting to the continuous formulation of this model in order to perform predictions at a higher test frequency. This experiment is presented in section 2.2.2.a. Subsequently, in [66], we leveraged similar ideas in order to learn from irregularly-sampled data, and presented an experiment on naturally incomplete satellite image time series. The method and the experiment are detailed in section 2.2.2.b. It should be noted, however, that this training context was still relatively simple since the data consisted in regularly-spaced (in time) data samples where some points are missing. Hence, all time increments between two consecutive samples are multiples of a reference duration, which is simply the frequency at which the data is theoretically sampled. In a more general context, the time separating consecutive samples may take arbitrary values. We have not presented any experiment for this more difficult context in our published works yet, but we still presented a detailed methodology to handle this kind of data in [66]. It will be presented in section 2.2.2.c, along with a new experiment on a toy dataset to validate this methodology.

2.2.2.a Upsampling the predictions to a higher frequency after training at a low frequency

Here, we leverage the inherent duality between the continuous and the discrete formulation of linear dynamical systems in the particular context of Koopman autoencoders. Although this property of linear dynamical systems is a very well known fact, we are not aware of works other than ours which use it to assess the capacity of Koopman autoencoders to predict on time frequencies different from the one on which they have been trained. In our opinion, such tests are essential in order to get an idea of the robustness and the physical soundness of the trained model. For this reason, in [65] we developed the following procedure:

1) Train a KAE on a set of trajectories sampled at a low frequency ω. From this training, we retrieve the components $\mathbf{K}$, ϕ and ψ of the model. Note that the trained advancement matrix $\mathbf{K}$ corresponds to one time step at frequency ω, or equivalently at period $1/\omega = \tau$.

2) Compute the matrix logarithm $\mathbf{L}$ corresponding to $\mathbf{K}$. This is done by computing the complex eigendecomposition of $\mathbf{K}$, which is written

$$\mathbf{K} = \mathbf{U}^{-1}\mathbf{D}\mathbf{U} \tag{2.44}$$

where $\mathbf{U} \in \mathbb{C}^{n\times n}$ and $\mathbf{D} \in \mathbb{C}^{n\times n}$ is a diagonal matrix. Then, one simply has to compute the principal logarithm of each diagonal coefficient of $\mathbf{D}$, which results in

$$\mathbf{L} = \mathbf{U}^{-1}\log(\mathbf{D})\mathbf{U}. \tag{2.45}$$

3) Given a higher sampling frequency $\bar{\omega}$, one can then simply write the matrix corresponding to a discrete advancement at frequency $\bar{\omega}$ (or time step $\bar{\tau}$) as

$$\bar{\mathbf{K}} = \exp\left(\frac{\omega}{\bar{\omega}}\log\mathbf{K}\right) = \exp\left(\frac{\omega}{\bar{\omega}}\mathbf{L}\right) = \exp\left(\frac{\bar{\tau}}{\tau}\mathbf{L}\right). \tag{2.46}$$

4) Finally, given an initial condition $\bar{\mathbf{x}}_{i,0}$, one can compare the groundtruth time series $(\bar{\mathbf{x}}_{i,t})_{0\leq t\leq T}$ sampled at frequency $\bar{\omega}$ to the predictions given by

$$\hat{\mathbf{x}}_{i,t} = \psi(\bar{\mathbf{K}}^t\phi(\bar{\mathbf{x}}_{i,0})). \tag{2.47}$$

We apply this method on the simple pendulum dynamical system. This system can be fully described using two variables: the angle of the pendulum $x_1 = \theta$ and its angular velocity $x_2 = \dot{\theta}$. The evolution of $\mathbf{x}_1$ and $\mathbf{x}_2$ are governed by the following system of ordinary differential equations:

$$\begin{aligned}\dot{x}_1 &= x_2,\\ \dot{x}_2 &= -\sin x_1.\end{aligned} \tag{2.48}$$

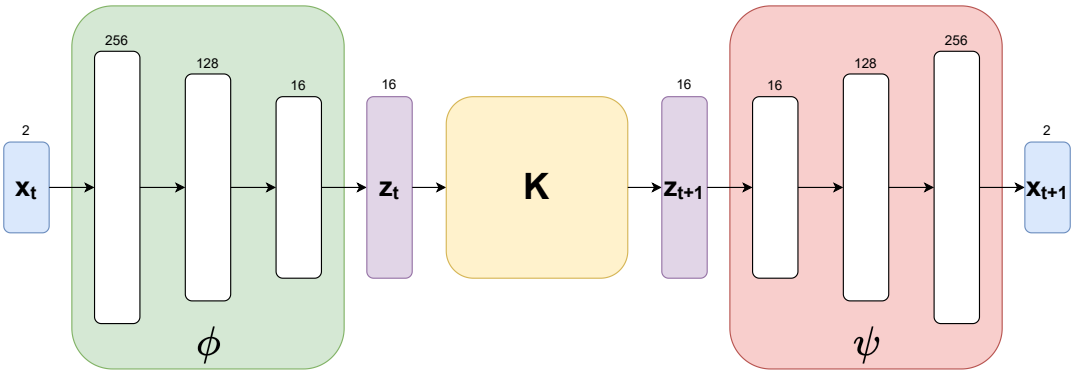

Figure 2.6 – A graphical representation of our Koopman autoencoder for the simple pendulum time series, with the sizes of each vector and linear layer explicited. The white rectangles represent linear layers, while the blue and purple rectangles represent vectors.

Although this dynamical system is apparently simple, it is difficult in practice to obtain a compact solution for it. For any initial condition, this system induces a periodic behaviour. Moreover, since the period of the system depends continuously on the amplitude of the oscillations, the simple pendulum exhibits a continuous spectrum.

In practice, we model the simple pendulum through our own variation of the Koopman autoencoder, with or without the orthogonality loss term introduced in section 2.2.1. We sample 100 trajectories, each with a duration of 10 seconds. Each of these trajectories starts from an angle sampled in a wide uniform distribution, with no initial velocity. For each trajectory, we retain a high-frequency version sampled at $\bar{\omega} = 100$ Hz (hence containing 1000 datapoints) and a low-frequency one sampled at $\omega = 5$ Hz (hence containing 50 datapoints). While the high-frequency time series qualitatively gives a complete understanding of the dynamical system's behaviour, the low-frequency one gives a lacunar view of it. Thus, the main goal of the experiment is to assess whether one can retrieve a good approximation of the dynamical system at frequency $\bar{\omega}$ when training only at a frequency ω.

We train a Koopman autoencoder with the following architecture: the encoder network ϕ is a multi-layer perceptron (MLP) with 3 layers, from which the hidden sizes are 256 and 128, with ReLU nonlinearities and a final Koopman embedding of dimension $d = 16$. The decoder also is a 3-layer MLP with symmetric hidden sizes of 256 and 128. Again, we graphically represent this architecture on figure 2.6. The only difference with the model for Sentinel-2 data that was represented in figure 2.3 is that the linear layers contain less parameters. In particular, the latent state is in this case of size 16 instead of 32. The cumulated number of parameters for ϕ, ψ and $\mathbf{K}$ is about 70 000. The model is tested both with and without the orthogonality loss term from equation (2.34).

As a point of comparison, we use the architecture proposed in [32] with an auxiliary network

Table 2.2 – Mean squared error averaged over all points from all high-frequency testing trajectories. (HF) and (LF) stand respectively for "high frequency" and "low frequency" training data.

Dataset	Our method	DeepKoopman	Our method (no orthogonality loss)
Pendulum (HF)	5.38×10^{-4}	$\mathbf{1.41 \times 10^{-4}}$	4.42×10^{-2}
Pendulum (LF)	$\mathbf{6.82 \times 10^{-4}}$	1.62×10^{-3}	8.02×10^{-4}

computing a new matrix at every iteration $\mathbf{K}$ according to the latent state. This architecture, which we refer to as DeepKoopman, is expected to perform well on simple pendulum data since it is able to model its continuous spectrum and has been shown by [32] to produce excellent predictions on this precise dataset. Since it is much more difficult with this method to make predictions at a different frequency from the one it has been trained on, we will simply perform a linear interpolation between the points predicted at the low training frequency. We interpolate in the latent space since it gives slightly better results than interpolating directly in the observation space.

The results of this experiment are presented in table 2.2. As expected, the DeepKoopman model is more accurate than ours in the ideal high-frequency sampling context. This is due to the auxiliary network which enables to apply different linear transformations to different latent states while our framework always applies the same transformation. However, our framework outperforms this method in low frequency settings, mostly because it is intrinsically able to upsample its prediction to a higher frequency, resulting in a nontrivial interpolation which is much better than the agnostic linear interpolation. One can visually assess the quality of the interpolation for the compared models on figure 2.7 for predictions from 4 different initial conditions. In addition, we plot the squared errors of both models over time for 2 of these initial conditions in figure 2.8. Interestingly, one can see that the DeepKoopman model predicts well on points corresponding to its training frequency, but makes large errors in between those. In contrast, our model does not seem to perform worse between the points corresponding to its training frequency than on these particular points.

However, training a Koopman autoencoder without the orthogonality loss from equation (2.34) notoriously fails at modeling the pendulum system in high frequency, which shows that the orthogonality constraint is crucial to keep the predictions stable when modeling very long time series, in particular for conservative models such as this one. It can still be noted that the Koopman autoencoder with no orthogonality loss manages to model the dynamical system when trained on low frequency data: this is due to the shortened length of the training time series (50 points instead of 1000). Indeed, with such shortened time series, the long-term stability of the discrete model is not as critical as in the high-frequency setting.

As a second experiment, we now work on another dataset which was originally proposed in [32]. This dataset consists in a 3-dimensional dynamical system arising from fluid dynamics.

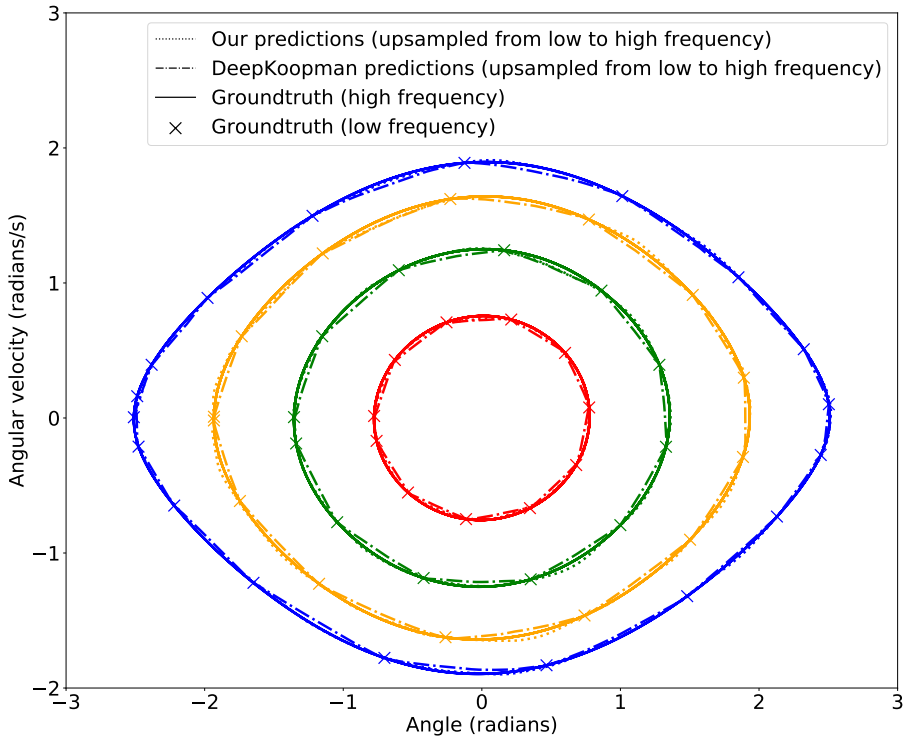

Figure 2.7 – Comparison of our model (trained with the orthogonality loss) to the DeepKoopman model, on the task of predicting long-term pendulum trajectories from an initial condition. Each color denotes a different initial condition.

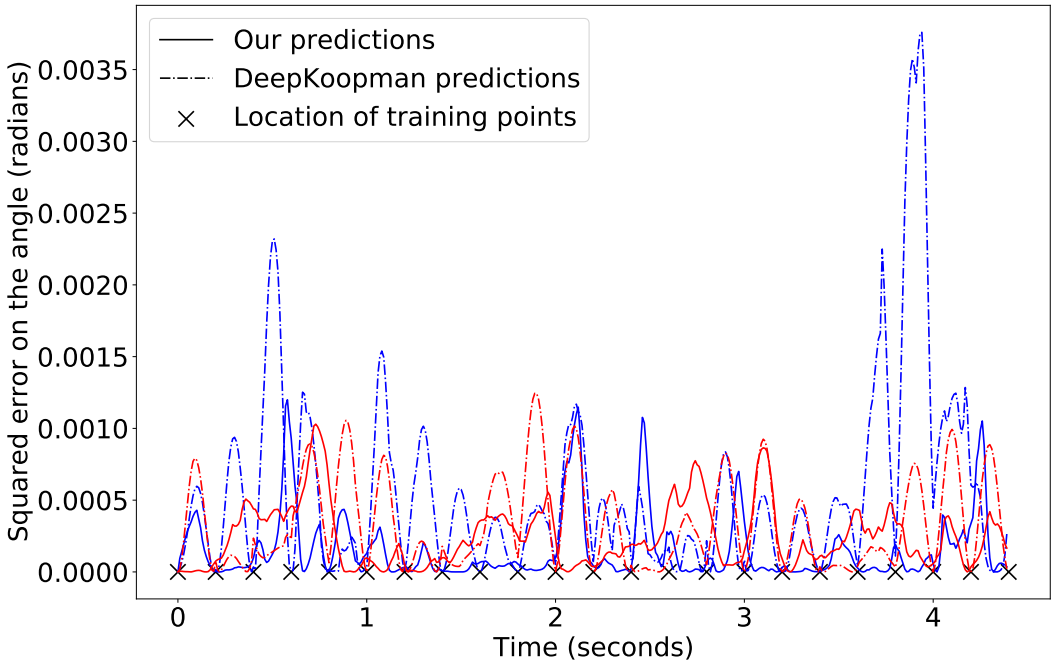

Figure 2.8 – Comparison of our model (trained with the orthogonality loss) to the DeepKoopman model, on the task of predicting long-term pendulum trajectories from an initial condition. Each color denotes a different initial condition.

The nonlinear fluid flow past a cylinder with a Reynolds number of 100 has been a fluid dynamics benchmark for decades, and it was proven by [73] that its high-dimensional dynamics evolves on a 3-dimensional attractor with the model:

$$\begin{aligned} \dot{x} &= \mu x - y - xz \\ \dot{y} &= \mu y + x - yz \\ \dot{z} &= -y + x^2 + y^2. \end{aligned} \tag{2.49}$$

This dynamical system is not periodic, yet it exhibits a stable limit cycle and an unstable equilibrium point at the origin.

In our experiments, we use the training and test data from [32], which have been generated by numerically integrating equations (2.49). Our model has the same architecture as in the previous experiment on simple pendulum data. We compare it against the consistent Koopman autoencoder model [34] (cKAE), for which we use the same architecture, the only difference being the backward evolution matrix $\mathbf{D}$ of the same size as $\mathbf{K}$. We also compare against the DeepKoopman model [32], for which we use the hyperparameters reported in [32].

In order to show the ability of our architecture to model a continuous dynamical system even when trained on discrete data, we choose our reference data to be of the same high frequency as

originally proposed by [32], i.e. $\bar{\omega} = 50$ Hz. The models are trained with 5 different frequencies ω, ranging from 2.5 Hz to $\bar{\omega} = 50$ Hz. In detail, those frequencies are 2.5 Hz, 5 Hz, 10 Hz, 25 Hz and 50 Hz

Note that, for this experiment, our models are trained without the orthogonality loss term from equation (2.34) since the considered dynamical system at hand, having no quasi-periodic component, is not well adapted for this regularisation. We compare our results against DeepKoopman [32] and cKAE [34] models trained on the same data and for which we similarly computed $\bar{\mathbf{K}}$. This was not proposed in their original papers, but a more naive interpolation method, such as a linear interpolation in the latent space, yields unsatisfactory results. Note that both cKAE and our model feature a single matrix $\mathbf{K}$, and therefore only one matrix $\bar{\mathbf{K}}$ needs to be computed for these models. In contrast, DeepKoopman computes a new matrix $\mathbf{K}$ for each state and time increment, and one must compute a new $\bar{\mathbf{K}}$ for each of those, leading to a far greater amount of computation at inference. In addition, the DeepKoopman model is significantly more costly to train than the other models due to the necessity to compute a new matrix $\mathbf{K}$ through the auxiliary network at each time step. cKAE is itself slower to train than our model because of its consistency loss from equation (2.21), which requires computing $2d$ (where d is the latent dimension) additional matrix multiplications for each training step. Concretely, in our configuration, the duration of a training epoch in frequency $\omega = 50$ Hz is approximately 3.4 seconds for DeepKoopman, 1 second for cKAE and 0.9 second for our model.

Table 2.3 – Mean squared errors ($\times 10^{-6}$) for 50 Hz interpolation of fluid flow dynamics with various training frequencies

Training frequency	DeepKoopman	cKAE	Our method
50 Hz	$\mathbf{1.16 \pm 0.05}$	1.31 ± 0.23	1.51 ± 0.19
25 Hz	1.30 ± 0.09	1.84 ± 0.57	$\mathbf{1.20 \pm 0.13}$
10 Hz	1.59 ± 0.01	2.92 ± 1.01	$\mathbf{1.42 \pm 0.19}$
5 Hz	37.4 ± 44.2	18.7 ± 6.25	$\mathbf{1.56 \pm 0.03}$
2.5 Hz	294 ± 578	971 ± 403	$\mathbf{2.02 \pm 0.42}$

We report in table 2.3 the results obtained with various training frequencies ω. All training runs are repeated 5 times with different initialisations of the model's parameters, and we report the mean and standard deviation of the errors over these 5 runs. The standard deviation corresponds to the uncertainty associated to the initial parameterization of the model. The results show that the quality of the high-frequency predictions of our model depends very little on the training frequency. The MSE for DeepKoopman and cKAE are similar to ours for the highest training frequencies, and the DeepKoopman model even has the lowest error when trained on the test frequency. However, the errors of these two models increase faster as the training frequency decreases. We emphasize that all instances of all models excel at predicting in their training

frequency. In particular, for every training frequency, the DeepKoopman model yields the best performance when testing it on the frequency on which it was trained. However, the main difficulty in this experiment is to obtain models that are also good at interpolating to a high frequency. In this regard, at training frequencies 5 Hz and 2.5 Hz, some instances of DeepKoopman achieve a relative success with a MSE in the order of 10^{-6} while some remarkably fail with a MSE in the order of 10^{-3} or 10^{-4} (hence the very high standard deviations). However, none of the instances is as good as the mean performance of our model for these frequencies. cKAE, in contrast, consistently fails to interpolate in the lowest frequencies. The main reason for this failure seems to be the absence of a linearity term in their loss function. Indeed, a quick (unreported) experiment enabled us to obtain an error in the order of 10^{-6} with training frequency 2.5 Hz by training a cKAE instance with an additional forward linearity loss term. [4]

Our model, however, successfully combines the information of many low-resolution time series to construct a faithful continuous representation of the dynamics. On Figure 2.9, we show the interpolation by the best instance trained at $\omega = 2.5$ Hz for each of the three models on a test trajectory. It appears that, although all models are able to fit the groundtruth points corresponding to the low training frequency, they are not all able to interpolate to a higher frequency. Only our model performs as well for interpolating as for fitting the points corresponding to its training frequency.

2.2.2.b Training from regularly-sampled data with missing values

We now discuss the training of Koopman autoencoder models on irregularly-sampled data. For now, we focus on the case where the irregularly-sampled data simply come as the result of a regular sampling where some values are missing. This case is relatively common in practice, since the missing values might correspond to measurements which have been acquired but intentionally put aside as they were obviously corrupted. For example, in the case of Sentinel-2 satellite image time series, the data come with a regular sampling pattern with a period of 5 days, yet a significant part of the images are unusable due to the presence of clouds between the sensor and the surface of the Earth.

In such a context, the data may be denoted as $(\mathbf{x}_{i,t})_{1\leq i\leq N, 1\leq t\leq T}$, with the binary observation variable $(\mathbf{H}_{i,t})_{1\leq i\leq N, 1\leq t\leq T}$ being so that $\mathbf{H}_{i,t} = 1$ if $\mathbf{x}_{i,t}$ is actually observed and 0 otherwise. The indexes may be rearranged so that $\mathbf{x}_{i,0}$ is always observed. Then, one can very easily multiply the terms of the prediction, auto-encoding and linearity loss terms from equations (2.9), (2.12) and (2.13) by the corresponding mask terms $\mathbf{H}_{i,t}$ to adapt for the presence or absence of the

4. This result tends to suggest that, although designing a loss function without a linearity term from equation (2.13) might improve the performance in very specific easy settings, it leads to models that are considerably less robust to changes in either the input data or the expected tasks. Therefore, we strongly recommend using all three terms from equation (2.14) as a baseline for designing any new loss function for a Koopman autoencoder model.

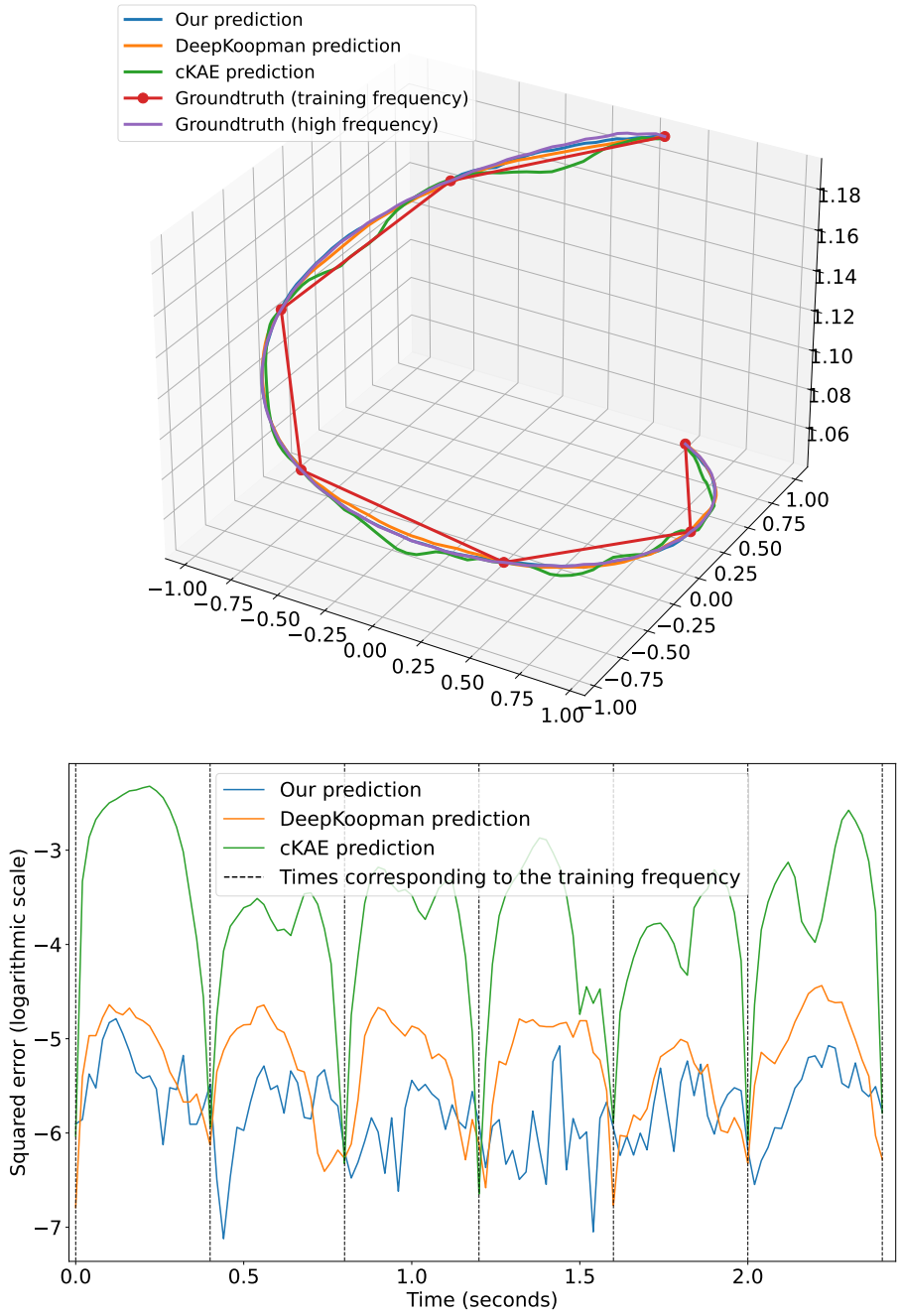

Figure 2.9 – Upsampling experiment on fluid flow data. We learn a model on low-frequency (2.5 Hz) data and then use the continuous representation to make a high-frequency ($\bar{\omega} = 50$ Hz) prediction which we compare to the groundtruth on test trajectories. Top: the three tested models compared to a groundtruth trajectory. Bottom: corresponding mean squared errors over time, in a logarithmic scale.

corresponding observations. Concretely, this gives:

$$L^H_{pred}(\theta) = \sum_{1\leq i\leq N} \sum_{1\leq \tau\leq T} \mathbf{H}_{i,\tau} ||\mathbf{x}_{i,\tau} - \psi(\mathbf{K}^\tau \phi(\mathbf{x}_{i,0}))||^2, \tag{2.50}$$

$$L^H_{ae}(\theta) = \sum_{1\leq i\leq N} \sum_{0\leq t\leq T} \mathbf{H}_{i,t} ||\mathbf{x}_{i,t} - \psi(\phi(\mathbf{x}_{i,t}))||^2, \tag{2.51}$$

$$L^H_{lin}(\theta) = \sum_{1\leq i\leq N} \sum_{1\leq \tau\leq T} \mathbf{H}_{i,\tau} ||\phi(\mathbf{x}_{i,\tau}) - \mathbf{K}^\tau \phi(\mathbf{x}_{i,0})||^2, \tag{2.52}$$

Thus, using the unchanged orthogonality loss function presented in section 2.2.1, we obtain the following complete loss function:

$$L^H_{total}(\theta) = L^H_{pred}(\theta) + \alpha L^H_{ae}(\theta) + \beta L^H_{lin}(\theta) + \lambda L_{orth}(\theta). \tag{2.53}$$

Again, α and β are set to 1 in our experiments while λ is tuned to be a suitable power of 10.

In order to test the ability of our Koopman autoencoder model from section 2.2.1 in this unusual setup, we leverage Sentinel-2 time series. These time series naturally correspond to the considered context, as a high proportion of the images are usually corrupted by clouds which prevent from seeing the surface. Moreover, the pattern of cloud corruption is a complex one, yet one can derive two principles from it in our case: 1) the winter is in average more cloudy than the summer, thus leading to a higher rate of cloud corruption, and 2) the images tend to be corrupted for multiple time steps in a row, and then uncorrupted for multiple time steps in a row. Compared to synthetic data in which one could have chosen to randomly mask the images using, e.g., a Bernoulli random variable, this leads to more complicated data in practice. Regardless, we still trained Koopman autoencoder models on irregularly-sampled Sentinel-2 data from the forest of Fontainebleau.

We found that training on irregular data makes our model more subject to overfitting. Indeed, the model is not forced to predict a smooth evolution anymore but only to be able to correctly reconstruct some sparsely located points. Therefore, the regularisation terms from equation (2.53) seem to be of critical importance for the model to properly generalize. To quantify this importance, we perform an ablation study by testing series of models with different versions of the complete loss function proposed in equation (2.53). Concretely, we test 5 variants: the variant where all terms are used and four variants where one of the terms has been removed. For each of these variants, we trained models on the Fontainebleau data from 5 different initialisations. We then retrieved the mean and standard deviations of the mean squared errors obtained when performing assimilation-forecasting.

In a few words, assimilation-forecasting consists in minimising a variational cost where the optimized variable is the latent initial state of a pre-trained Koopman autoencoder model

prediction. The variational cost is the mean squared error between the prediction obtained by predicting from the latent initial condition and the groundtruth, averaged over all spectral bands and all time steps for which a groundtruth datapoint is available. Thus, contrarily to a naive prediction through equation (2.7), assimilation-forecasting makes use of several datapoints located on different time steps rather than of a single datapoint. For this reason, it is more robust to the potential distribution shifts or noise in the data. However, it is also prone to overfitting by nature since an excessively expressive model might produce, through assimilation-forecasting, trajectories that do very well match the available datapoints but fail to have a globally realistic structure. A complete description of assimilation-forecasting, and more generally of the usage of a trained Koopman autoencoder in conjunction with data assimilation methods, will be presented in chapter 3.

The results are presented in table 2.4. One can see that the final results on the Fontainebleau area largely depend on the model initialisation, yet both the mean and the standard deviation of the MSE are lower when using all loss terms. As for the test Orléans area, although the results are not as clear as on the training area, they tend to suggest that the model performs slightly better on average when trained with the complete loss function, with more consistent results.

Table 2.4 – Assimilation-forecasting MSE ($\times 10^{-3}$) of models trained on irregularly-sampled data with different loss functions

	Fontainebleau	Orléans
Complete loss	$\mathbf{0.699 \pm 0.130}$	$\mathbf{3.480 \pm 0.198}$
No orthogonality	0.922 ± 0.233	3.564 ± 0.301
No linearity	2.722 ± 0.576	5.454 ± 0.515
No auto-encoding	1.252 ± 0.128	3.770 ± 0.172
No prediction	3.514 ± 1.276	4.586 ± 0.232

We show qualitative results of the assimilation-forecasting of irregular data using one of our models trained on irregular data with the complete loss function in figure 2.10. We emphasize that the blue curve is only a Cressman interpolation of the groundtruth points and should not be seen as a groundtruth here. Our model fits the training points well and, in some way, performs a more realistic-looking interpolation than the Cressman method that was used to obtain the regularly-sampled data. Indeed, in some cases, the Cressman interpolation exhibits successions of stagnating regions and brutal variations, which are not physically realistic.

Interestingly, when tested on the same irregular test data, models trained on interpolated Fontainebleau data have better interpolation performance but lower forecasting performance than models trained on irregular Fontainebleau data. Using interpolation as a pre-processing step is not a trivial choice since models trained on these data will learn the interpolation scheme along with the true data. However, it can be seen as a form of data augmentation.

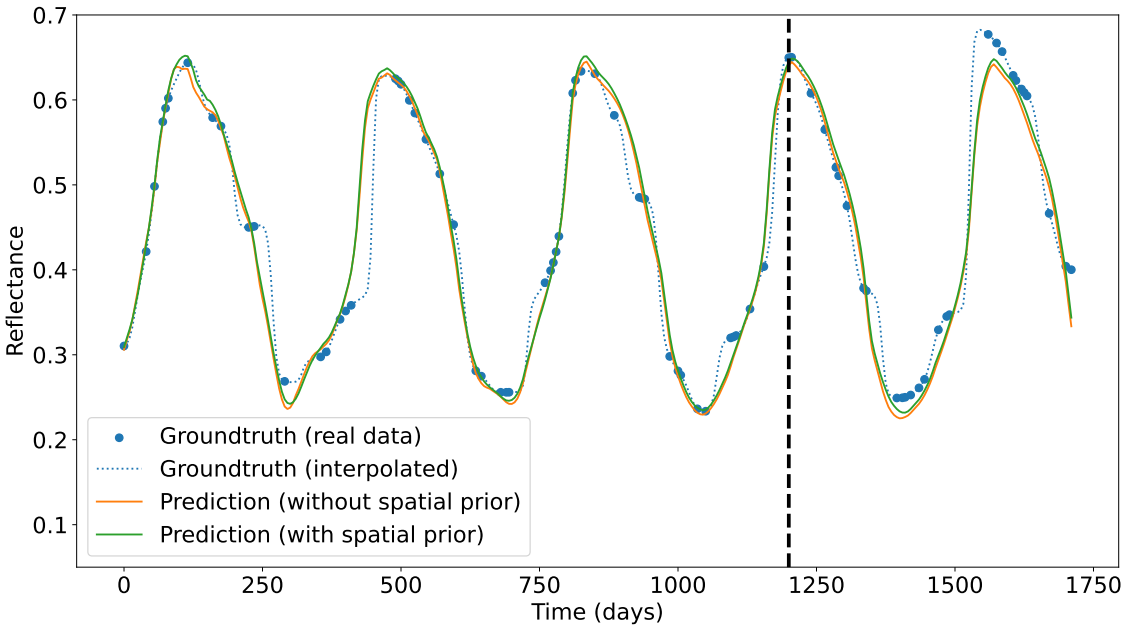

Figure 2.10 – Forecasting results on the B7 band, with irregular data from the forest of Fontainebleau. The considered pixel is marked by a blue dot on figure 2.2.

2.2.2.c Training from arbitrarily-sampled data

We now describe the hardest possible kind of time sampling. In the previous section, we examined the case where the studied irregularly-sampled data can actually be seen as a set of regularly-sampled data in which some datapoints are missing. This setting allowed us to perform the optimization with a discrete formulation, as all time increments that separated consecutive datapoints were multiples of a reference duration, which is the sampling frequency of the underlying regular sampling process. Therefore, the optimized components of our model were ϕ, ψ and $\mathbf{K}$, which are the same as in the ideal case with regularly-sampled training data.

In this section, we describe the more general case where the time increments separating the available values are completely arbitrary. Formally, for each index $1 \leq i \leq N$, we denote the trajectory $\mathbf{x}_i$ as a list of $T_i + 1$ time-value pairs $(t_{i,k}, \mathbf{x}_{i,k})_{0 \leq k \leq T_i}$. Without loss of generality, one can suppose that the pairs are ordered by increasing times, with $t_{i,0} = 0$. In order to learn from this data, one has to work in the continuous formulation of the Koopman autoencoder. Therefore, instead of learning a matrix $\mathbf{K}$ corresponding to an advancement by one discrete time step, as we have done so far, we now work with the continuous counterpart $\mathbf{L}$ of $\mathbf{K}$. Thus, considering the relation (2.8) that links $\mathbf{L}$ to $\mathbf{K}$, the fundamental equation of advancement (2.7) of the KAE model can be reformulated as

$$\psi(\exp(\tau\mathbf{L})\phi(\mathbf{x}_t)) \approx \mathbf{x}_{t+\tau}. \tag{2.54}$$

From this equation, the prediction, auto-encoding and linearity loss terms, which we introduced in a context of regularly-sampled data in equations (2.9), (2.12) and (2.13), can be rewritten as follows:

$$L^*_{pred}(\Theta) = \sum_{1\leq i\leq N} \sum_{1\leq k\leq T_i} ||\mathbf{x}_{i,k} - \psi(\mathbf{K}^{t_{i,k}}\phi(\mathbf{x}_{i,0}))||^2, \tag{2.55}$$

$$L^*_{ae}(\Theta) = \sum_{1\leq i\leq N} \sum_{0\leq k\leq T_i} ||\mathbf{x}_{i,k} - \psi(\phi(\mathbf{x}_{i,k}))||^2, \tag{2.56}$$

$$L^*_{lin}(\Theta) = \sum_{1\leq i\leq N} \sum_{1\leq k\leq T_i} ||\phi(\mathbf{x}_{i,k}) - \mathbf{K}^{t_{i,k}}\phi(\mathbf{x}_{i,0})||^2, \tag{2.57}$$

As a reminder from section 1.2, we use the slightly abusive notation $\mathbf{K}^t = \exp(t\mathbf{L})$ for any real time increment t. This notation is not widespread since the notion of a non-integer power of a matrix is not common. However, it should be noted that we "define" such non-integer powers in the exact same way as the non-integer powers of real numbers are defined, i.e. through the (matrix) exponential function. In this regard, it should be noted that, since the matrix exponential is always defined for any real square matrix, the non-integer powers of such a matrix $\mathbf{A}$ are all defined as long as $\mathbf{A}$ admits a matrix logarithm. Using these loss terms, one can rewrite the baseline loss function from equation (2.14) for the specific context of arbitrarily-sampled data as follows:

$$L^*_{total}(\theta) = L^*_{pred}(\theta) + \alpha L^*_{ae}(\theta) + \beta L^*_{lin}(\theta). \tag{2.58}$$

Again, we did not observe a necessity to tune the parameters α and β, so that we simply set them to 1. It can be noted that, in this context, the orthogonality of any discrete advancement $\mathbf{K}^t = \exp(t\mathbf{L})$ can be guaranteed by simply making sure that the matrix $\mathbf{L}$ is antisymmetric. This can be done by parameterizing $\mathbf{L}$ through another (strictly lower or upper triangular) matrix $\mathbf{A}$ such that $\mathbf{L} = \alpha(\mathbf{A})$, with the isomorphism α defined in equation (2.24). This is the choice made by [37], although with a different loss function. Alternatively, the approximate orthogonality can be obtained in a similar way as with the orthogonality loss term (2.34) on the discrete $\mathbf{K}$, by softly penalizing the symmetricity of the optimized continuous matrix $\mathbf{L}$, e.g. through

$$L^*_{orth}(\mathbf{L}) = ||\mathbf{L} + \mathbf{L}^T||^2_F. \tag{2.59}$$

Note that, by definition, $\mathbf{L}$ is antisymmetric if and only if $L^*_{orth}(\mathbf{L}) = 0$.

In fact, the above formulation also encapsulates the case where the data are regularly-sampled. Therefore, one could also try to optimize from the derivative of the Koopman system even when this is not necessary. However, this requires computing the matrix exponential $\mathbf{K} = \exp(\mathbf{L})$ after each update of $\mathbf{L}$ instead of working on $\mathbf{K}$ directly. We experimentally found that, when training on regularly-sampled data, the discrete formulation had slightly better performance than the continuous one. We conjecture that this is due to the gradient of the loss function being

more complex when performing a matrix exponential, and we recommend using the continuous formulation only when necessary.

We will now show through a preliminary experiment that the continuous formulation of the Koopman autoencoder indeed enables to train such a model from arbitrarily-sampled data in practice. In this experiment, in contrast to the other experiments from this chapter, we do not train a model on multiple data trajectories but from a simple trajectory with an arbitrary sampling pattern. Thus, we will test the interpolation as well as the extrapolation capabilities of our Koopman autoencoder model in this context where a very low amount of data is available.

First, we generate a synthetic trajectory based on the numerical integration of a system of ordinary differential equations based on the PROSAIL model [5]. This model aims at simulating the reflectance of vegetation as a function of several phenological parameters, among which the leaf structure index (leaf mass/area), the chlorophyll a and b concentration, the carotenoid concentration, the brown pigment concentration, the dry matter content, and equivalent water thickness (the latter two are kept constant here). According to [74], those parameters mostly evolve using seasonal dynamics, with two different phases. In our case, these are simulated using a predator/prey model [75] over two variables, which are then affinely transformed to roughly match the data in [74]. Those parameters are then passed as inputs to the (nonlinear) function outputting the reflectance. This strategy aims at mimicking the seasonal growth dynamics of vegetation using a semi-realistic model. We use it to simulate the reflectance of 11 spectral bands for a duration of 5 years. Although this synthetic time series is regularly sampled, we assume that its fixed time step is sufficiently low to give a complete view of the studied dynamical system. We show this 11-dimensional time series in figure 2.11. From the form of the phase portraits, one can check that the trajectory is indeed periodic, with a period of 1 year, as one can expect for vegetation.

In order to obtain an irregular sampling pattern for this trajectory, we uniformly sample $T = 50$ continuous timestamps among the first two and a half years. The values associated to these continuous timestamps are simply obtained through a weighted average of the two values that are the closest in time with the regular sampling of the data. The resulting irregular time series comes in the form of a set of couples $(t_k, \mathbf{x}_k)_{1 \leq k \leq T}$, representing the datapoints with their associated timestamps.

As explained earlier, we will learn from this dataset using a Koopman autoencoder model from which we train the continuous advancement matrix $\mathbf{L}$. In practice, directly training on long trajectories was observed to be more difficult in this setup than in the classical setup in which the optimization is performed on $\mathbf{K}$. Therefore, we chose to train in a way more similar to EDMD, by cutting our dataset into couples of consecutive values with associated time advancements, and learning to obtain the second value from the first one. Thus, in practice, we minimize the

5. `http://teledetection.ipgp.jussieu.fr/prosail/`

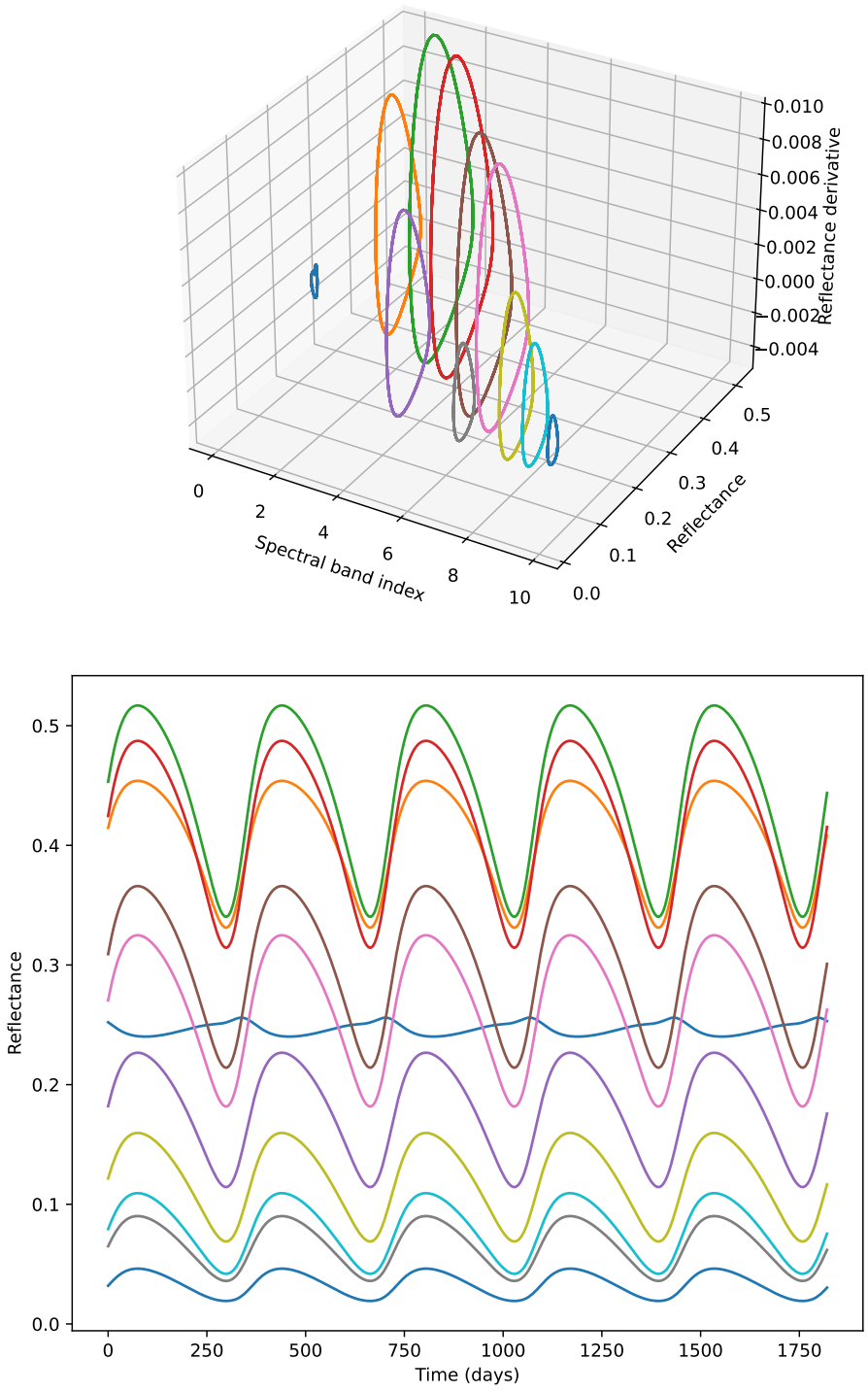

Figure 2.11 – Plot of the synthetic PROSAIL time series. Top: Phase portrait of each of the 11 spectral bands. Bottom: reflectance of each spectral band as a function of time.

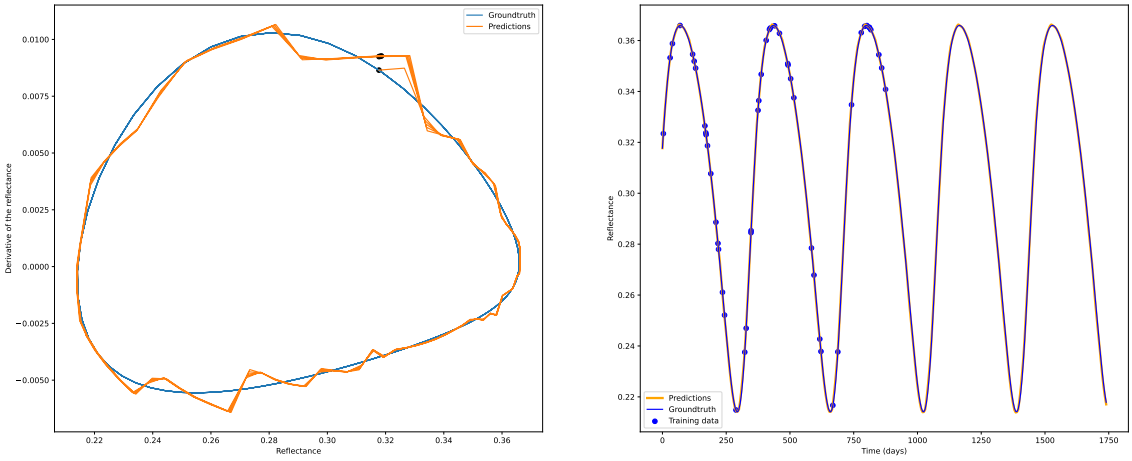

Figure 2.12 – Prediction performed from the initial state of the time series, by a Koopman autoencoder trained on a set of 50 datapoints. One can see from the phase portrait on the left that the time derivative of the prediction is slightly more erratic than the time derivative of the groundtruth. However, as can be seen on the right plot, these local deviations of the derivative do not visibly reflect on the predictions.

loss function from equation (2.58) with $N = 49$ time series, each of length 2. Since the studied dynamical system is close to linear, the main goal of this optimization is to assess the capacity of the model to capture the underlying periodic pattern when fed only with a low amount of information which is arbitrarily sampled. In figure 2.12, we show the prediction made by a trained model from the initial state of the trajectory, along with this model's training data. It appears that the model is able to make a near-perfect interpolation between its training points, but also to successfully extrapolate beyond these points. Indeed, the model has apparently been able to capture the periodic pattern of its training data.

2.3 Conclusion

In this chapter, we have introduced the Koopman operator, which is a common tool for building a linear model from an arbitrary dynamical system. We reviewed some of the most renowned methods that leverage the Koopman operator theory, i.e. the (extended) dynamic mode decomposition (EDMD) and a variety of methods based on nonlinear embeddings of the input state through neural encoders. In particular, we have described at length the Koopman autoencoder (KAE) framework along with some recent works that leverage it. Then, we have described our own contributions to the KAE framework. Those came in the form of a new orthogonality-promoting loss function term and of various techniques that leverage the continuous formulation of the Koopman autoencoder to get a continuous model from discrete training data or to train on datasets sampled with more complex patterns than the classical regular high-frequency sampling.

So far, our models are purely sequential, and they take only an initial state of the system as an input to produce their predictions. In the next chapter, we will study how, with no changes to the model or the training procedure, one can include a Koopman autoencoder model in a data assimilation scheme in order to either improve its forecasting ability or use it to solve completely different tasks such as interpolation or denoising.

Chapter 3

Using a learnt Koopman model as a dynamical prior for data assimilation

In the previous chapter, we detailed methods to train a Koopman autoencoder (KAE) model from a set of regularly or irregularly sampled data. One thing that we noticed throughout the presented experiments is that one often tried to forecast dynamical systems that exhibit some form of seasonality, which manifests in the form of a pseudo-periodic behaviour. From this observation, it seems clear that it is much easier to predict the future evolution of a state when one has knowledge of many of its past values than when one can only access one initial observation. Intuitively, when having access to information over several periods of the seasonality, one can simply infer the periodic pattern of the system to already obtain a good idea of how the system can be expected to behave in the future. Yet, the easiest way to forecast a dynamical system using a trained KAE is through equation (2.7), which only takes the initial condition $\mathbf{x}_t$ into account. Therefore, in this part, we introduce a framework to leverage a trained KAE to make more accurate predictions of dynamical systems by conditioning the model on more than one datapoint, and even using the spatial structure of the data when it exhibits some. This framework relies on variational data assimilation, for which we shall first give an overview in section 3.1. This overview notably details the difference between constrained and unconstrained variational data assimilation through a series of experiments. It also briefly describes how the differentiation of an assimilation cost has been made easier by recent automatic differentiation frameworks, and the way in which data assimilation has recently been used in conjunction with machine learning in the literature. Then, in section 3.2, we present our methods that leverage variational data assimilation with a pre-trained KAE model as a dynamical prior. The utility and versatility of these methods is demonstrated through denoising, forecasting and interpolation experiments on Sentinel-2 image time series.

3.1 An overview on variational data assimilation

Data assimilation is, by definition, a discipline in which one seeks to combine two sources of information: a set of observed data $\mathbf{X}_o$ and some prior information $\mathcal{P}$. In this regard, it can be seen as a subfield of inverse problems [76] in which one works on time series data with prior information on the dynamics. It has been used in a variety of fields, including numerical weather prediction [77], crop yield estimation [78], surface water quality modeling [79], forest inventory [80] and oceanography [81].

One of the most renowned techniques in data assimilation is the Kalman filter [82]. In a few words, the Kalman filter is a sequential model which relies on the assumption that both the observation operator and the evolution of the state are linear with a Gaussian noise. By leveraging a prior model and a set of observed states (which are linked to the state space through a linear observation model), it can provide an estimate of the mean and covariance of the state at any point in time. In cases where the dimension of the state is too high to keep track of its covariance matrix in practice, one can instead resort to the ensemble Kalman filter [83, 84]. This technique consists in representing the statistics of the state implicitly through a collection of state vectors rather than explicitly with an expression of its mean and covariance. The most noticeable advantage of the classical and ensemble Kalman filters is that they are good at providing uncertainty estimates. Although we will discuss stochastic aspects in chapter 4, we are only interested for now in producing a single prediction with no uncertainty estimates. For this reason, we will focus on another class of techniques in data assimilation, called variational data assimilation, which can be more straightforwarly used in conjunction with our KAE model developed in chapter 2.

3.1.1 Unconstrained variational data assimilation

Variational data assimilation is often used in dynamical systems study but also in a lot of other fields, notably for inverse problems. In particular, in variational data assimilation, one seeks to design a variational cost function which takes into account the data and the prior, and which one can minimize in order to obtain a solution to the problem at hand. We denote $\mathcal{X} \in \mathbb{R}^n$ the set in which the observed data $\mathbf{X}_o$ and the true state reside. Typically, this can be a set of images or time series. It should be noted however that some definitions of data assimilation are restricted to studying data that have a notion of time. Therefore, when dealing with a single image, we more generally talk about the "minimization of a variational cost", which is a notion encompassing variational data assimilation. With the assumption that the observation operator is an identity matrix, the cost function is generally of the form

$$\mathcal{C}(\mathbf{X}) = \mathcal{F}(\mathbf{X}, \mathbf{X}_o) + \mathcal{P}(\mathbf{X}) \tag{3.1}$$

where $\mathbf{X} \in \mathcal{X}$, and $\mathcal{F}(., \mathbf{X}_o)$ is a function of fidelity to the observed data, which can often be expressed as a distance, hence featuring $\mathcal{F}(\mathbf{X}_o, \mathbf{X}_o) = 0$. The prior term $\mathcal{P}(\mathbf{X})$ leverages some physical prior information on the data in order to provide some regularisation on the solution of the optimization problem. Once the variational cost is defined, one can minimize it using some variation of a gradient descent algorithm.

The gradient of $\mathcal{C}(\mathbf{X})$ is classically obtained analytically. For this purpose, one often seeks to compute the transpose (or adjoint) of the Jacobian matrix of the variational cost $\mathcal{C}(\mathbf{X})$ according to the optimized variable $\mathbf{X}$, which is the essence of the so-called *adjoint* methods [85, 86]. Recently, the computation of the gradient of the assimilation cost has been made more straightforward by the development of automatic differentiation tools, enabling to solve the optimization problem with recent dedicated frameworks like Pytorch [87], Tensorflow [88] or Julia [89]. Note that, although automatic differentiation is a necessary tool for deep learning, it is still nothing more than a tool for computing gradients. Hence, minimizing a variational cost through automatic differentiation is only an alternative to analytical gradient computation. In all of our experiments involving variational data assimilation techniques, we will leverage automatic differentiation with Pytorch to minimize the variational cost.

In order to illustrate the aforementioned minimization, our first example will be a famous inverse problem: image denoising. Let us suppose that we have at disposal a noisy image and that we want to retrieve the corresponding image without noise. This is an inverse problem in the sense that one seeks to find an initial image from which the addition of noise may have lead to the observed noisy image. In order to solve this problem, one can leverage an intuitive property of natural images, which is that they are relatively smooth in space. Indeed, we are seeking an image which lives (when having 3 channels each encoded with 8 bits and $W \times H$ pixels) in the space $[\![0, 255]\!]^{W \times H \times 3}$. It is obvious that, when uniformly sampling an image in this space, it is highly unlikely that we obtain a natural-looking image since the sampled image will not exhibit any spatial structure. In other words, in natural images, neighbouring pixels tend to have close values, which is not the case when randomly sampling an image in $[\![0, 255]\!]^{W \times H \times 3}$. A consequence of this fact is that the sum of spatial gradients in a natural image is always expected to be small. This sum of spatial gradients can be expressed, for example, using channelwise first order forward finite differences along the horizontal and vertical spatial dimensions. Concretely, for $\mathbf{X} \in [\![0, 255]\!]^{W \times H \times 3}$, it can be written as

$$\mathcal{P}_{\mathcal{T}}(\mathbf{X}) = ||\mathbf{X}_{1:W-1,.,.} - \mathbf{X}_{2:W,.,.}||_F^2 + ||\mathbf{X}_{.,1:H-1,.} - \mathbf{X}_{.,2:H,.}||_F^2, \tag{3.2}$$

where $||.||_F$ denotes the (tensor) Frobenius norm, i.e. the square root of the sum of all squared coefficients. Importantly, using this particular formulation, the color channels are processed independently from one another. From a probabilistic perspective, in the prior probability distribution on $[\![0, 255]\!]^{W \times H \times 3}$ associated to $\mathcal{P}_{\mathcal{T}}$, two univariate values are independent from one

another as soon as their color channels are different. This is a simplifying assumption, which may be challenged by resorting to a weighted norm with non-diagonal weights in equation (3.2), yet we stick to the simple (anisotropic) case here. As $\mathcal{P}_{\mathcal{T}}(\mathbf{X})$ is a differentiable function, its gradient can easily be computed, making it suitable for the inclusion in a variational cost. Thus, one can define the variational cost

$$\mathcal{C}_{\mathcal{T}}(\mathbf{X}) = ||\mathbf{X} - \mathbf{X}_o||_F^2 + \alpha \mathcal{P}_{\mathcal{T}}(\mathbf{X}), \tag{3.3}$$

where, with the notations of equation (3.1), $\mathcal{F}(\mathbf{X}, \mathbf{X}_o)$ is the squared Frobenius norm between the optimised image $\mathbf{X}$ and the observed image $\mathbf{X}_o$ while the prior $\mathcal{P}_{\mathcal{T}}$ is called a Tikhonov regularisation [90]. α is the weight attributed to this prior relatively to the data fidelity term. We show in figure 3.1 an example where we added uniformly distributed Gaussian noise to a reference image and then minimized the variational cost from equation (3.3) using the Adam optimizer [91] in Pytorch. The original image is obtained from the Kodak lossless true color image suite [1]. One can see that the simple prior knowledge that natural images should vary smoothly in space enabled to successfully retrieve an image which is closer to the original one in the sense of the peak signal-to-noise ratio (PSNR). The PSNR is defined as

$$\text{PSNR}(\mathbf{X}, \mathbf{Y}) = 20 \log_{10}(255/\sqrt{\text{MSE}(\mathbf{X}, \mathbf{Y})}) \tag{3.4}$$

where "MSE" denotes the mean squared error between the two images. Thus, higher values of the PSNR are better.

One can also qualitatively assess the influence of differing weights α to the Tikhonov prior term: smaller values of α might mean that some visual artifacts are conserved, but higher values imply an excessive blurriness in the obtained image. Note that, in order to avoid blurring the edges of the image, the Total Variation cost [92] is often preferred to the Tikhonov regularisation. The Total Variation is defined similarly to the Tikhonov term, yet it penalizes the largest spatial gradients less heavily, enabling to better preserve the edges that are present in natural images. The goal of the present experiment is not to present a strong denoising technique, but rather to illustrate how automatic differentiation can be be used to minimize a variational cost through a basic example that gives insight about the more complex methods detailed in section 3.2. Since such simple uses are rather uncommon in the literature, we release the code of this example publicly [2] in hope that it can be of pedagogical use.

As a second example, we come back to a dynamical systems context by using the Sentinel-2 time series dataset that we introduced in section 1.4. We now seek to obtain an image time series $\mathbf{X} \in \mathbb{R}^{T \times L \times W \times H}$, where T, L, W, H are respectively for the temporal, spectral, width and height dimensions. This setup is essentially the same as for images except that a time dimension was

1. `https://r0k.us/graphics/kodak`
2. `https://github.com/anthony-frion/automatic-differentiation-denoising`

Figure 3.1 – Results of denoising an image by minimizing the variational cost of equation 3.3 through automatic differentiation, with two values of α.

added. It is important to note that multi/hyperspectral images share the property of natural images that they generally exhibit strong spatial structure, in the sense that they are also spatially smooth. Therefore, one can also use equation (3.2) as a spatial prior for satellite image time series (SITS). Moreover, one can make a similar observation in the time dimension. Indeed, since Sentinel-2 images are taken every 5 days on the same area with similar conditions, it is unlikely in natural environments that the reflectance vector for a given pixel radically changes between 2 observations. This justifies the addition of a temporal smoothness term in a new Tikhonov prior designed for spatio-temporal data:

$$\mathcal{P}_{\mathcal{T}}(\mathbf{X}) = ||\mathbf{X}_{.,.,1:W-1,.} - \mathbf{X}_{.,.,2:W,.}||_F^2 + ||\mathbf{X}_{.,.,.,1:H-1} - \mathbf{X}_{.,.,.,2:H}||_F^2 + \alpha_t ||\mathbf{X}_{1:T-1,.,.,.} - \mathbf{X}_{2:T,.,.,.}||_F^2. \quad (3.5)$$

This new definition of the prior penalizes high gradients in the two spatial dimensions as well as in the temporal dimension of satellite image time series. The weight α_t enables to take into account the fact that the time series might not be equally smooth in all of its dimensions, and that its gradients along different dimensions should be penalized accordingly. It should be noted that this approach is similar to the optimal interpolation method introduced in [93], yet in this case we assume that the observed images are complete while the observation mask is spatio-temporal in the general case. This notably induces the choice to only link each value in $\mathbf{X}$ to its preceding

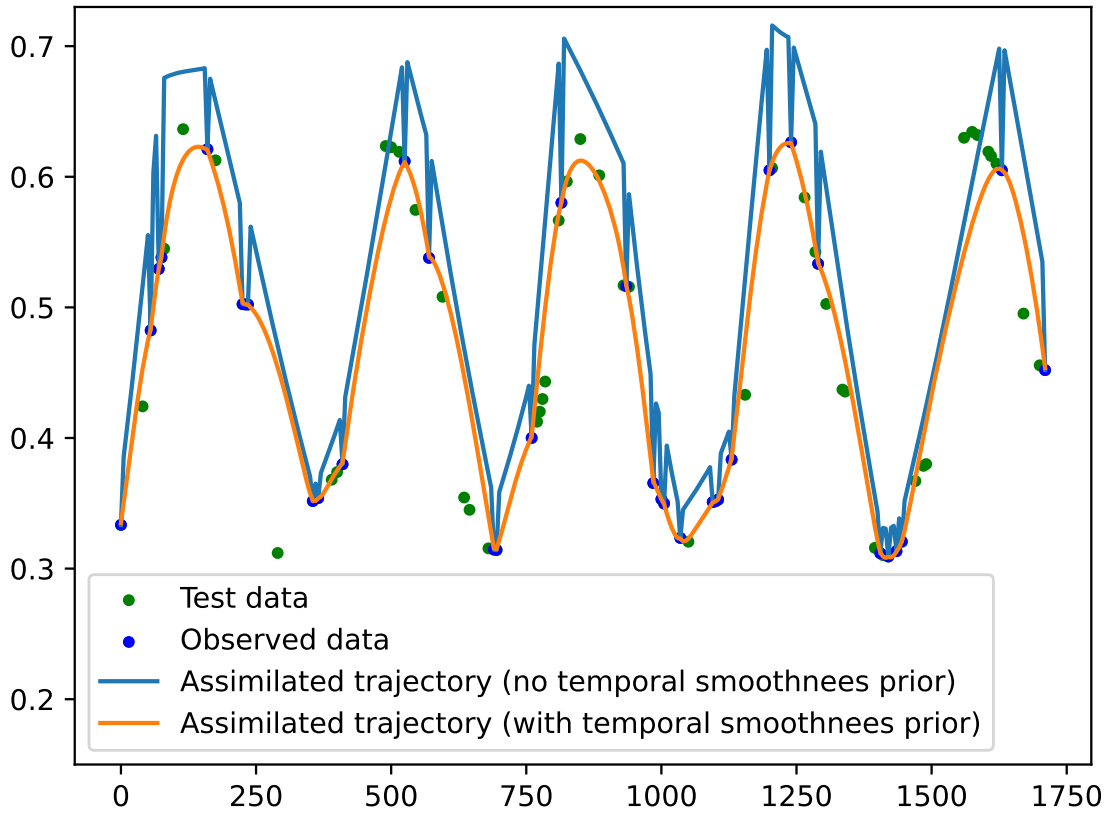

Figure 3.2 – Interpolation of a trajectory for a pixel of the Fontainebleau area, for the B7 band. We show the interpolation results when minimizing the cost of equation (3.5) without a temporal prior ($\alpha = 0$) or with a temporal prior ($\alpha > 0$).

and following values in the temporal prior, while optimal interpolation typically uses a Gaussian kernel. In addition, one could further generalize this expression by adding a spectral Tikhonov prior, i.e. a smoothness prior in the spectral dimension. This would be particularly reasonable for hyperspectral data, in which there may be hundreds of spectral bands and neighbouring bands are expected to have similar reflectances. Since we are mainly working on Sentinel-2 images with only 10 spectral bands, we do not include a spectral smoothness prior.

We show in figure 3.2 the result of solving a variational data assimilation cost composed of the prior $\mathcal{P}_{\mathcal{T}}(\mathbf{X})$ from equation (3.5) and from a squared Frobenius norm as the data fidelity term, for interpolating a time series of Sentinel-2 images from the forest of Fontainebleau. In particular, we test using a zero or nonzero weight α_t for the temporal smoothness prior in order to assess what is gained from it. Qualitatively, one can see that the time series obtained without a temporal regularisation takes aberrant values on the timestamps for which there is no observed data. This can be explained by the structure of the assimilation cost: when $\alpha_t = 0$ in equation (3.5), the images at different times are assimilated independently from each other. This

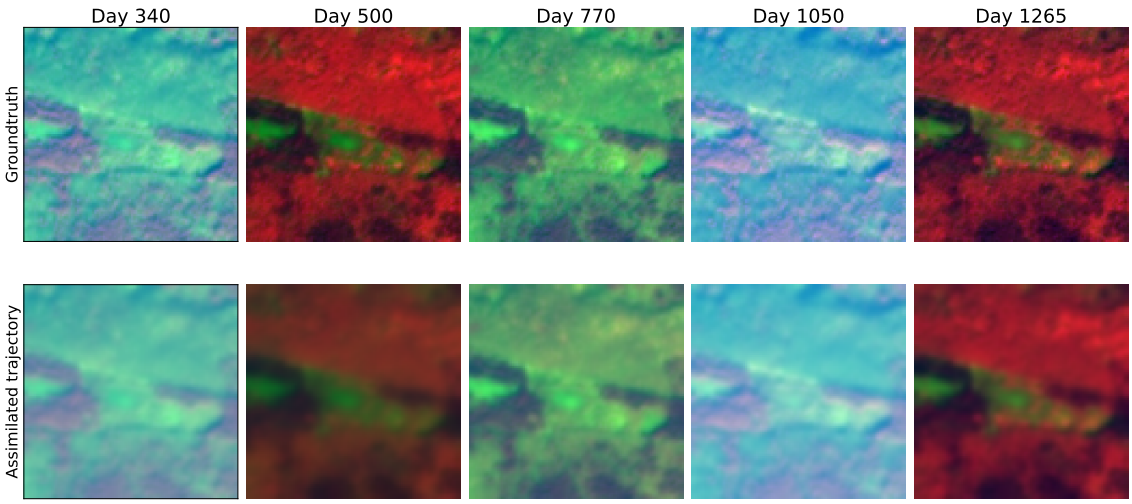

Figure 3.3 – RGB composition of an area of size 100×100 for which we perform interpolation through equation (3.5) on a set of sparsely sampled images. The colors are obtained through a 3-dimensional principal component analysis of the 10-dimensional pixel reflectance vectors, in order to obtain as much information as possible with 3 colors. The closest times on which an image is observed for the interpolation are respectively days 355, 525, 760, 1035 and 1290.

independence can be understood in a probabilistic sense when using the Bayesian perspective of variational data assimilation [94], which we discuss in section 3.1.4. Hence, when considering the variational cost for one particular time, if there is no observed data for this time, then the image that minimizes the cost is a monochromatic image, which is obviously not a desired result. In fact, the only reason why there is still some form of temporal structure in the predictions made with no temporal prior in figure 3.2 is that the optimized trajectory was initialised to a linear interpolation of the available data. In contrast, when including a temporal prior to the variational cost, one can see that the temporal evolution for a particular pixel is much smoother and more realistic. This highlights the need to take into account the temporal information when dealing with SITS. However, one can observe that the interpolation fails when the observed data is sampled too sparsely. In particular, the interpolated trajectory struggles to recover the local minima for which there is not enough observed data. We will see later that this problem can be addressed by explicitly leveraging the pseudo-periodicity of the dynamics through a constrained parameterization.

In figure 3.3, we show the reconstructions of images at some unobserved timestamps and compare those to the corresponding groundtruth. One can see that the interpolation preserves the global structure, but tends to produce blurry images, especially when there is no nearby observed image. One can see that there is indeed a clear link between the qualitative proximity of the interpolated image to the groundtruth and the temporal distance to an observed image (see the legend of the figure).

3.1.2 Constrained variational data assimilation

Alternatively to the classical case of equation 3.1 where the variable on which the optimization is performed is precisely the quantity that one seeks to reconstruct, one can implicitly define the solution of the problem through a usually lower-dimensional parameterization. In particular, when the solution has a temporal structure, the optimized variable can be the initial state $\mathbf{x}(0)$ of the defined trajectory. In this case, one can implicitly define the solution as the propagation of this initial condition through a dynamical model $\mathcal{M}$ [3]. This approach can be described as follows:

$$\mathcal{C}(\mathbf{x}(0)) = \mathcal{D}(\mathcal{M}(\mathbf{x}(0)), \mathbf{X}_o) + \mathcal{P}(\mathcal{M}(\mathbf{x}(0))). \tag{3.6}$$

The link to equation (3.1) can be made by letting $\mathbf{X} = \mathcal{M}(\mathbf{x}(0))$. However, the formulations are not equivalent in the general case since the model $\mathcal{M}$ is rarely able to span the whole space of solutions $\mathcal{X}$. Thus, putting aside the form of the gradient through $\mathcal{M}$, the main difference with equation (3.1) is that, when solving (3.6), the result is necessarily a trajectory that has been produced by the model $\mathcal{M}$. Thus, we refer to this approach as constrained variational data assimilation, as the obtained trajectory is constrained by the model $\mathcal{M}$. Importantly, since a well-chosen model $\mathcal{M}$ is very informative on the properties that the solution should exhibit, one can often obtain satisfactory solutions even when no prior $\mathcal{P}$ is defined in the constrained assimilation cost. Alternatively, $\mathcal{M}$ could be used as a dynamical prior in equation (3.1), which may result in the following variational cost:

$$\mathcal{C}(\mathbf{X}) = \mathcal{D}(\mathbf{X}, \mathbf{X}_o) + \mathcal{D}(\mathbf{X}, \mathcal{M}(\mathbf{X}_o(0))), \tag{3.7}$$

where $\mathbf{X}_o(0)$ is the initial observed state in $\mathbf{X}_o$. Note that, in this case, the dynamical prior is used for long-term prediction. Another common choice is to compute a prior term with only one-time-step predictions of the dynamical model $\mathcal{M}$. In order to distinguish this approach from the newly introduced constrained variational data assimilation, we would then refer to the cost of equation (3.7) as an unconstrained variational data assimilation cost. We emphasize that the key difference between constrained and unconstrained variational data assimilation is whether the optimized variable is the output trajectory itself (i.e. unconstrained assimilation) or a variable that parameterizes the output trajectory through a model (i.e. constrained assimilation).

In figure 3.4, we graphically represent the concept of constrained variational data assimilation. In order to further illustrate this approach, we will now use it on a toy example. Let us imagine that we are working on a 3-dimensional continuous dynamical system. We assume that the system

3. In order to keep the notations simple, we will denote as $\mathcal{M}(\mathbf{x}(0))$ the trajectory obtained by unrolling the model $\mathcal{M}$ over the whole time range of interest, although $\mathcal{M}(.)$ classically designates the advancement by one time step through the model.

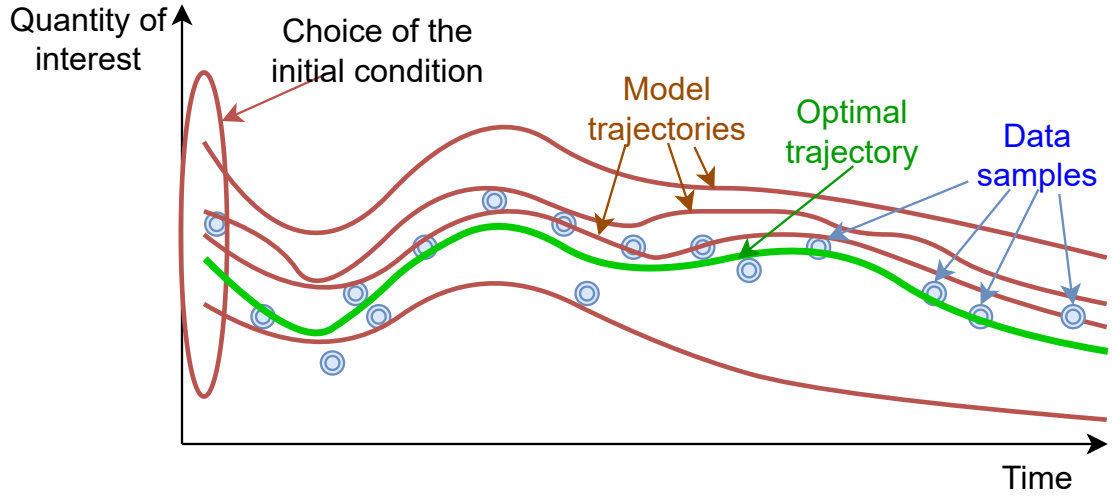

Figure 3.4 – Visual representation of constrained variational data assimilation. It consists in choosing the initial condition from which the model's trajectory minimizes the distance to the sampled data. One could also include a prior in the variational cost on the initial condition, such as the trajectory smoothness.

is driven by the following system of ordinary differential equations:

$$\begin{aligned} \frac{dx_1}{dt} &= \sin(x_1 + x_3) \\ \frac{dx_2}{dt} &= \sin(x_2 + x_3) \\ \frac{dx_3}{dt} &= \sin(x_1 + x_2 + x_3). \end{aligned} \tag{3.8}$$

We represent on figure 3.5 one trajectory of this dynamical system, which is obtained by solving the system of equations from a random initial state with the LSODA ordinary differential equation solver [20]. We will investigate the possibility of retrieving the state at time 0 when only the state of the system from time 10 to 50 is known.

We denote $\mathbf{x}(0) = (x_1(0), x_2(0), x_3(0))^\intercal \in \mathbb{R}^3$ the optimized variable. Similarly, the observed and modeled trajectories are respectively denoted as $\mathbf{x}_o$ and $\mathbf{x}$. The set of times on which the modelled trajectory will be compared to the observed trajectory is $H = (10, 10.01, 10.02, ..., 50)$: it contains 4000 timestamps. We then define the following cost function:

$$\mathcal{C}(\mathbf{x}(0)) = \sum_{t \in H} ||\mathbf{x}(t) - \mathbf{x}_o(t)||^2, \tag{3.9}$$

where, in practice, $\mathbf{x}$ is obtained by solving the initial value problem from $\mathbf{x}(0)$ with the forward Euler method with a time step of 0.01. The Euler method is much less efficient than the LSODA

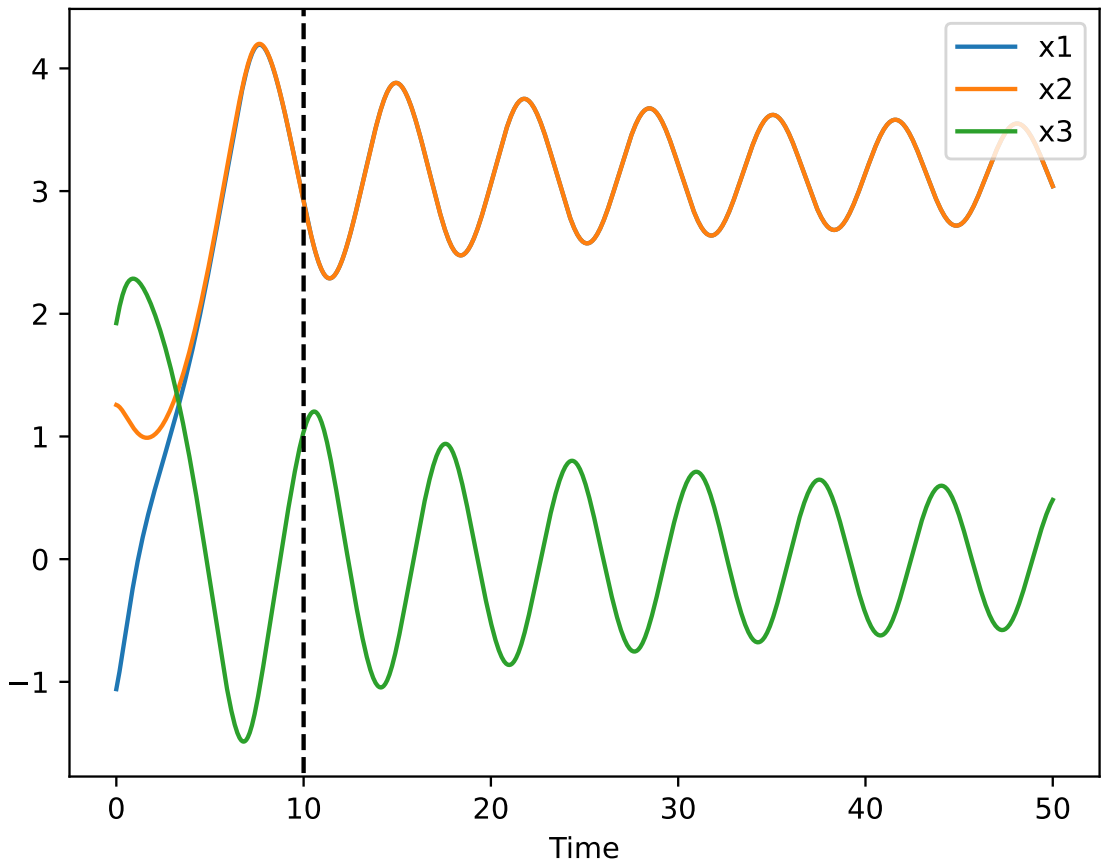

Figure 3.5 – A trajectory from the system of equations defined by (3.8), obtained by numerically integrating the equations from a random initial condition. The vertical line marks the beginning of the set H of observed timestamps in our associated constrained assimilation experiment.

solver, but it has the advantage of being easily differentiable by the automatic differentiation framework (i.e. Pytorch in our case). We have qualitatively observed that solving the initial value problem with the LSODA or the Euler scheme yielded the same results here. Therefore, when identifying the practical implementation of equation (3.9) to the general formulation of equation (3.6), we have that the model $\mathcal{M}(\mathbf{x}(0))$ corresponds to the result of the Euler resolution scheme and that $\mathbf{X}_o$ is defined on times H. There is no (unconstrained) prior $\mathcal{P}$ in equation (3.9). Indeed, the inclusion of such a prior appears to be unnecessary here since the prior dynamical model $\mathcal{M}$ is very accurate.

When minimizing the variational cost defined by equation (3.9), we manage to obtain a trajectory that matches the observed data but also the unobserved data from time $t = 0$ to 10. We do not show, on figure 3.5, the reconstructed trajectory since it is visually indistinguishable from the true trajectory. The initial condition that corresponds to figure 3.5 is $\mathbf{x}_o(0) \approx (-1.059, 1.257, 1.924)^\intercal$. The initial condition that we obtained through the constrained data assimilation cost of equation (3.9) was $\mathbf{x}(0) \approx (-1.058, 1.247, 1.917)^\intercal$. Therefore, this technique enabled us to infer the initial state with a relative precision of around 1%.

We now present a second constrained variational assimilation experiment, this time on the interpolation of Sentinel-2 image time series from the forest of Orléans (which is more complex than the Fontainebleau area since its pixels follow more diverse dynamical patterns). We will explicitly leverage the observed pseudo-periodicity of the data, by defining a dynamical model which is periodic with the known period of one year. We will only use the temporal structure of the data, which can still lead to good modelling performance, as we have shown in chapter 2 and will show again in section 3.2. We will therefore process the pixels independently from each other. We use a simple model which is defined by a repeating learnt pattern with a duration of one year. Thus, we use a constrained variational formulation that slightly deviates from the most common case in which the optimized variable is not the initial condition but a one-year-long pattern. Formally, using the notation $t\%P$ for the remainder of the Euclidean division of t by P, the assimilation cost is defined as

$$\mathcal{C}(\mathbf{x}_p) = \sum_{t \in H} ||\mathbf{x}_p(t\%P) - \mathbf{x}(t)||^2 + \alpha \mathcal{P}_\mathcal{T}(\mathbf{x}_p), \tag{3.10}$$

where P is the pseudo-period of the time series and $\mathbf{x}_p \in \mathbb{R}^{P \times L}$ is the periodic pattern of length P, with $\mathbf{x}_p(t) \in \mathbb{R}^L$ a reflectance vector for any $0 \leq t \leq P-1$. In this case, $\mathcal{P}_\mathcal{T}$ is a Tikhonov temporal prior with a cyclic structure, i.e. $\mathcal{P}_\mathcal{T}(\mathbf{x}_p) = \sum_{t=0}^{P-2}(||\mathbf{x}_p(t) - \mathbf{x}_p(t+1)||^2) + ||\mathbf{x}_p(P-1) - \mathbf{x}_p(0)||^2$.

To sum up, the variational cost on the optimized periodic pattern is driven by two terms: the fidelity of the corresponding periodic dynamics to the observed data and the temporal smoothness of the pattern. We show on figure 3.6 the result obtained by minimizing this cost on an example pixel from the Orléans area and then periodically repeating the obtained pattern. By comparing

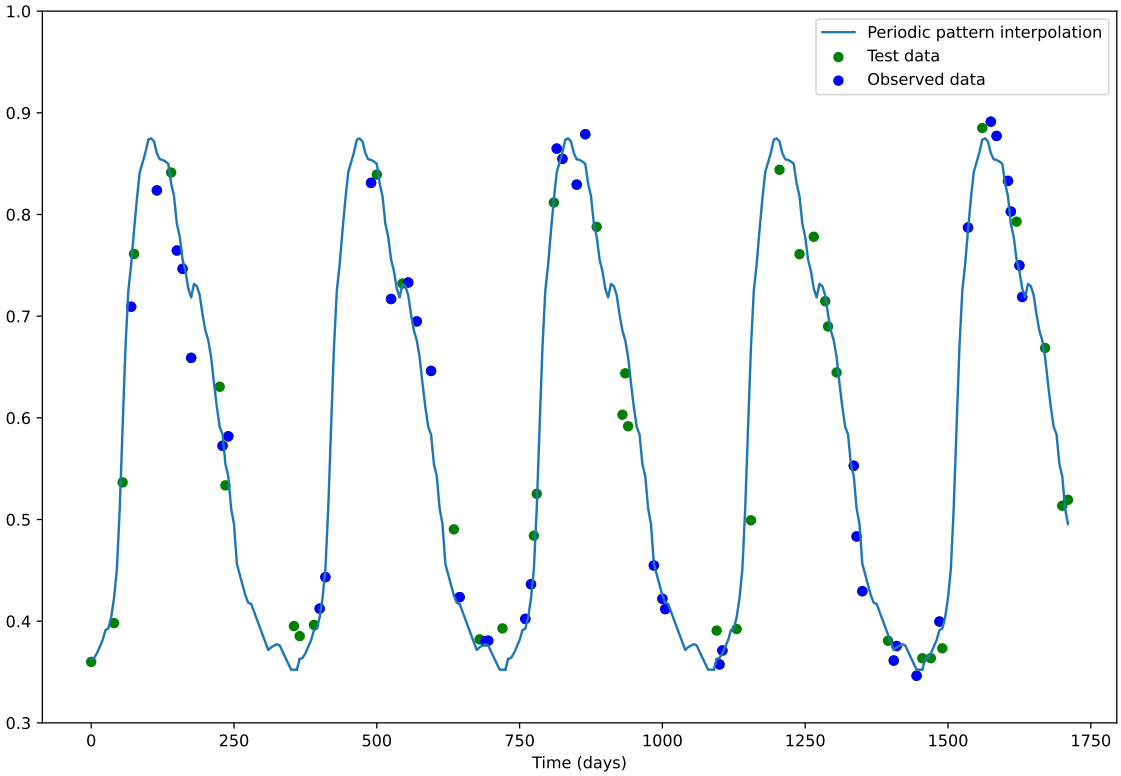

Figure 3.6 – Assimilation of a periodic trajectory using the variational cost of equation (3.10) on a set of observed datapoints for a pixel of the forest of Orléans.

this result to the one of figure 3.2 in which only a Tikhonov prior was used, one can see that the trajectory fits the test datapoints much closer. In particular, the constraint that the trajectory should be periodic is useful in the sense that it enables to leverage several years of data in order to obtain a general idea of what happens in average over one year. This means that the model does not suffer from a lack of observed data on a particular local extremum since it manages to exploit the data from corresponding parts of the other years in order to fill the gaps. In essence, this model is better because it has been informed about the physical properties of the SITS, in a similar spirit to the well-known physics-informed neural networks [95]. It should be noted, however, that this physical information comes in the form of a constraint on the modelled trajectory rather than of a loss term, which is more usual for physics-informed neural networks. This implies similar consequences as the one discussed in section 2.2.1 when comparing the Koopman models with exact or approximated orthogonality.

The assimilation of a periodic pattern will be used as a baseline against the interpolation with a pre-trained Koopman model as a dynamical prior in section 3.2.2.c.

3.1.3 Machine learning and data assimilation

As explained and illustrated in the last section, a variational data assimilation cost can be minimized through automatic differentiation, which is far easier than computing the gradient of the cost analytically, as long as the assimilation cost is expressed in a framework that supports automatic differentiation, e.g. Pytorch or Tensorflow. While there are many very simple priors which can easily be implemented in such a framework, e.g. the Tikhonov prior presented in last section, some commonly used priors are much more complex. For example, many operational methods in geophysics perform variational data assimilation with a complex physical model as a prior. Traditionally, those complex models are differentiated through the so-called *adjoint* methods [86], and they cannot be easily reimplemented in an automatic differentiation framework. While efforts have been made to reimplement some of these physical models (see [96] for a review), another line of work [97, 98] has explored the substitution of these physical priors by data-driven priors. Thus, this line consists in learning the prior function $\mathcal{P}$ in equations (3.1) or (3.6) through machine learning tools instead of using a physically sound function such as a Tikhonov regularisation or a complex physical model. Other works [99] have proposed to learn a data-driven surrogate model to predict the residual error of an existing physics-based model, which finally results in a hybrid model. Those models have the advantage of being fully differentiable and natively implemented in an automatic differentiation framework, which means that their associated cost can be differentiated automatically via the chain rule. Overall, linking data assimilation and machine learning is a very hot topic, which has been recently reviewed in [100].

3.1.4 The Bayesian interpretation of variational data assimilation

A notable disadvantage of using a temporal Tikhonov prior as presented in the introductory example with equation (3.5) is that it can only be used to interpolate between the observed data. Indeed, this temporal prior enables to characterize the relative values taken by the images, yet it does not enable to perform any guesses in temporal areas where no observations are available. Therefore, as long as a time on which one is trying to infer a value is located between two timestamps where data is available, the result should be reasonable. However, as soon as one is trying to predict a value outside of this range, all that the Tikhonov temporal prior can do is to predict a value that is equal to the last "truly" inferred value.

This issue can be best understood when considering the variational data assimilation problem through a Bayesian perspective. This perspective is rather intuitive when noting that the goal of variational data assimilation is to infer the most probable value of the (random) variable $\mathbf{X}$ given an observed variable $\mathbf{X}_o$, i.e. to model $p(\mathbf{X}|\mathbf{X}_o)$. More precisely, equation (3.1) can be seen as a Bayesian formulation of the likelihood of $\mathbf{X}$ given $\mathbf{X}_o$. Indeed, one can interpret $\mathcal{C}(\mathbf{X})$ as the negative log-likelihood of the probability distribution of $\mathbf{X}$ given an observed value $\mathbf{X}_o$. Using

Bayes' theorem, one has that

$$p(\mathbf{X}|\mathbf{X}_o) = \frac{p(\mathbf{X}_o|\mathbf{X})p(\mathbf{X})}{p(\mathbf{X}_o)} \tag{3.11}$$

From this, one obtains an expression of the negative log-likelihood:

$$-\log p(\mathbf{X}|\mathbf{X}_o) = -\log p(\mathbf{X}_o|\mathbf{X}) - \log p(\mathbf{X}) + \log p(\mathbf{X}_o). \tag{3.12}$$

Since $\mathbf{X}_o$ is observed, $\log p(\mathbf{X}_o)$ is constant and should therefore not be considered in the optimization problem from equation (3.1). The remaining terms can be linked in the following way:

$$-\log p(\mathbf{X}|\mathbf{X}_o) = \mathcal{C}(\mathbf{X}) + \text{cte} \tag{3.13}$$

$$-\log p(\mathbf{X}_o|\mathbf{X}) = \mathcal{F}(\mathbf{X}, \mathbf{X}_o) + \text{cte} \tag{3.14}$$

$$-\log p(\mathbf{X}) = \mathcal{P}(\mathbf{X}) + \text{cte}, \tag{3.15}$$

where "cte" denotes an unknown constant value. Hence, the data assimilation prior $\mathcal{P}$ is actually directly linked to a Bayesian prior, which explains the choice of the term "prior". This means that designing a prior in data assimilation actually boils down to designing a Bayesian prior distribution, although it is formulated implicitly and can be hard to write analytically. For example, the Tikhonov prior from equation (3.2) corresponds to a probability distribution on the space of images in which the likelihood of a value decreases with the sum of its squared spatial gradients. The choice of the data fidelity term $\mathcal{F}(\mathbf{X}, \mathbf{X}_0) = ||\mathbf{X} - \mathbf{X}_0||_F^2$ corresponds to a Gaussian distribution of mean $\mathbf{X}_0$ and unit covariance, making it a natural choice. When combining these two terms in the cost function (3.3), one obtains a posterior probability distribution which is a compromise between these two probability distributions. In some cases, this distribution might be relatively simple: for example, if the prior and the data fidelity term both induce Gaussian distributions, then the posterior distribution is also Gaussian. However, in the general case, the posterior distribution cannot be described analytically or easily sampled from.

Going back to our claim that the spatio-temporal Tikhonov prior is not suited for extrapolation, one can see that the value taken by $\mathbf{X}$ after the last observed time in $\mathbf{X}_o$ is not directly influenced by the data fidelity term, so that the compromise between the prior and the data fidelity term turns in this particular case into only the prior. Since the spatio-temporal smoothness prior always favors constant values, this will result in a constant prediction as soon as there are no more observations available. It should be noted that a temporal prior associated to a more complex joint distribution of the states, such as a Gaussian kernel (classical in optimal interpolation) should perform better in interpolation but still struggle for extrapolation. Therefore, performing forecasting of a SITS through variational data assimilation cannot be done with such a simple model. One would instead need to use a prior which can actually model the probability of the next

time step in $\mathbf{X}$ given the previous one in a less trivial way, taking into account the specificities of the dynamical system at hand (while the Tikhonov prior - or even a Gaussian kernel - is data agnostic). The constrained assimilation of a periodic pattern, corresponding to equation (3.10), is already an example of such a prior. It can indeed enable to perform extrapolation, although with a very simple model of the evolution. The object of the following section will be to use a more complex pre-trained data-driven dynamical model as a prior in order to obtain more complex predictions, in extrapolation as well as interpolation and denoising contexts.

3.2 Using a Koopman autoencoder as a dynamical prior

3.2.1 Our proposed assimilation frameworks

We assume here that a KAE has been trained on forecasting Sentinel-2 image time series at the pixel level, as described in chapter 2. Our main motivation for using it in a data assimilation framework rather than simply forecasting from an initial condition through equation (2.7) is that we want to be able to take multiple observations into account, as well as to include a spatial prior to the predictions. In order to do so, we introduce several assimilation frameworks, belonging to two categories: unconstrained assimilation, which is described in a general way by equation (3.1), and constrained data assimilation, similarly corresponding to equation (3.6). In essence, what we call constrained data assimilation still falls under the variational data assimilation principle, yet it it slightly more complex in the sense that the quantity that is optimized is not directly the one on which the data fidelity term is computed, but rather a latent parameterization of this term. More precisely, in our constrained assimilation objectives, we will be optimizing on the initial encoding of the pre-trained KAE, and we will seek the value of this latent variable which, when deterministically propagated through the model, gives the closest trajectory to the observed data. We emphasize that we will compute the gradients of the assimilation costs using automatic differentiation, which is a natural choice in our case since our prior dynamical models are differentiable by design and yet too complex for their gradients to be computed analytically without the chain rule.

When examining our global methodology from a deep learning perspective, one can observe that we first train a dynamical model to forecast from an initial state and then use this trained model to perform - through data assimilation - more complex tasks in which it has access to more data to perform a prediction. Thus, the training task of the model can be understood as a pretext task that is used to learn general information about the dataset. The various possible uses as a data assimilation prior can then be interpreted as downstream tasks, which are more complex and varied than the pretext training task. Therefore, this two-step procedure corresponds to the usual definition of self-supervised learning (SSL). SSL is a learning paradigm which differs from the currently most frequent paradigm in deep learning, i.e. supervised learning. It is precisely

described by a duality between the pretext task and the potential downstream tasks, and its most famous application is language modelling through large language models. Since this is not the main topic of the chapter, we propose a review of SSL in appendix B.

Let us now present the cost function for an unconstrained variational data assimilation scheme using a pre-trained KAE. As a reminder from chapter 2, a KAE consists in three components: an encoder ϕ, a decoder ψ and an advancement matrix $\mathbf{K}$. ϕ and ψ form together an autoencoder and they enable to go to and from a latent or encoded space, in which the evolution of the state is linear and defined by the matrix $\mathbf{K}$ through equation (2.7). Suppose now that we are looking to reconstruct a complete time series $\mathbf{X} \in \mathbb{R}^{n \times N \times (T+1)}$, with n the dimension of the state of a pixel, N the number of pixels and $(T+1)$ the number of timestamps, indexed from 0 to T. We will denote by $\mathbf{X}(t) \in \mathbb{R}^{n \times N}$ the state of $\mathbf{X}$ at timestamp t. We further assume that we have at disposal a possibly noisy and incomplete set of observed data $(\mathbf{X}_o(t))_{t \in H}$, where $H \subset [\![0, T]\!]$ is the set of observed timestamps. We assume for simplicity that $\mathbf{X}_o(t) \in \mathbb{R}^{n \times N}$ for any $t \in H$. A more general framework would have accounted for spatio-temporal masks H instead of only temporal masks, yet we do not consider this case in our experiments here. Our first proposed assimilation cost is an instantiation of (3.1) as:

$$\sum_{t \in H} ||\mathbf{X}_o(t) - \mathbf{X}(t)||^2 + \alpha \sum_{t=0}^{T-1} ||\mathbf{X}(t+1) - \mathcal{M}(\mathbf{X}(t))||^2 + \beta S(\mathbf{X}) \tag{3.16}$$

where $\mathcal{M}(\mathbf{x}_t) = \psi(\mathbf{K}\phi(\mathbf{x}_t))$ is a one-time-step prediction by the KAE and S gives the structure of the spatial prior. In practice, S can be a classical spatial regularisation leading to spatially smooth images, such as a Tikhonov regularisation [90] (as presented in equation (3.2)) or the total variation [92]. The first term of equation (3.16) corresponds to the fidelity to the observed data (i.e. first term of equation (3.1)) while the second and third terms of equation (3.16) form together the prior term (i.e. second term of equation (3.1)). Hereafter, we respectively refer to the second and third terms of equation (3.16) as the dynamical prior and the spatial prior.

This kind of cost is rather common and could actually be used in conjunction with any advancement model of the dynamical system. With a more general probabilistic formulation, the key point here is that the dynamical prior comes in the form of a sequential autoregressive model. If we put aside the spatial prior for clarity, this means that the prior probability $p(\mathbf{X})$ is simplified through the Markovian assumption, which leads to:

$$p(\mathbf{X}) = p(\mathbf{X}(0)) \prod_{t=0}^{T-1} p(\mathbf{X}(t+1)|\mathbf{X}(t)). \tag{3.17}$$

In practice, the prior probability $p(\mathbf{X}(0))$ of the initial state is often ignored (i.e. set to 1). Thus,

by making use of equation (3.15), we obtain

$$\alpha \sum_{t=0}^{T-1} ||\mathbf{X}(t+1) - \mathcal{M}(\mathbf{X}(t))||^2 = -\sum_{t=0}^{T-1} \log p(\mathbf{X}(t+1)|\mathbf{X}(t)). \tag{3.18}$$

Thus, we implicitly assume that $p(\mathbf{X}(t+1)|\mathbf{X}(t))$ follows a Gaussian distribution centered on $\psi(\mathbf{K}\phi(\mathbf{X}(t)))$. In contrast, the spatial prior characterizes the prior probability of each independent image, according to a same prior distribution. Thus, a prior $\mathcal{P}$ consisting in the sum of the dynamical and spatial priors corresponds to a more complex probability distribution through equation (3.15), but can still be relatively easily understood.

We now move on to the constrained assimilation cost, corresponding to the sketch of figure 3.4. With similar notations to the ones of equation (3.16), we seek to solve

$$\mathbf{z}_0^* = \underset{\mathbf{z}_0 \in \mathbb{R}^{N \times d}}{\arg\min} \sum_{t \in H} ||\mathbf{X}_o(t) - \mathbf{X}(\mathbf{z}_0)(t)||^2 + \beta S(\mathbf{X}(\mathbf{z}_0)), \tag{3.19}$$

where, for any time t, $\mathbf{X}(\mathbf{z}_0)(t) = \psi(\mathbf{K}^t\mathbf{z}_0)$. Since this optimization problem is not convex, it is helpful to initialize $\mathbf{z}_0$ to a good a priori value, which is given by the pre-trained encoder ϕ of the KAE as $\phi(\mathbf{X}_o(0))$. Hence, conceptually, we start from an initial value of $\mathbf{z}_0$ that only takes the initial observed state into account, and look to take into account the remaining of the observed data through gradient descent.

This time, the optimized variable is not $\mathbf{X}$ but rather $\mathbf{z}_0$, which parameterizes a set of possible trajectories $\mathbf{X}$ but which is not able to model all the theoretically possible trajectories. Indeed, there is no guarantee that, for a given $\mathbf{X} \in \mathbb{R}^{n \times N \times T}$, there exists a latent initial condition $\mathbf{z}_0$ whose propagation through the model with equation (2.7) gives $\mathbf{X}$. However, we assume that not every value $\mathbf{X} \in \mathbb{R}^{n \times N \times T}$ is a realistic trajectory for the dynamical system. In other words, the set of actually realistic trajectories is contained in a lower-dimensional submanifold of $\mathbb{R}^{n \times N \times T}$, which is too complex to be described analytically [4], but from which we hope that the pre-trained Koopman model can give a good approximation. Thus, restraining the optimization to the set of trajectories that can be produced by a pre-trained Koopman dynamical model is useful since it is assumed that any trajectory produced by the model is a reasonably realistic one. For this reason, solving for an assimilation cost formulated in this way is effective in more difficult setups, notably when the observed data is scarce and very noisy, since the parameterization through the Koopman dynamical model is moderately sensitive to this noise. In contrast, when the data is abundant and exhibits little to no noise, constrained assimilation frameworks will not perform well since the error of reconstructing the observed data through the model might actually be of higher magnitude than the noise inherent to the observations. We will illustrate this distinction

4. This corresponds to the manifold hypothesis, which is central in machine learning. See, e.g., [101] for a detailed development.

between the cases where constrained or unconstrained assimilation perform best in subsequent experiments.

Importantly, when obtaining a latent initial state $\mathbf{z}_0^*$ as a result of solving equation (3.19), one can obtain from it a trajectory that is not limited in time. Indeed, using the decoder ψ and the Koopman matrix $\mathbf{K}$ of the pre-trained KAE, a prediction $\hat{\mathbf{x}}_t = \psi(\mathbf{K}^t\mathbf{z}_0)$ can be obtained at a very low cost for any desired time t. Thus, solving equation (3.19) enables to perform denoising, interpolation, and forecasting of the observed data at the same time and without any change to the method.

One possible issue that could arise when minimizing a constrained assimilation cost is that the data that we assimilate might be very different from the data on which the model has been trained. In the context of SITS, this can happen when the model has been trained on a given area and is then tested on a different one which features a distribution shift in the data. In order to alleviate this issue, we propose to include the parameters of the model inside of the assimilation framework. In this way, the model is fitted to the assimilation data on the fly in order to be able to fit it better while its initial parameterization might not have been able to fit the data at hand. Concretely, this is done by solving an optimization problem similar to the constrained assimilation from equation (3.19) but where the parameters of the model are optimized jointly with the initial latent state (while they were previously fixed). This results in

$$\min_{\mathbf{z}_0,\mathbf{K},\psi} \sum_{t\in H} ||\mathbf{X}_o(t) - \mathbf{X}(\mathbf{z}_0)(t)||^2 + \beta S(\mathbf{X}(\mathbf{z}_0)). \tag{3.20}$$

It should be noted however that such a technique is very sensitive to overfitting since the model will be fine-tuned to a very small amount of data. For this reason, there are several good practices that one should keep in mind.

— Data assimilation is typically less sensible to noise when used for interpolation than for extrapolation. Indeed, the missing data inside of the time range of available data is typically close to the distribution of the available data, and the very structure of the model (with an underlying linear evolution) makes it quite robust in this context.
— In any case, the fine-tuned parameterization of the model should not get too far from its pre-trained configuration. This advocates for using a typically much smaller learning rate during the fine-tuning step from equation (3.20) than during the training step.

Besides, one might be tempted to skip the pre-training step and to try solving equation (3.20) with a randomly initialised KAE model, but this task is much more difficult to solve than the forecasting pretext task of chapter 2, which means that it cannot be solved in practice without a good initialisation of its parameters.

3.2.2 Experiments on Sentinel-2 image time series

We now illustrate our discussions of section 3.2.1 through a series of experiments in which we leverage the assimilation frameworks introduced in equations (3.16), (3.19) and (3.20). The experiments will be separated according to the applications considered: the denoising, forecasting and interpolation experiments will respectively be presented in sections 3.2.2.a, 3.2.2.b and 3.2.2.c.

3.2.2.a Denoising experiments

In this section, we focus on a case where the available data is regularly-sampled at a sufficiently high frequency, but where it contains a certain amount of noise that one is looking to filter. For this purpose, we use the true Sentinel-2 SITS introduced in section 1.4, to which we artificially add a Gaussian noise with a given standard deviation, independently for each pixel, spectral band and time. We are aware that the denoising of Sentinel-2 data has limited practical interest (contrarily to forercasting and interpolation) since this satellite constellation has a more favorable signal-to-noise ratio than other satellites. We still perform these experiments for illustrative purposes, in order to derive general principles for the problem of denoising time series using a KAE as a dynamical prior in an assimilation framework.

Concretely, we denote by $\mathbf{X}_{ref}$ the data that we consider as the groundtruth, i.e. the data directly corresponding to the SITS datasets from section 1.4. $\mathbf{X}_{ref}$ is a tensor of order 4, with a temporal dimension, a spectral dimension and 2 spatial dimensions. Then, the observed data $\mathbf{X}_o$ is obtained by independently adding a Gaussian white noise of standard deviation σ to each scalar entry of $\mathbf{X}_{ref}$, i.e.

$$\mathbf{X}_o = \mathbf{X}_{ref} + \mathcal{E}, \tag{3.21}$$

where $\mathcal{E}$ is a random tensor of the same size as $\mathbf{X}_{ref}$ where the entries are independent from each other and all follow the Gaussian distribution $\mathcal{N}(0, \sigma^2)$.

In particular, like in chapter 2, we will be using the interpolated (regular) version of the Fontainebleau and the original (incomplete) version of the Orléans time series. The pre-trained Koopman model is trained on the regular Fontainebleau time series, as described in the experiment of section 2.2.1.

For now, we restrain to methods that process pixels independently from each other, without using the spatial structure of the images. In particular, we will not make use of any spatial prior $\mathcal{S}$ in the assimilation costs. Concretely, this means that equation (3.16) is simplified as

$$\sum_{t \in H} ||\mathbf{X}_o(t) - \mathbf{X}(t)||^2 + \alpha \sum_{t=0}^{T-1} ||\mathbf{X}(t+1) - \mathcal{M}(\mathbf{X}(t))||^2, \tag{3.22}$$

while equation (3.19) is simplified as

$$\mathbf{z}_0^* = \underset{\mathbf{z}_0 \in \mathbb{R}^{N \times d}}{\arg\min} \sum_{t \in H} ||\mathbf{X}_o(t) - \mathbf{X}(\mathbf{z}_0)(t)||^2. \tag{3.23}$$

Besides, we will not make use of the fine-tuning framework in equation (3.20) for the denoising experiments. Indeed, fine-tuning the model in the context of noisy data does not seem to be a reasonable choice, precisely because the objective of the assimilation is to find the realistic trajectory that is closest to the noisy one rather than to fit the noisy trajectory. Thus, the noisy trajectory at hand should not interfere with the prior definition of what should be seen as a "realistic trajectory".

Our point of comparison for the performance of our denoising methods will be a classical interpolation method called the Cressman interpolation [30]. The idea of this method is to compute, either from a complete or an incomplete observed time series, a smooth output in which each value is obtained as a weighted average of the neighbouring observed values. For a given interpolated timestamp, the weight attributed to the different observed values decreases with the distance of their timestamps through a Gaussian kernel function. This Gaussian kernel is defined as

$$w_{t,t'} = \exp \frac{-(t-t')^2}{2R^2}, \tag{3.24}$$

where R is the radius of the Gaussian kernel, which determines how fast the weight associated to an observation should decrease as the temporal distance increases. Then, considering that the observed data $\mathbf{X}_o$ are located at timestamps H, the Cressman interpolation at timestamp t is given by

$$\mathbf{X}_C(t) = \frac{1}{\sum_{t' \in H} w_{t,t'}} \sum_{t' \in H} w_{t,t'} \mathbf{X}_o(t'). \tag{3.25}$$

Although this is originally an interpolation framework, it can also be used for denoising purpose, which is the way that we will use it here. The results of the Cressman method are quantitatively compared to the results of our methods by computing the mean squared error (MSE) between the method's output and the reference trajectories of $\mathbf{X}_{ref}$.

We distinguish 2 cases in our denoising experiments: the case with a high noise magnitude and the case with a low noise magnitude.

For the low noise magnitude case, we work with the interpolated version of the Fontainebleau time series (i.e. the training data of the Koopman prior) as our reference data. We will test the following values for the noise standard deviation σ in equation (3.21): 0.005, 0.01, 0.02, 0.04, 0.05. As a point of comparison, the standard deviation of the Fontainebleau time series (all times, pixels and spectral bands confounded) is around 0.2. Thus, although the signal-to-noise ratio (SNR) depends on the pixel and the spectral band that is considered, the noise standard deviations that we listed above roughly correspond to values of the SNR ranging from 6 dB to

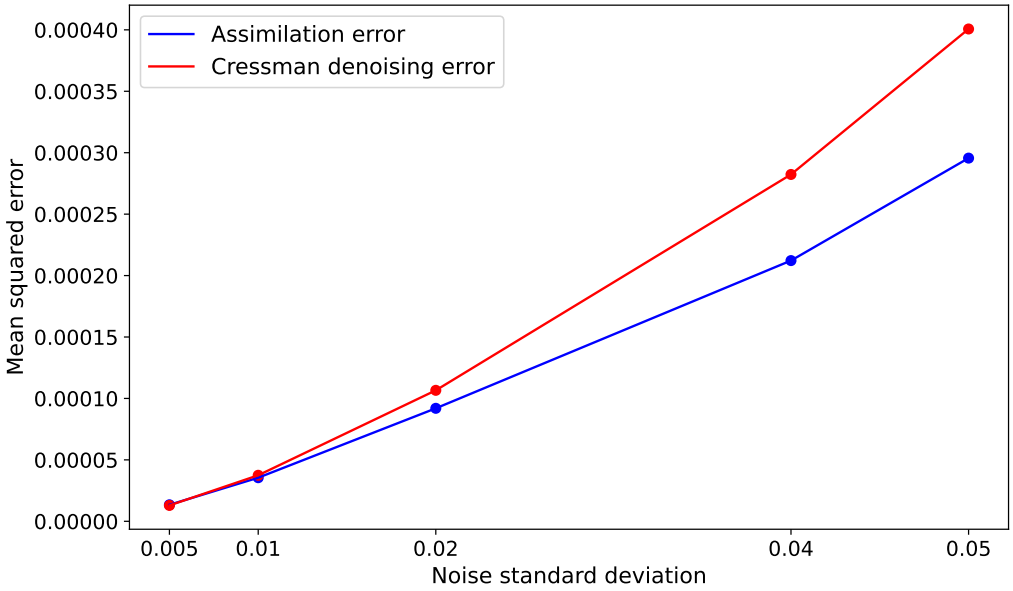

Figure 3.7 – Comparison of the denoising performance for different levels of noise, with the Cressman method from equation (3.25) and the assimilation with a Koopman prior with equation (3.22). The hyperparameters of the methods are fine-tuned for each level of noise.

16 dB. For the lowest of these noise magnitudes, the unconstrained assimilation framework from equation (3.16) is more appropriate than the constrained assimilation from equation (3.19). This is simply due to the fact that the constrained formulation yields a non-negligible reconstruction error even when used on the original data without noise. For this reason, when the magnitude of the noise is comparable to the magnitude of the reconstruction error by the model, the constrained assimilation cannot successfully improve the proximity to the reference data compared to the noisy observed data. In contrast, the unconstrained formulation should always produce a result that is at least as close to the reference data as the observed data.

The Cressman method from equation (3.25) and the unconstrained assimilation with a Koopman dynamical prior from equation (3.22) both have one hyperparameter, which is respectively the Gaussian radius R and the weight α of the dynamical prior. For R, we search for the optimal value in the range $(0.1, 0.2, ..., 3)$. For α, we search in the range $(1, 2, ..., 10)$. For each noise magnitude, we randomly sample 100 pixels in the image. The hyperparameter search for both methods is performed independently for each noise level, and the optimal hyperparameter is then shared for all of the 100 sampled pixels.

We show in figure 3.7 the mean squared errors obtained by the two methods as a function of the magnitude of the noise in the observed data. From this figure, one can see that, while the

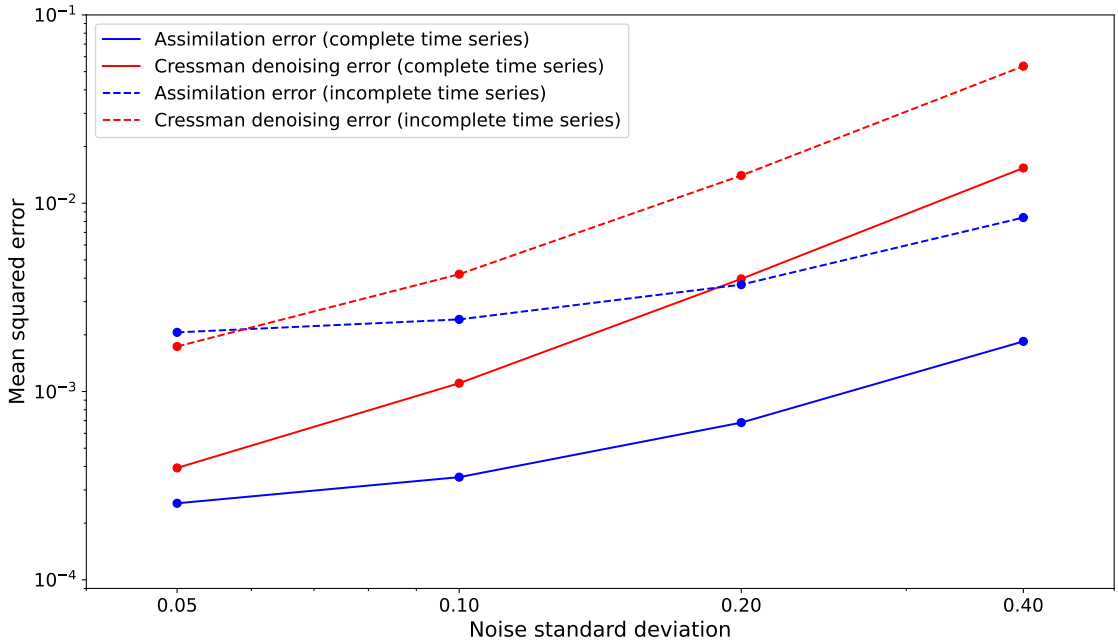

Figure 3.8 – Comparison of the denoising performance for different "high amplitude" levels of noise, with the Cressman method from equation (3.25) and the constrained assimilation with a Koopman prior with equation (3.23). The complete time series corresponds to the training data from the forest of Fontainebleau and the incomplete time series corresponds to test data from the forest of Orléans.

performance is equivalent for the lowest noise magnitudes, our method performs a significantly better denoising when the magnitude of the noise becomes larger.

We now move on to a set of higher noise levels. Hence, we now use the same experimental conditions but with the standard deviation of the noise taking the following values: 0.05, 0.10, 0.20, 0.40. Using the same approximation as earlier, one can say that this roughly corresponds to an SNR ranging from -3 dB to 6 dB. For these increased noise levels, we now resort to the constrained assimilation cost of equation (3.23). This cost is more appropriate than the unconstrained variational cost from equation (3.22) since it is far more robust to noise. Indeed, since the result of this method is a prediction performed by the Koopman model, it is (in theory) necessarily a natural-looking trajectory. In addition, this method has no hyperparameter to tune. Concerning the tuning of the hyperparameter R for the Cressman denoising in equation (3.25), we observed that it had little influence on the performance in this range of noise levels, so we simply set it to $R = 3$.

The mean squared errors of both methods as a function of the noise magnitude are shown in figure 3.8. Note that we chose a logarithmic scale for this figure. As a reminder, since the standard deviation of the reference Fontainebleau data is approximately 0.2, the SNR is smaller

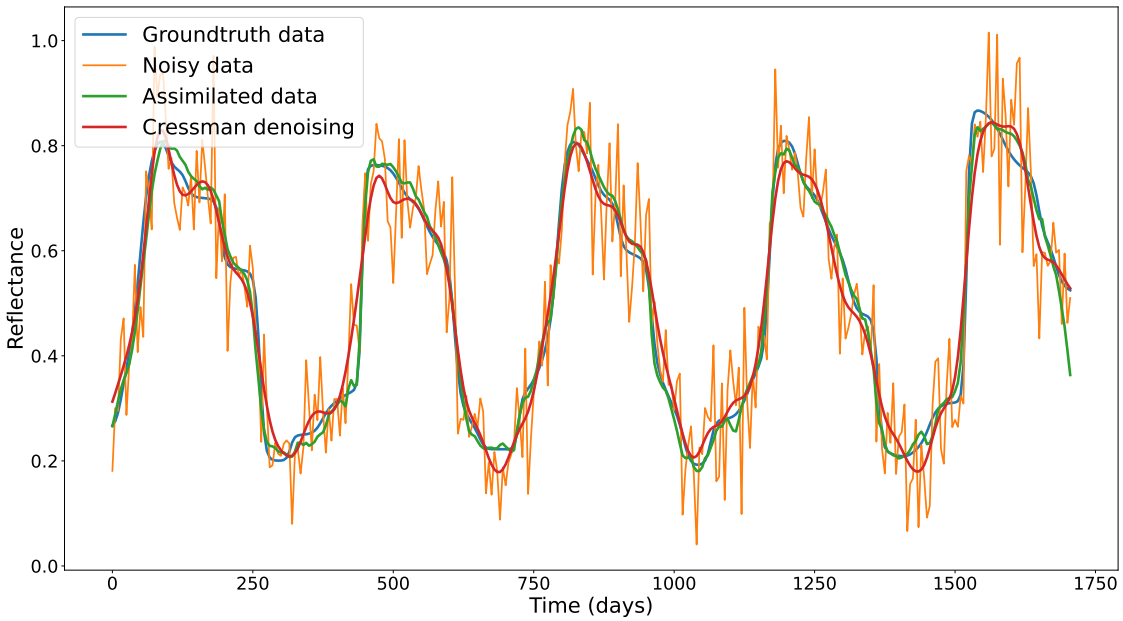

Figure 3.9 – Comparison of the denoising performed by the Cressman method and the constrained assimilation method for a particular pixel. The represented spectral band is the B7 band (which belongs to the near infrared domain), and the level of noise is $\sigma = 0.1$.

than 1 for the highest noise values that are considered. It appears from this figure that our constrained assimilation with a pre-trained KAE as a dynamical prior performs far better than the Cressman denoising. The most significant differences are obtained for the high noise levels on the regular Fontainebleau time series, which can be explained by the fact that it corresponds to the training data of the Koopman model. On the irregular Orléans time series, both methods perform significantly worse since the time series are incomplete, making it more difficult to leverage the temporal smoothness prior. Although our method is still significantly stronger than the Cressman method, the difference is smaller since this data was not seen by the KAE during its training.

In figure 3.9, one can qualitatively assess the denoising performance of the two considered methods on the most energetic spectral band for a given pixel. In addition, we show in figure 3.10 the pixel-by-pixel denoised image sampled in a time series with noise standard deviation $\sigma = 0.05$. It should be noted that both methods use the temporal (which is not represented in the figure) rather than the spatial information to perform their denoising. In addition, the 3 bands in the visible domain have low amplitudes relatively to the infrared bands of the Sentinel-2 images. This explains why they are so sensitive to noise and why our method performs so much better than the Cressman method on these particular bands. Indeed, the Cressman interpolation is performed individually on each spectral band since this method uses no physical information on the joint distribution of values for the 10 available spectral bands of the reflectance vectors.

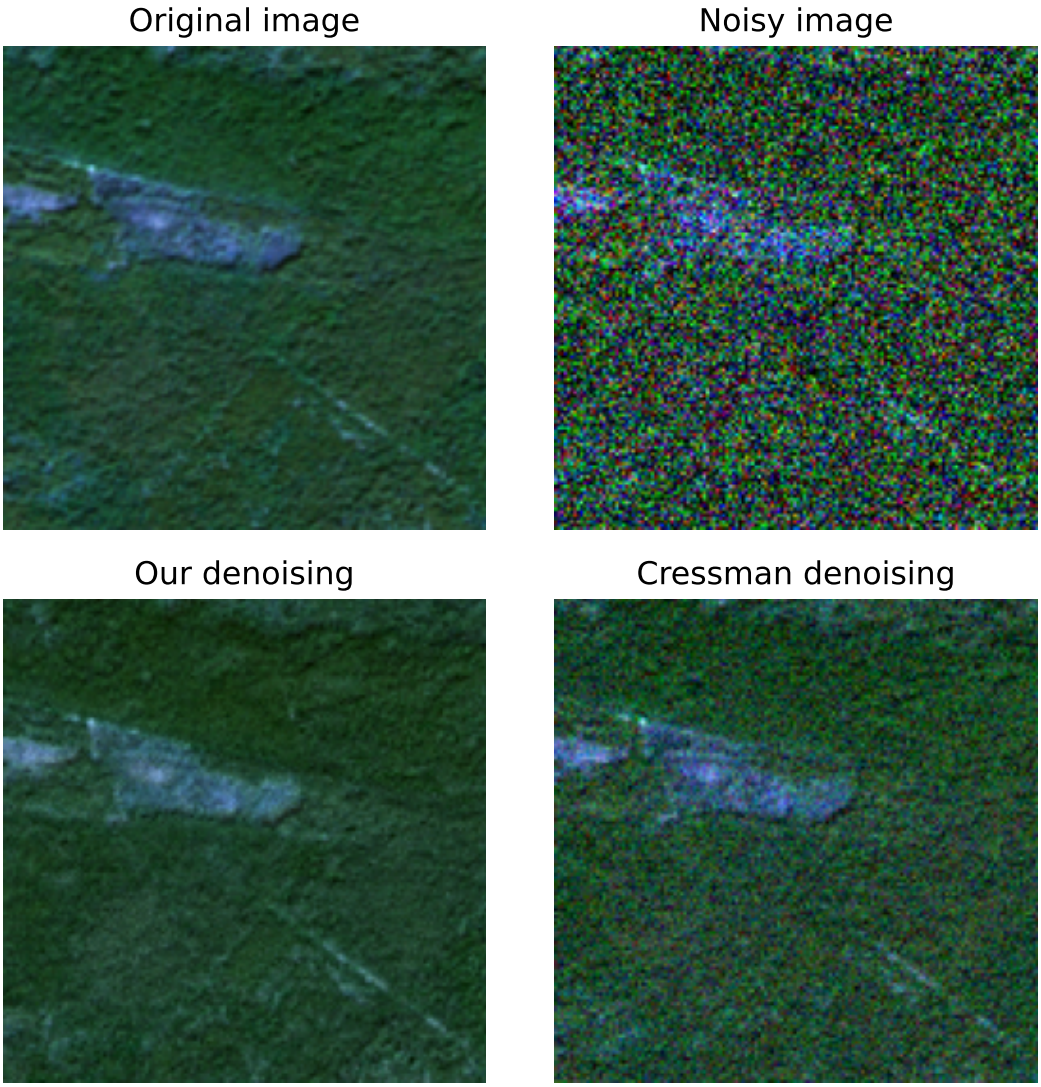

Figure 3.10 – Representation of the RGB bands (B4,B3,B2) for a reference image, its noisy version and the denoisings proposed by the two methods.

In contrast, the Koopman prior leverages the information of the relatively less affected infrared bands in order to better reconstruct the bands of the visible domain.

3.2.2.b Forecasting experiments

From now on, we focus on the constrained assimilation framework of equation (3.19), which in our opinion is the most powerful framework for leveraging a pre-trained KAE model in a semi-supervised fashion. As previously explained, when obtaining an initial latent state $\mathbf{z}_0^*$ through equation (3.19), we have access to a modelled trajectory that is not limited in time, hence enabling to easily perform forecasting. This methodology, which we call assimilation-forecasting, is expected to yield much better results than a naive prediction from an initial reflectance vector through equation (2.7) since it enables to take into account multiple data snapshots rather than a single one, hence better leveraging the temporal structure of the observed data. In addition, including a spatial prior $\mathcal{S}$ in the assimilation cost can further improve the predictions by enabling to also take the spatial structure of the data into account.

In order to concretely illustrate the boost in performance when predicting through assimilation of an observed trajectory rather than from a single initial condition, we go back to our experiment on Sentinel-2 time series forecasting in section 2.2.1. As a reminder, we had benchmarked 4 models for the long-term forecasting of the reflectance at the pixel level. The first two models were the KAE architecture with and without the orthogonality loss introduced in equation (2.34). The third model was a classical long-short-term memory neural network (LSTM), trained in the same conditions. The fourth model, which we called long term DMD, was a linear autoregressive model also trained for long-term prediction, therefore analogous to a KAE but where identity functions are used instead of ϕ and ψ. Each of these models (except long term DMD) was trained from 5 different random initialisations to get an idea of their respective mean performance. When testing the models for extrapolation to unknown dates, we saw that, although the KAE trained with the orthogonality loss performed the best (c.f. table 2.1), the models still struggled to make coherent long-term predictions, as could be qualitatively seen in figure 2.5.

We now use the same trained instances of the models, but we include them in our constrained assimilation framework instead of using them in the same way as they were trained. Concretely, we consider that the Sentinel-2 time series are observed from time 0 to $T_{train} = 242$. Thus, in the case of the interpolated Fontainebleau data, the set of observed times is $H = [\![0, 242]\!]$. This also corresponds to the data on which the model was pre-trained. For the pre-trained KAE instances, in the framework of equation (3.19), we seek to solve

$$\mathbf{z}_0^* = \underset{\mathbf{z}_0 \in \mathbb{R}^{N \times d}}{\arg\min} \sum_{t=0}^{T_{train}} ||\mathbf{X}_o(t) - \mathbf{X}(\mathbf{z}_0)(t)||^2 + \mathcal{S}(\mathbf{X}(\mathbf{z}_0)). \tag{3.26}$$

Note that, although the pre-trained encoder ϕ does not appear in this assimilation cost, it is used

for initializing its solution to $\mathbf{z}_0 = \phi(\mathbf{X}_o(0))$. Thus, assuming that the final solution $\mathbf{z}_0^*$ is the same as this initial condition, the output of this result would be the same as when simply using the advancement function from equation (2.7). In practice, the final solution always varies from the initialisation as there exist initial latent states that match the remaining of the observed trajectory far better than $\phi(\mathbf{X}_o(0))$ does. However, this initialisation is still a reasonable guess, and we observed that, in average, it enables to get better final results than an initialisation to a random latent state or a vector of zeros. Indeed, as is often the case in deep learning optimization, there is no guarantee that the variational cost that we define has a convex landscape, which means that the choice of the initialization of $\mathbf{z}_0$ might have a strong influence on the final result.

We also adapt this framework to the LSTM and to long term DMD, yet in these adaptations we have to optimise on the input initial condition (which we denote as $\mathbf{x}_0$) rather than on the encoded initial condition $\mathbf{z}_0$. Thus, equation (3.26) becomes:

$$\mathbf{x}_0^* = \underset{\mathbf{x}_0 \in \mathbb{R}^{N \times n}}{\arg\min} \sum_{t=0}^{T_{train}} ||\mathbf{X}_o(t) - \mathbf{X}(\mathbf{x}_0)(t)||^2 + \mathcal{S}(\mathbf{X}(\mathbf{x}_0)), \tag{3.27}$$

where $\mathbf{X}(\mathbf{x}_0)(t)$ is the prediction from either the LSTM or long term DMD at time t from $\mathbf{x}_0$. The initial value that we use is, of course, the observed initial reflectance vector, but it might not be the value that best fits the subsequent observed states in practice. As in the case of KAEs, using a reasonable initialization of the optimized variable may significantly improve the final result because of the presumed non-convexity of the variational cost. Again, since these models are sequential, solving this problem enables to produce an arbitrary long predicted time series.

Let us first discuss the computational requirements of solving these optimization problems. It should be noted that, in order to include a spatial prior $\mathcal{S}$, the predictions of the pre-trained model have to be computed along with their gradients on all N pixels in parallel. Therefore, the memory consumption of this computation is critical, and it scales linearly with the number of pixels in the image. One could imagine splitting the large images into smaller images in order to alleviate the memory requirements, yet considering too small images might lead to stronger undesired boundary effects, which is why we always seek to keep as large images as the memory allows.

In addition, the memory requirements are more important when the model for the evolution of the state is computationally heavy. Therefore, for the KAEs, since the latent evolution is linear, this source of memory consumption is marginal compared to the final decoding of the latent states. In fact, since one only has to decode the states for the timestamps in H on which the observed data are defined, the assimilation for autoencoders is far less costly when the observed data is sparse. For long term DMD, since no encoding and decoding is performed, the memory consumption is always much smaller than for the other models. In contrast, the LSTM, since it

features a nonlinear latent evolution of its hidden vectors, is by far the least memory-efficient of the benchmarked models.

When no spatial prior is included in the assimilation cost, the predictions for the different pixels are independent from each other, which means that they do not have to be computed in parallel. Since the LSTM model consumes much more memory than the Koopman model, we choose simply not to include a spatial prior in the assimilation cost for now in order to ensure a fair comparison.

Table 3.1 – Assimilation-forecasting MSE ($\times 10^{-3}$) for different areas and methods

	Fontainebleau	Orléans (irregular data)
Our model (with orthogonality)	1.13 ± 0.04 (**1.08**)	3.46 ± 0.20 (**3.22**)
Our model (no orthogonality)	1.23 ± 0.10 (1.13)	4.27 ± 0.55 (3.78)
LSTM	1.50 ± 0.08 (1.38)	5.42 ± 0.84 (4.60)
Long term DMD	2.13	4.33
Direct forecasting (best model)	1.76	8.93

Again, as far as the Orléans time series is concerned, we use the irregular version of the dataset instead of its interpolated version. Thus, the assimilation-forecasting is performed in the same way as for regularly-sampled time series, with the only difference that the term of fidelity to the observed data is computed only on the available timestamps up to time T_{train} rather than for all positive integer timestamps up to time T_{train}. All assimilation-forecasting performance of the tested models with no spatial prior are reported in table 3.1. From comparing the result to those from table 2.1, one can see that all models perform significantly better when used as a dynamical prior for data assimilation than when used in the way in which they were trained to predict from a single input state. In order to better assess the interest of assimilation-forecasting, we also report the best result of table 2.1 in the last line of table 3.1. In addition, the KAE models seem to be the ones that get the greatest benefit from being used in a data assimilation context. Indeed, their mean squared forecasting errors are approximately divided by 2 on the Fontainebleau dataset and by 3 on the Orléans dataset. Besides, the fact that the improvement on the Orléans data is greater than on the Fontainebleau data is also very interesting. We conjecture that this is due to the fact that all models are trained on the Fontainebleau data, and therefore struggle to address the distribution shift between the spatial areas when performing naive forecasting. Resorting to data assimilation seems to partially solve this problem, since the models are still able to produce trajectories that match an observed trajectory from the Orléans area and that extrapolate reasonably well. In particular, one can see that, when used

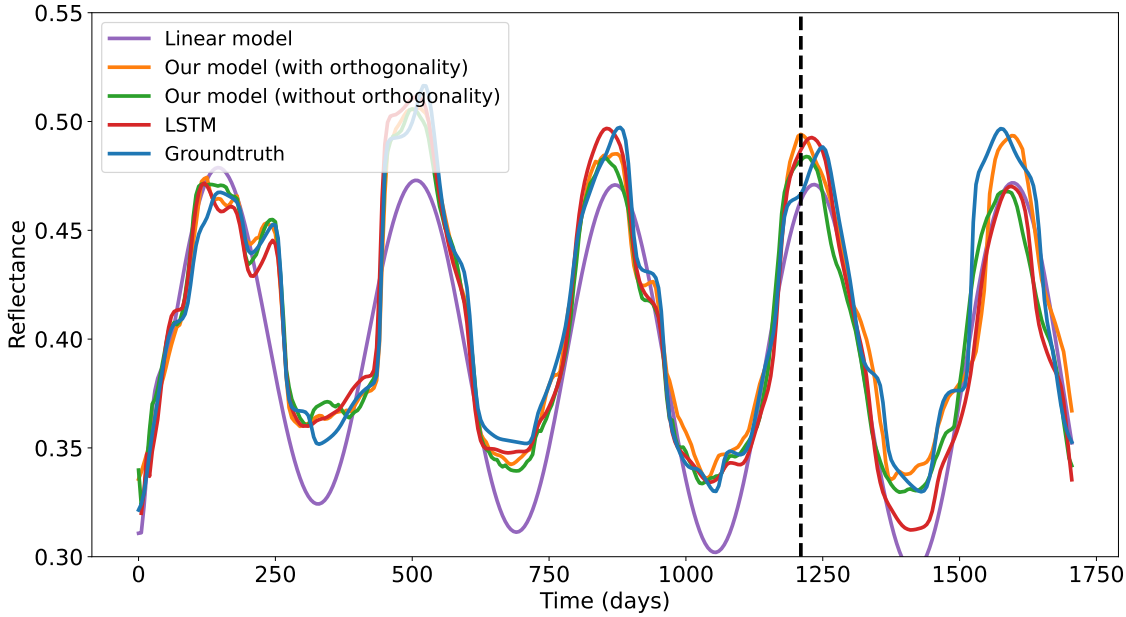

Figure 3.11 – Assimilation-forecasting results with different dynamical priors for the reflectance of the B7 band (in near infrared). The vertical line marks the limit between the assimilated and the extrapolated data. The pixel of interest is marked by a red dot on the Fontainebleau image from figure 2.2.

as a prior for constrained data assimilation, the KAE model trained with an orthogonality loss performs significantly better than the long term DMD on the test Orléans area, which was not the case when testing both models on naive forecasting in section 2.2.1. The LSTM model is more expressive than ours and it is therefore able to fit very precisely the data that it assimilates, yet it tends to overfit on these data, which is why it performs significantly worse than ours on the extrapolation data. On the other hand, long term DMD is a very simple model, making it unable to fit the training data as well as the other models, but it performs reasonably well when transferred to the test Fontainebleau area.

Qualitative results for a given pixel can be observed on figure 3.11, where we show the assimilation-forecasting with the best instance of each model. From this figure, one can see that all models except for the linear one are able to fit the assimilated (training) data very closely but that the differences reside in their respective capacities to extrapolate beyond the training data.

Finally, we now represent the assimilation-forecasting ability of our model on a whole image, in order to assess the spatial coherence of its predictions. Remind that, in Figure 2.5, we had similarly reconstructed the whole spatial forecast of our pixelwise KAE model trained with an orthogonality loss. The predictions were then obtained by simply using the equation (2.7) from an initial state at time 0, and one could see on this picture that the spatial structure of the subsequent images was only partially preserved. This time, we represent the assimilation-forecasting results

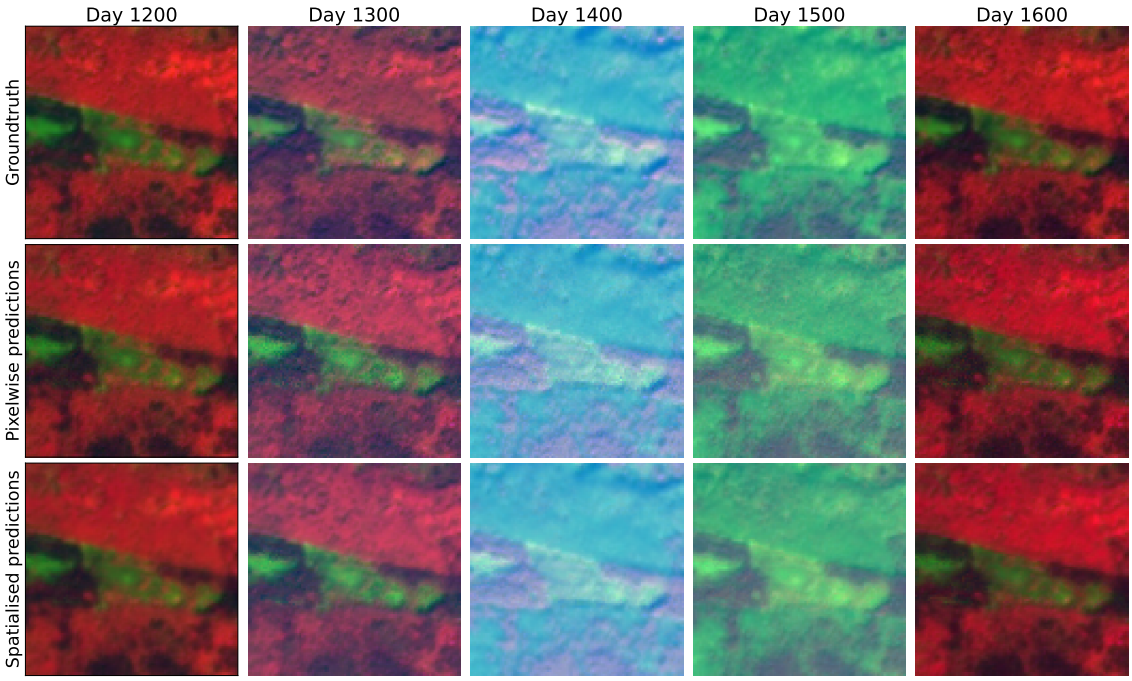

Figure 3.12 – Top: groundtruth images of Fontainebleau, corresponding to test times. Middle: predictions made by our model by assimilating the time series up to day 1200 with a trained model. Bottom: Same as middle but including a spatial regularisation in the variational cost. The colors result from a 3-dimensional principal component analysis (PCA) of the 10 spectral bands performed globally on all the Fontainebleau data. This is much more informative than an RGB composition, mainly because vegetation is very reflective in the near-infrared domain.

obtained with equation (3.26). When no spatial prior $\mathcal{S}$ is used (i.e. when $\beta = 0$), this method still does not leverage the spatial structure of the data, yet it better leverages its temporal structure, which leads to visually more coherent results. Besides, we also test including a very simple Tikhonov regularisation (as defined in equation (3.2)) on the two spatial dimensions of the predictions. As explained earlier, since the predictions for the entire image have to be computed in parallel in order to include the spatial prior in the assimilation framework, the size of the image is limited in practice. Concretely, using a Tesla A100 graphical processing unit (GPU) with 40GB of RAM, we were able to make this computation on a square area of size 100×100 pixels. The spatialised results, with or without the Tikhonov spatial prior, are shown in figure 3.12. When comparing to the results of figure 2.5, one can see that the general performance as well as the spatial coherence of the predictions are largely improved. Note that, even though all the pixels correspond to a forest environment, one can visually see that there are various long-term patterns among the pixels, and that our model can reproduce all of them. In addition, although the spatial prior has a modest influence, it makes the predictions visibly smoother in space.

3.2.2.c Interpolation experiments

Finally, we study the use of a pre-trained KAE model as a prior in data assimilation for the interpolation of Sentinel-2 image time series. This application is qualitatively different from extrapolation, as there are a number of simple methods that can produce a reasonably good interpolation result while being unable to perform a nontrivial extrapolation. One example of such methods is the unconstrained assimilation with a Tikhonov prior, as discussed earlier. The Cressman method is also unable to perform extrapolation. In fact, interpolation can be seen as more simple as extrapolation since it does not require to have a dynamical model for the dataset, and instead relies more heavily on the available data. In particular, in cases such as ours where we still use a dynamical model, having a model that overfits the observed data is less dangerous than in the case of extrapolation. Indeed, the stability induced by the underlying linear evolution of our model does not guarantee a good extrapolation, but it still seems to induce enough regularity to avoid obtaining an unrealistic interpolation when the model is fine-tuned on some observed data.

For this reason, contrarily to what was done in denoising and extrapolation contexts, we will now fine-tune the pre-trained Koopman model to the interpolated observed data. Thus, we will use the cost function introduced in equation (3.20), in which the optimization is performed jointly on the initial latent state $\mathbf{z}_0$ and on the components $\mathbf{K}$ and ψ of the Koopman model. It is important to note that this loss function cannot be successfully minimized from a random parameterization. In particular, using a pre-trained model to initialize $\mathbf{K}$ and ψ is crucial since such a model then benefits from all the knowledge acquired during the pre-training process, enabling it to perform a

better interpolation. In fact, the main interest of performing the fine-tuning compared to simply optimizing on the initial latent state through equation (3.19) is that it enables to fit the observed data much more closely. When comparing against methods that do not rely on a dynamical model and that are potentially unconstrained, being able to closely fit the observed data is indeed an important matter. However, we assume that the observed data might itself be a noisy version of the true underlying process, so that we do not intend to perform a perfect fit. This further justifies the idea of using a pre-trained model from which we optimize with a low learning rate, hence only slightly deviating from it.

For this experiment, we will be working on irregular data from the forests of Fontainebleau and Orléans, using a Koopman model that has been trained on interpolated data from the forest of Fontainebleau only. The irregular data correspond to true satellite observations, on which the only pre-processing was a smoothing with the Cressman method, yet without completing the missing data. We perform this smoothing in order for the incomplete data to still have a structure similar to the interpolated version corresponding to the pre-trained model's training data. The baseline that we will use against the assimilation with a pre-trained Koopman dynamical prior is the constrained assimilation of a periodic prior, which we introduced in section 3.1.2. This is a pretty strong baseline since it leverages the important property that the vegetation image time series have a quasi-periodic behaviour. In particular, with the amount of data that we can access with our testing setup, the assimilation of a periodic pattern performs significantly better than the Cressman interpolation, precisely because the latter is unable to attain the local extrema which are not covered by the observed data. However, the assimilation of a periodic pattern cannot characterize the deviations from this periodicity, while we will show that the assimilation with a dynamical Koopman prior is able to do it.

For both areas, we have at disposal a set of around 85 images of shape $100 \times 100 \times 10$, the first two of which being spatial dimensions and the third one the spectral dimension. These images each have an associated timestamp, and those are irregularly sampled in time since they correspond to the times on which an image without clouds could be retrieved. We randomly separate this set between a set of observed images and a set of test images. This is done by computing a mask with independent Bernoulli variables with parameter $p = 1/2$. Since this procedure leads to highly different masks between samples of the Bernoulli variables, we independently sample 20 different masks, and report the mean and standard deviation of the squared interpolation errors with all masks. For each mask, we solve the variational cost of equation (3.20) with the set of observed timestamps H corresponding to the unmasked timestamps. This gives us a trajectory, from which we compare all values corresponding to the masked timestamps to the groundtruth retained images in order to get an estimate of the interpolation error.

In figure 3.13, we show an example of a repartition between the assimilated and the testing datapoints, along with the interpolations obtained by the different tested models. For this pixel,

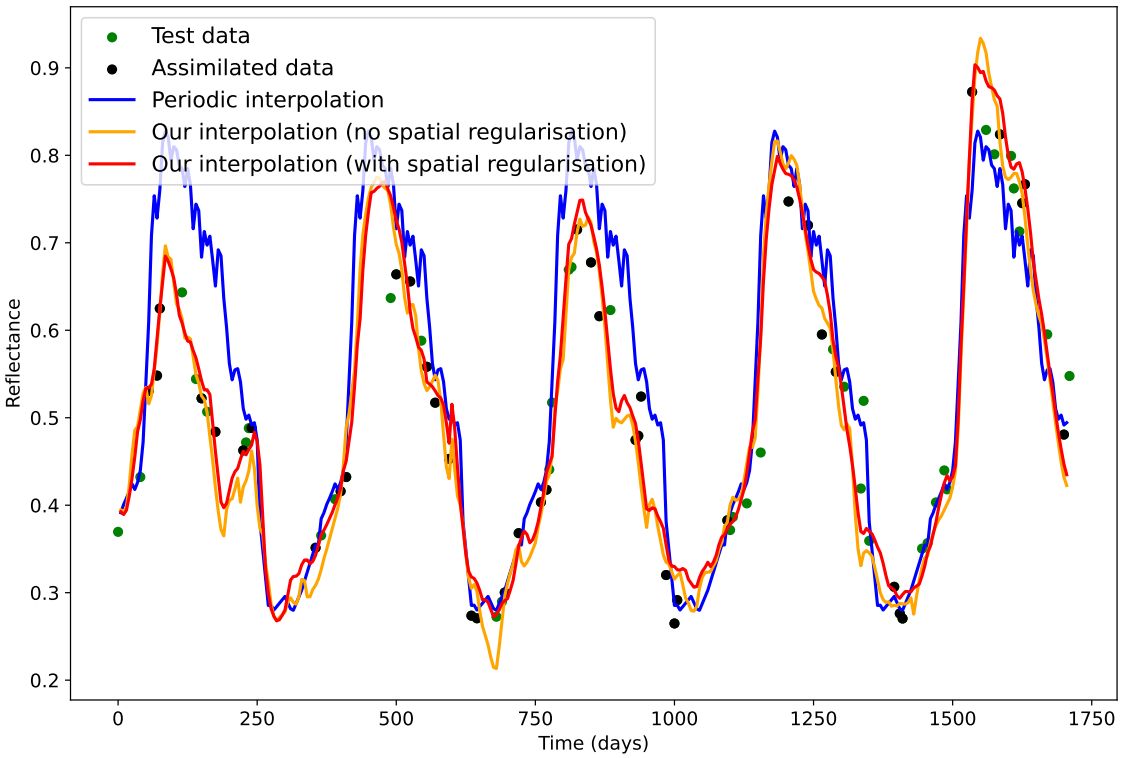

Figure 3.13 – Comparison of interpolations for an Orléans pixel on the B7 band, using periodic interpolation and using data assimilation with our model trained on Fontainebleau data. We show the results of assimilating with or without an additional Tikhonov spatial regularisation in the variational cost. The pixel of interest is marked by a red dot on the Orléans image on figure 2.2.

the data in the first year of observation strongly differ from what is observed in the subsequent years, which is why the periodic pattern cannot perform a good overall interpolation. In contrast, our method is able to model this deviation from the pseudo-periodic pattern, while still leveraging this pattern to provide meaningful estimates of the state on times where no data is observed.

In addition, it appears from this example that the inclusion of a Tikhonov spatial prior in the variational cost seems to bring more regularity to the obtained trajectory. Indeed, it enables to better fit the test datapoints although the (potentially noisy) observed points are slightly less well fitted.

The full results on both 100×100 pixel areas and with the 20 computed masks are shown in table 3.2. One can see from these results that our method, consisting in a constrained assimilation with the fine-tuning of a pre-trained Koopman dynamical prior, significantly outperforms the constrained assimilation of a periodic dynamical pattern. In particular, one can see that our model performs best (in comparison to the baseline) on the Fontainebleau area, which can be explained by the fact that the pre-trained Koopman model was trained on this area. However,

Table 3.2 – Interpolation MSE ($\times 10^{-3}$) for different areas and methods

Periodic	Fontainebleau	Orléans
Periodic assimilation	0.628 ± 0.128	2.37 ± 0.38
Our method	0.349 ± 0.154	2.22 ± 0.37
Our method (with spatial prior)	$\mathbf{0.335 \pm 0.146}$	$\mathbf{2.13 \pm 0.34}$

this framework also outperforms the baseline on the Orléans area, although it was not seen during the training of the dynamical prior. In addition, the results show that the inclusion of a spatial smoothness prior in the assimilation cost can further improve the quality of the interpolation. Again, since the spatial prior that is used is only a very simple Tikhonov term, one could hope that a more complex (potentially data-driven) spatial prior might enable to reach even better results.

3.3 Conclusion

In this part, we first gave an overview of variational data assimilation. In particular, we described how the differentiation of an assimilation cost has recently been made straightforward by automatic differentiation, and how recent works have proposed to use machine learning techniques as a tool for data assimilation.

Then, we have discussed the use of a pre-trained dynamical model as a prior in constrained or unconstrained variational data assimilation. In particular, we have focused on the usage of a KAE, defined and trained as described in chapter 2, for this purpose. We have noticed that this methodology might be described as a form of self-supervised learning. Indeed, the forecasting objective that is used to train a KAE can be seen as a pretext task in order to learn general information about the statistics of the studied dynamical system. When using a model that is trained to this task as a prior for variational assimilation, one can then use it for, e.g., denoising or interpolation of satellite image time series. Such applications, for which the Koopman model was not specifically trained, can be seen as downstream tasks.

Through a series of experiments on Sentinel-2 images, we showed that the assimilation frameworks that we introduced enabled us to solve these downstream tasks, as well as to significantly boost the forecasting performance by taking multiple datapoints rather than a single observed initial state into account. In addition, we showed that it was possible to use a spatial prior on the data in conjunction with the Koopman dynamical prior, therefore enabling to take the spatial structure of the images into account while our Koopman models operate at the pixel level. We showed that including a spatial cost in the assimilation function enabled to obtain

slightly improved interpolation and extrapolation performance, with visually smoother results for the predicted images.

Although the results were significantly improved, a flaw of all approaches considered so far is that we always produced single predictions. In fact, the state of the satellite images is influenced by a lot of factors that are not taken into account, which means that it is impossible to design a model that can predict it with a perfect accuracy. For this reason, having a notion of the uncertainty of a model's prediction is a critical aspect in practice. In order to address this aspect, in the following chapter, we will study variations of our methods from chapters 2 and 3 that enable to perform uncertainty quantification. More precisely, we will discuss stochastic models.

Chapter 4

Stochastic models

In chapters 2 and 3, we have discussed at length a set of data-driven methods based on Koopman operator theory and data assimilation, which enables to solve a variety of tasks including forecasting, interpolation and denoising. However, none of these methods included a quantification of the uncertainty of the predictions that they made, as they all provided pointwise estimates of the state.

In this chapter, we will study stochastic models. Contrarily to the previously described deterministic models, stochastic models produce probability distributions instead of single predictions. In fact, a deterministic model produces a very simple probability distribution in which all of the weight is contained in one point, i.e. a Dirac distribution. The most simple way to obtain a stochastic prediction is to consider several models, possibly with different weights, so that the global probability distribution of the predictions is a - possibly weighted - sum of Dirac distributions. This approach is known as ensembling, and the individual models are generally referred to as members of the ensemble. One could consider a Gaussian distribution, which is characterized by only 2 quantities: a mean state and a covariance matrix for the state. The Kalman filter [82] is a very famous example of a stochastic model from which the predictions follow a Gaussian distribution, based on the assumptions that the observation operator and the dynamical system are linear with Gaussian noise. Some distributions may be even more complex, making it impossible to express them analytically. This is the case for most of generative models that rely on neural network models, such as variational autoencoders [102] (VAEs), generative adversarial networks [103] (GANs) and normalizing flows [51].

First, in section 4.1, we introduce important notions about stochastic models, with the specificities associated to the context of time series data. Then, in section 4.2, we detail our work published in the EUSIPCO 2024 conference [104]. It deals with the joint training of several Koopman autoencoder (KAE) models (from which a single instance was introduced in chapter 2), with an additional variance-promoting loss term which aims at producing higher variances than the ones obtained with individually trained members. Finally, in section 4.3, we introduce our so-called Augmented Variational Invertible Koopman AutoEncoder (AVIKAE) model. This model is based on the KAE framework, with several extensions compared to the implementation of chapter 2. Namely, the encoder is composed of two parts: an invertible part which is a normalizing flow, and an augmentation part, which is a classical multi layer perceptron model which enables

to have a higher-dimensional latent state while keeping the invertibility of the model. Besides, the encoder is variational, which means that the initial latent state associated to an observed initial condition follows a Gaussian distribution. Although the latent dynamical model is deterministic, this enables to obtain a complex probability distribution for the long-term prediction from an initial condition (or when assimilating on several observed points). This work has not been submitted for a scientific publication yet, but is expected to be submitted for a journal publication later.

4.1 Background on stochastic models

4.1.1 Classical methods for fitting a probability distribution

As briefly mentioned in the introduction, stochastic models are models from which the output is a probability distribution rather than a single prediction of the state. One of their key advantages compared to deterministic approaches like the ones presented in chapters 2 and 3 is their ability to produce an estimation of the uncertainty of their predictions, i.e. uncertainty quantification. In order to explain uncertainty quantification, it is crucial to distinguish between two sources of uncertainty: aleatoric uncertainty and epistemic uncertainty. In the machine learning community, the aleatoric uncertainty is defined as coming from the data on which a model is trained. Indeed, the training dataset does generally not contain all information required to generalize to any possible input of the model, and the information that it contains might additionally be inaccurate. Basically, this source of uncertainty is related to the lack of generality of the training data. Then, the epistemic uncertainty is related to the model that learns from the training data. Notably, this source of uncertainty characterizes the unability of the trained model to perfectly fit its training data, due e.g. to a lack of expressivity. Importantly, these definitions are the ones that are most often used in machine learning, but they differ from the mathematical definitions, in which the aleatoric uncertainty only corresponds to the stochastic nature of the data while the epistemic uncertainty refers to everything else. More details about these differing definitions can be found in [105]. For our purpose, the type of uncertainty that we primarily seek to address is the aleatoric (in the machine learning sense) uncertainty. This means that we want to train a model that performs a reasonable quantification of the uncertainty, whether it comes from the intrinsic stochasticity of the data or from an insufficient sampling.

To begin with, let us focus on a context in which one seeks to model a true probability distribution, living in a (possibly multidimensional) space $\mathcal{X} \subset \mathbb{R}^n$. We assume that this probability distribution admits a probability density function (PDF), which we will denote by $\mathcal{P}_r : \mathcal{X} \to \mathbb{R}$. By definition, $\mathcal{P}_r$ should be non-negative and sum to 1 over its domain $\mathcal{X}$. In practice, $\mathcal{P}_r$ is often known only through a finite set of samples $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N) \in \mathcal{X}^N$. Thus, one classically seeks to design a synthetic probability distribution, characterized by its PDF $\mathcal{P}_\theta$,

that best matches the observed datapoints $\mathbf{X}$. A natural way of quantifying whether $\mathcal{P}_\theta$ "matches" $\mathbf{X}$ is the likelihood of $\mathbf{X}$ given the parameters θ of $\mathcal{P}_\theta$. With the assumption that $(\mathbf{x}_1, ..., \mathbf{x}_n)$ are independent samples from $\mathcal{P}_r$, this likelihood can be generically written as

$$\mathcal{L}(\mathbf{X}|\theta) = \prod_{i=1}^{N} \mathcal{P}_\theta(\mathbf{x}_i). \tag{4.1}$$

Since this product can be difficult to manipulate in practice, one often considers the log-likelihood instead in order to deal with a sum. In addition, many popular probability distributions (among which the Gaussian) belong to the so-called "exponential family" [106], which means that their PDF is simplified by the logarithm. The log-likelihood of $\mathbf{X}$ given θ can be written as:

$$\log(\mathcal{L}(\mathbf{X}|\theta)) = \sum_{i=1}^{N} \log(\mathcal{P}_\theta(\mathbf{x}_i)). \tag{4.2}$$

In many cases, one seeks to minimize the well-known negative log-likelihood (NLL) according to θ, which boils down to maximizing the likelihood of $\mathbf{X}$ given θ.

As a simple example, let us assume that we are working in the space $\mathcal{X} = \mathbb{R}$ and that we seek to find the Gaussian distribution that best matches a dataset $\mathbf{X} = (x_1, ..., x_N) \in \mathcal{X}^N$. A 1-dimensional Gaussian distribution is entirely defined by only two scalar parameters: its mean μ and its standard deviation σ. Therefore, the minimization of the NLL in this case can be written as

$$\arg\min_{\mu,\sigma} \sum_{i=1}^{N} -\log(\mathcal{N}(x_i; \mu, \sigma)), \tag{4.3}$$

where $\mathcal{N}(x_i; \mu, \sigma)$ denotes the PDF of a Gaussian of mean μ and standard deviation σ evaluated on x_i. It is a well known result (see e.g. [106]) that the values that minimize the NLL are:

$$\mu^* = \frac{1}{N}\sum_{i=1}^{N} x_i; \quad \sigma^* = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - x^*)^2}. \tag{4.4}$$

Thus, the mean μ^* corresponds to the empirical mean of the dataset, but the variance $(\sigma^*)^2$ is a biased estimator of the empirical variance of $\mathbf{X}$. Indeed, the true empirical variance of $\mathbf{X}$ is $\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - x^*)^2$. Thus, this example shows that the maximum likelihood approach can result in biased estimators. However, this estimator is asymptotically unbiased as N tends to infinity.

This result for the maximum likelihood estimation of Gaussian distributions can be extended for multidimensional probability distributions. However, when $\mathcal{X}$ is of dimension n, the corresponding variance is a matrix $\mathbf{\Sigma}$ of size $n \times n$. Thus, computing the maximum likelihood estimator can become problematic as the dimension n of the data grows high, which is the case

e.g. when $\mathcal{X}$ is a space of images. To alleviate this issue, one can make simplifying assumptions on the covariance matrix $\mathbf{\Sigma}$. For instance, if $\mathbf{\Sigma}$ is assumed to be diagonal, than one only needs to compute n one-dimensional variance parameters, which is a more tractable operation. However, this assumption might be too simplifying to reflect the complexity of the true data distribution in practice.

Setting aside the problem of high dimensions, one can observe that the class of Gaussian distributions is very limited in practice. Notably, a Gaussian distribution is necessarily unimodal, which means that its PDF exhibits only one local maximum. Thus, in order to model multimodal distributions, one must resort to mixtures of Gaussians, i.e. find a weighted sum of several Gaussian distributions with differing parameters. In this case, apart from the mean and variance of each Gaussian distribution, one must determine their weights in the mixture distribution, which have to sum to 1 in order to obtain a valid probability distribution. The parameters of a Gaussian mixture model are generally obtained by running the expectation maximization algorithm [107]. A Gaussian mixture model can provide some interpretation of the dataset by separating the datapoints into fuzzy clusters. Indeed, after the parameters of the Gaussian distributions have been determined, one can compute its probability of belonging to each of the inferred Gaussians, which can be interpreted as clusters. However, Gaussian mixture models remain limited in practice when a dataset is not clearly separated in clusters, when the clusters have concave shapes or when it is difficult to estimate the number of clusters in the dataset. For this reason, many methods have been proposed to model more complex probability distributions using nonparametric approaches, based on e.g. histograms [108], kernel density estimation [109] and k-nearest neighbors [110]. However, these methods struggle to scale in dimension, which is why we will focus our attention on recent approaches based on neural networks.

4.1.2 Neural stochastic models

We have reviewed in the last section the general principles for fitting a probability distribution to an observed dataset, and explained that classical distributions such as Gaussians and mixtures of Gaussians can be ineffective to model complex datasets such as image datasets. We now introduce neural generative models that seek to answer the limits of these classical techniques. As often in machine learning, one gains in expressivity but loses in interpretability. We will however mention some keys for interpreting neural stochastic models. We will focus our analysis on three classes of models: VAEs, normalizing flows and GANs. Of these three classes, the first two will be used in our model described in section 4.3. Other important classes of generative models include restricted Boltzmann machines [111], (deep) Gaussian processes [112] and diffusion models [113].

The main idea of VAEs, normalizing flows and generative adversarial networks is to learn a complex function f_θ that links a chosen simple PDF $\mathcal{P}_Z$, which can be written analytically and sampled from, to the PDF $\mathcal{P}_\theta$ that corresponds to the actual predictions of the model. $\mathcal{P}_Z$

lives in the d-dimensional space $\mathcal{Z} \subset \mathbb{R}^d$ which, in the machine learning terminology, is called the latent space of the generative model. A vector $\mathbf{z} \in \mathcal{Z}$ will generally be referred to as a code or a latent state. The latent dimension d is often, but not always, chosen to be smaller than the dimension n of the data. As a special case, for normalizing flows, we always have that $d = n$, for reasons which will be later described. From now on, we will assume that the simple probability distribution of our choice is a centered Gaussian distribution with an identity covariance, i.e. $\mathcal{P}_Z = \mathcal{N}(0, \mathbf{I}_d)$. This choice is, by far, the most common one in the literature.

The function $f_\theta : \mathcal{Z} \to \mathcal{X}$ is a neural network with a set of parameters θ [1]. It can be understood as a decoder network since it goes from the latent space $\mathcal{Z}$ to the space $\mathcal{X}$ of the data. f_θ takes a latent state $\mathbf{z}$ as its input to produce a point $\mathbf{x}$ that should belong to the same distribution as the data $\mathbf{X}$. More precisely, the push-forward measure of the Gaussian distribution $\mathcal{P}_Z$ through f_θ is the modelled probability distribution $\mathcal{P}_\theta$.

In practice, this means that one can sample a point $\mathbf{z}$ from $\mathcal{N}(0, \mathbf{I}_d)$, and then compute a corresponding point $\mathbf{x} = f_\theta(\mathbf{z})$. This simple procedure amounts to implicitly sampling a point in the modelled distribution $\mathcal{P}_\theta$. However, since the function f_θ is a complex nonlinear neural network, which can be seen as a black box, $\mathcal{P}_\theta$ cannot be defined more explicitly than through this push-forward expression. In particular, when sampling a point $\mathbf{z}$ in $\mathcal{Z}$, the probability of this point according to $\mathcal{P}_Z$ gives an information about the corresponding "probability" of its image $f_\theta(\mathbf{x})$ in $\mathcal{X}$. This observation is the heart of the "truncation trick" introduced for GANs in [114] and extensively used in subsequent works. This trick consists in truncating the samples of $\mathcal{P}_Z$ to a chosen neighborhood of 0 (generally defined by a threshold in the L_1 norm) in order to improve the quality of the generated images, at the cost of a reduced diversity.

However, it is important to note that the probability density of $f_\theta(\mathbf{z})$ in $\mathcal{X}$ does not straightforwardly correspond to the probability density of $\mathbf{z}$ in $\mathcal{P}_Z$. First, if f_θ is built with a classical deep learning architecture such as a multi-layer perceptron or a convolutional neural network, there is not even a guarantee that it is injective, which means that several different inputs in $\mathcal{Z}$ might lead to the same output in $\mathcal{X}$.

Then, even if f_θ is assumed to be bijective, it induces a deformation of the space that has to be taken into account when computing the probability density. To do so, when further assuming that $n = d$, one uses the change of variable formula:

$$\mathcal{P}_\theta(f_\theta(\mathbf{z})) = \mathcal{P}_Z(\mathbf{z}) \left| \det \frac{\partial f_\theta(\mathbf{z})}{\partial \mathbf{z}} \right|^{-1}. \tag{4.5}$$

Thus, in order for this quantity to be computable in practice, the Jacobian $|\det \frac{\partial f_\theta(\mathbf{z})}{\partial \mathbf{z}}|$ of f_θ has to be tractably computable. For the most widespread neural network architectures, there is no

1. In what follows, we will consider θ to be the set of parameters for a whole model, including the parameters of f_θ along with the parameters for the potential other components of the model discussed.

bijectivity property and the Jacobian gets hard to compute as the dimension increases. For these reasons, GANs and VAEs, which rely on such classical architectures, do not enable to compute the likelihood associated to a given point $\mathbf{x}$. Normalizing flows, however, are specifically designed to do so, as they rely on more specific neural network architectures which have two uncommon properties: 1) They are diffeomorphisms, which means that they are differentiable functions with an inverse which is also differentiable. In addition, their inverse function can be written explicitly. 2) Their Jacobian can be easily computed, and so can the Jacobian of their inverse, with only a linear complexity in the size of the state vector. A popular way to obtain these properties is to leverage a set of coupling layers, as first proposed in [51]. In short, coupling layers take a vector $\mathbf{h}_l \in \mathbb{R}^n$ as their input and produce an output vector $\mathbf{h}_{l+1} \in \mathbb{R}^n$ such that

$$\begin{cases} \mathbf{h}_{l+1,\sigma_l(1:m)} = \mathbf{h}_{l,\sigma_l(1:m)} \\ \mathbf{h}_{l+1,\sigma_l(m+1:n)} = c_l(\mathbf{h}_{l,\sigma_l(m+1:n)}, f_l(\mathbf{h}_{l,\sigma_l(1:m)}; \theta_l)) \end{cases} \tag{4.6}$$

where σ_l is a chosen permutation of the n indexes, m is an integer strictly smaller than n, c_l is a coupling law which must be invertible w.r.t its first argument given the second, and f_l is an arbitrarily complex function with parameters θ_l. Then, leveraging the invertibility of c_l one can easily retrieve $\mathbf{h}_l$ from $\mathbf{h}_{l+1}$ with

$$\begin{cases} \mathbf{h}_{l,\sigma_l(1:m)} = \mathbf{h}_{l+1,\sigma_l(1:m)} \\ \mathbf{h}_{l,\sigma_l(m+1:n)} = c_l^{-1}(\mathbf{h}_{l+1,\sigma_l(m+1:n)}, f_l(\mathbf{h}_{l+1,\sigma_l(1:m)}; \theta_l) \end{cases} \tag{4.7}$$

This validates the invertibility property 1). As for the property 2) stating that the Jacobian of the function should be easy to compute, one can write the Jacobian corresponding to equation (4.6) (with a permutation σ_l of its columns) as follows:

$$\det\left(\frac{\partial \mathbf{h}_{l+1}}{\partial \mathbf{h}_l}\right) = \det \begin{pmatrix} \mathbf{I}_m & 0 \\ \frac{\partial \mathbf{h}_{l+1,\sigma_l(m+1:n)}}{\partial \mathbf{h}_{l,\sigma_l(1:m)}} & \frac{\partial \mathbf{h}_{l+1,\sigma_l(m+1:n)}}{\partial \mathbf{h}_{l,\sigma_l(m+1:n)}} \end{pmatrix} = \det \frac{\partial \mathbf{h}_{l+1,\sigma_l(m+1:n)}}{\partial \mathbf{h}_{l,\sigma_l(m+1:n)}}. \tag{4.8}$$

Thus, in order for this quantity to be computable in practice, one simply has to make sure that the link from $h_{l,\sigma_l(m+1:n)}$ to $h_{l+1,\sigma_l(m+1:n)}$ remains simple, while the link from $h_{l,\sigma_l(1:m)}$ to $h_{l+1,\sigma_l(m+1:n)}$ can be as complex as desired, which still enables for a large expressivity. As a practical implementation, one can examine the additive coupling law that was proposed in [51]: $c_l(\mathbf{a}, \mathbf{b}) = \mathbf{a} + \mathbf{b}$. When injecting this law in equation (4.6), we simply have that $\det \frac{\partial \mathbf{h}_{l+1,\sigma_l(m+1:n)}}{\partial \mathbf{h}_{l,\sigma_l(m+1:n)}} = 1$, which leads to a global Jacobian of 1. Thus, this law is said to be volume-preserving. Since this volume preservation was later observed to be too restrictive by [52], the authors introduced the Real-NVP coupling law, which can be written as

$$c_l(\mathbf{a}, \mathbf{b}) = \mathbf{a} \odot \exp(s_l(\mathbf{b})) + t_l(\mathbf{b}), \tag{4.9}$$

where t_l and s_l are respectively translation and scaling operators, parameterized by neural networks. Besides, f_l is an identity function in this model. When injecting this coupling law in equation (4.6), we obtain

$$\det \frac{\partial \mathbf{h}_{l+1,\sigma_l(m+1:n)}}{\partial \mathbf{h}_{l,\sigma_l(m+1:n)}} = \exp(\sum_{i=1}^{m} s_l(\mathbf{h}_{l,\sigma_l(1:m)})_i). \tag{4.10}$$

Thus, this coupling layer does not preserve the volume of the input but it still respects the 2 fundamental properties of normalizing flows.

Although a single coupling layer does not have a high expressivity, and in particular does not even act on the whole input $\mathbf{h}_l$, one can obtain a very complex transformation f_θ by using a succession of several coupling layers. Then, the corresponding Jacobian is simply the product of the Jacobians of all layers and the inverse of the stack is the composition of the inverses. Note that the inverse of f_θ, which we call g_θ, can be seen as an encoder network since it goes from a state space $\mathcal{X}$ to a latent space $\mathcal{Z}$. In order to maximise the expressivity, the permutations σ_l should be changed according to the layer so that, after some number l of layers, every variable of $\mathbf{h}_0$ has acted on every variable of $\mathbf{h}_l$. In particular, the most common choice is to alternate between two permutations σ_{even} and σ_{odd} so that $\sigma_{even}(1:m) = \sigma_{odd}(m+1:n)$ and $\sigma_{even}(m+1:n) = \sigma_{odd}(1:m)$, which necessarily implies that n is even and $m = n/2$. As noted by [51], with this particular choice, $l = 3$ layers are required for every component of $\mathbf{h}_0$ to have a (nonlinear) influence on every component of $\mathbf{h}_l$. Using even more layers can further improve the expressivity of the model at the cost of a longer gradient trace. We refer the readers interested in the universal approximation theory for normalizing flows with coupling layers to the recent work in [115]. In short, this work concludes that affine normalizing flows such as real-NVP [52] are universal approximators in terms of the Kullback-Leibler divergence, while volume-preserving flows such as NICE [51] are not, and are therefore significantly less expressive in theory. In the architecture of our new model presented in section 4.3, we will leverage a normalizing flow composed of a succession of 6 real-NVP layers.

Since normalizing flows are able to tractably compute the PDF that is defined by the push-forward of a Gaussian distribution through their encoder g_θ, it is possible to compute the likelihood of a dataset $\mathbf{X}$ given its parameters θ. Therefore, normalizing flows are trained through the minimization of the negative log-likelihood of $\mathbf{X}$ according to their parameters θ.

For VAEs, since the computation of the likelihood is impossible, the main idea is that the model learns an encoder network g_θ along with its decoder f_θ. g_θ is a stochastic function, in the sense that its output given an input $\mathbf{x}$ is not a pointwise prediction $\mathbf{z}$ but rather a probabilistic distribution $p_\theta(\mathbf{z}|\mathbf{x})$, which is commonly parameterized as a Gaussian distribution with diagonal covariance. The decoder f_θ similarly outputs a Gaussian distribution in $\mathcal{X}$ rather than a pointwise prediction. Thus, the training objective of a VAE is two-fold:

1) One must guarantee that the distribution of $\mathbf{z}$ does indeed follow the chosen prior $\mathcal{N}(0, \mathbf{I}_d)$. This is done by minimizing the Kullback-Leibler divergence between $p_\theta(\mathbf{z}|\mathbf{x})$ and $\mathcal{N}(0, \mathbf{I}_d)$.

2) Like for deterministic autoencoders, one must ensure that $f_\theta \circ g_\theta$ is a good approximation of the identity function. Since one manipulates probability distributions here, one maximises the probability of retrieving an input $\mathbf{x}$ when going through $f_\theta \circ g_\theta$, rather than minimizing the mean squared error between $\mathbf{x}$ and a pointwise prediction.

This two-fold objective stems from the so-called variational lower bound, also called evidence lower bound (ELBO), which was adapted to autoencoders by [102].

The GAN model differs from normalizing flows in the sense that it does not comprise an equivalent of the encoder g_θ. In short, in addition to the decoder f_θ (generally called the generator in the GAN literature), the architecture comprises a discriminator $h_\theta : \mathcal{X} \to [0, 1]$. The objective of h_θ is to evaluate the probability that a point $\mathcal{X}$ comes from the groundtruth distribution $\mathcal{P}_r$ (i.e. the distribution from which $\mathbf{X}$ is sampled) rather than from the synthetic distribution $\mathcal{P}_\theta$ generated by f_θ. Thus, h_θ is expected to assign label 1 to true data samples in $\mathbf{X}$ and to assign label 0 to synthetic data samples generated by f_θ. In the GAN training objective, f_θ and h_θ can be seen as the two players of a minimax game, where the value function is the following:

$$V(f_\theta, h_\theta) = \mathbb{E}_{\mathbf{x}\in\mathbf{X}}[\log h_\theta(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I}_d)}[\log(1 - h_\theta(f_\theta(\mathbf{z})))]. \tag{4.11}$$

The generator f_θ seeks to minimize this score (according to the parameters of f_θ only) while the discriminator h_θ seeks to maximize it (also according to its parameters only). Thus, the minimax game can be summed up as

$$\min_{f_\theta} \max_{h_\theta} V(f_\theta, h_\theta). \tag{4.12}$$

In practice, the optimization consists in alternatively performing gradient steps with respect to the parameters of f_θ and the ones of h_θ [2].

4.1.3 Conditional stochastic models

So far, we have only mentioned unconditional stochastic models. As mentioned earlier, this means that we seek to learn a probability distribution $\mathcal{P}_\theta$ in $\mathcal{X}$, which approximates an unknown groundtruth distribution $\mathcal{P}_r$ of the data. In contrast, conditional stochastic models take a vector $\mathbf{x} \in \mathcal{X}$ as their input to produce a probability distribution $\mathcal{P}_\theta(.|\mathbf{x})$ in $\mathcal{Y}$. This task has more degrees of freedom since there is one different probability distribution for each possible input vector. As a basic example, one can make a distinction between conditional and unconditional

2. Theoretical results state that the optimization process can indeed converge to a Nash equilibrium with a proper choice of the gradient algorithm and of the two learning rate schedules [116]. Besides, there exist variants where the discriminator and generator do not have opposite objectives, i.e. the minimax game is not zero-sum [117]. A popular variant is the Wasserstein GAN [118] in which the objective is based on the Wasserstein or earth mover's distance.

GANs trained on a dataset of images composed of several categories, such as the MNIST dataset [119] which contains handwritten images of each of the 10 digits. When trained on this dataset, a classical (unconditional) GAN should be able to produce new realistic-looking images. To generate an image through this model, one should simply sample a latent state in the pre-determined distribution (usually a multidimensional Gaussian distribution with a diagonal isotropic covariance), use this sample state as an input to the generator and then retrieve the corresponding output, which should be an image. When performing this procedure, one cannot know in advance which of the 10 digits the image will correspond to. Indeed, even during the training procedure, the GAN was not informed about the categories of digits, and it is hard to retrieve this information although the latent space might exhibit some structure. In contrast, when working with a conditional GAN [120], the input to the generator network f_θ has two components: one is, as previously, a randomly sampled latent state, and the second one contains categorical information about the digit that we seek to generate. For this particular example, the categorical information comes in the form of a 10-dimensional one-hot encoded vector. Thus, through this procedure, one can choose to generate images corresponding to a desired class rather than completely randomly sampled images. Conditional GANs have later been applied to more complex image datasets with more numerous categories, such as the ImageNet dataset [121] in [114]. It should be noted that the conditional GAN framework is not limited to conditioning on categorical embeddings. For example, some GAN models are conditioned on an entire input image. Such models can be used, e.g., for image captioning. In [122], the authors train a GAN conditioned on an image, which is able to produce a variety of text captions corresponding to a same input image.

In fact, the framework of conditional generative models is somewhat more general that the one of unconditional generative models. Indeed, this framework implies that the variability of the output can, in a large part, rely on the input rather than on sampling in a latent distribution in $\mathcal{Z}$. From now on, we stop focusing on the classes of models that consist in pushing forward a Gaussian distribution from $\mathcal{Z}$ to $\mathcal{X}$ or $\mathcal{Y}$, i.e. VAEs, GANS and normalizing flows, in order to hold a broader discussion on conditional generative models.

Let us suppose that we seek to solve a regression task, where the output lies in the same space as the input, i.e. $\mathcal{X} = \mathcal{Y}$. The tasks linked to time series analysis that we have examined in chapters 2 and 3 fall under this assumption if we put aside the fact that in most cases their inputs and outputs contain states at several differing times. A deterministic model that solves such a task simply returns a pointwise prediction, while a stochastic model would return a distribution of possible predictions, conditioned on the input to the model. Thus, solving a regression task with a stochastic (rather than a deterministic) model is a form of conditional generative modeling. Although a probability distribution intuitively holds much more information than a single point, one can still represent such a distribution in a synthetic way through a parameterization. For

example, instead of learning a deterministic function $f_\theta : \mathbf{x} \to \mathbf{y}$, one can leverage the stochastic approach $f_\theta : \mathbf{x} \to \boldsymbol{\mu}, \boldsymbol{\sigma}^2$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the parameters of a Gaussian distribution with diagonal covariance. [3]. Compared to a deterministic approach, this only requires doubling the size of the output layer of the neural network that parameterizes f_θ. Besides, the mean $\boldsymbol{\mu}$ of the stochastic model takes the same role as the entire output $\mathbf{y}$ of a deterministic model, so that it should converge to the same value. Through this viewpoint, the variance $\boldsymbol{\sigma}^2$ can be understood as a measure of the uncertainty of the model in its central prediction $\boldsymbol{\mu}$, for every dimension of the state. Although this example discusses a Gaussian output distribution, the same ideas hold for other distributions that can be described by differentiable parameters, such as the sinh–arcsinh (SHASH) [123]. It should be noted however that the loss function used to train such a model should be adapted compared to a deterministic model in order to take into account all parameters of the distribution (notably the variance in the Gaussian case). Thus, one of the most popular approaches is to maximise the likelihood of the data given the model (e.g. [124]), as explained earlier. Another possibility is to minimise a closed form expression of the continuous ranked probability score (CRPS), which we describe in equation (4.13) (see e.g. [125, 126]). Besides the parametric approach consisting in training a model that outputs the parameters of a probability distribution, there exist other simple approaches to train a stochastic model for a regression task. Some popular classes of methods are quantile regression [127], Monte Carlo dropout [128] and Bayesian neural networks [129]. We do not discuss these approaches further here, and refer to [105] for a more detailed review. We will now discuss some details that are useful for the evaluation (and training) of conditional stochastic models.

4.1.4 Evaluating stochastic models

When evaluating a stochastic model, one generally does not have access to the groundtruth probability distribution but only to groundtruth points that are assumed to be all sampled from the same (possibly conditional) probability distribution. Typically, for unconditional probability distributions, the groundtruth comes in the form of a dataset $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)$ of points that are supposedly sampled from a true distribution $p(\mathbf{x})$. In the case of a conditional distribution it comes in the form of a set of couples $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$ such that, for every index $1 \leq i \leq N$, $\mathbf{y}_i$ is supposed to be a sample from the true distribution $p(\mathbf{y}|\mathbf{x}_i)$. In order to evaluate the quality of a stochastic prediction, one cannot simply resort to a classical regression metric such as the mean squared error, which would only be able to evaluate the quality of the mean (or the median) prediction. Instead, one has to use a metric that enables to compare probability distributions. One such metric is the negative log-likelihood, which is widely used in the literature,

3. Here, since the covariance is assumed to be diagonal, it is represented by a vector rather than by a matrix. The assumption of a full covariance matrix implies a quadratic growth of the size of the output with the dimension of the state.

yet this metric is flawed for several reasons, among which its unboundedness. To illustrate this unboundedness, let us suppose that we are computing the negative log-likelihood of a dataset $\mathbf{X}$ given a distribution $f_\theta(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ (hence verifying $\theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$). Then, when assuming that the mean of the Gaussian is equal to one of the observed points in $\mathbf{X}$, i.e. $\boldsymbol{\mu} = \mathbf{x}_i$, we have that the negative log-likelihood of $\mathbf{X}$ tends to negative infinity as $\boldsymbol{\Sigma}$ tends to zero (with any matrix norm). In particular, since a deterministic prediction can be interpreted as a Gaussian distribution with a variance of 0, the associated negative log-likelihood would be either negative infinity if the prediction corresponds (up to machine precision) to a point in $\mathbf{X}$, otherwise it would be positive infinity. Thus, computing the negative log-likelihood makes very little sense for pointwise predictions, or more generally for probability distributions which contain any Dirac delta.

There are other metrics that do not exhibit the unboundedness flaw of the negative log-likelihood. One such popular metric for conditional generation is the continuous ranked probability score (CRPS). First introduced in [130], the CRPS is defined (in 1 dimension) as

$$\text{CRPS}(F, y_{true}) = \int_{-\infty}^{\infty} [F(y) - \mathbb{1}_{y \geq y_{true}}]^2 dy, \tag{4.13}$$

where F is the cumulated distribution function of the output distribution, y_{true} is a pointwise groundtruth and $\mathbb{1}_{y \geq y_{true}}$ is the Heaviside function, taking value 1 for $y \geq y_{true}$ and 0 otherwise. In figure 4.1, we show a graphical interpretation of the CRPS through the "area under the curve" corresponding to the integral in equation (4.13), for a deterministic as well as for a stochastic prediction. Since the definition that we provided only accounts for 1-dimensional spaces, when the space of the predictions is multi-dimensional, we will simply sum the integrals over the marginal cumulated distribution functions of each dimension.

As a special case, when the output distribution is a set of M equiprobable ensemble members $(y_j)_{j=1}^M$, the CRPS can be reframed [131] as

$$\text{CRPS} = \frac{1}{M} \sum_{i=1}^{M} |y_{true} - y_i| - \frac{1}{2} \frac{1}{M^2} \sum_{j=1}^{M} \sum_{k=1}^{M} |y_j - y_k|. \tag{4.14}$$

The first term is the mean absolute error (MAE) and the second term is the halved mean absolute pairwise difference between the ensemble members. In particular, since a deterministic model is nothing more than an ensemble of models with only one member, the CRPS of such a model is simply its mean absolute error. This interpretation shows that the CRPS enables to easily compare deterministic to truly stochastic predictions. In addition, equation (4.14) tells us that the CRPS can be decomposed into 2 components: the first term is an indication of the bias of a prediction, while the second term is related to the variability of the prediction. Thus, since the second term is always negative, one can see, informally, that a model should have a low bias and a

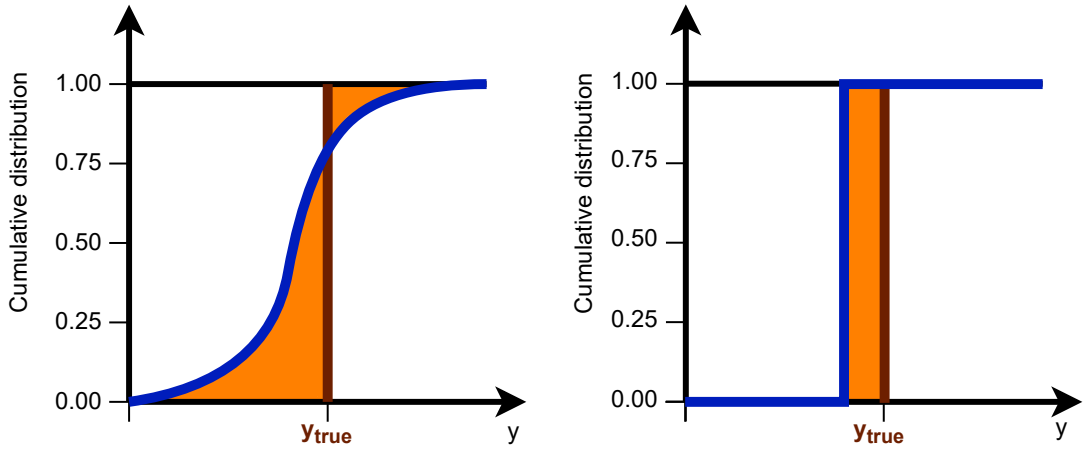

Figure 4.1 – Graphical interpretation of the CRPS. On both graphs, the brown vertical line marks the groundtruth value y_{true}, the blue curve is a cumulated distribution function F and the orange area is the one that appears in the integral of equation (4.13). On the left, the cumulated distribution function corresponds to a prediction containing no Dirac deltas. On the right, it corresponds to a deterministic prediction.

high variability to have a low CRPS. Concretely, an ensemble composed of several members each having a similar bias will always have a better CRPS than a pointwise prediction with the same bias, and the CRPS of this ensemble will decrease as the variance of its predictions increases. By generalizing this idea to other kinds of stochastic models, one can make the observation that stochastic models will usually perform better than deterministic models according to the CRPS when their mean predictions are approximately equivalent.

A qualitative way of evaluating the relevance of a conditional stochastic model is to realize a spread-skill plot, as first proposed in [132]. The skill of a stochastic prediction is defined as the root mean squared error between its mean value and the associated groundtruth. It is usually computed over a large number of data samples $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$ for a stochastic model f_θ which takes a state $\mathbf{x} \in \mathcal{X}$ as input and returns a probability distribution in $\mathcal{Y}$. It can then be written as:

$$\text{Skill}(f_\theta) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} ||\mathbb{E}[f_\theta(\mathbf{x}_i)] - \mathbf{y}_i||^2}. \tag{4.15}$$

The spread of a stochastic prediction, which is defined independently from an associated groundtruth, is the standard deviation of the probability distribution defined by the prediction [4]. Formally, with the same notations as in the previous equation, the spread of f_θ can be expressed

4. The interpretation of a stochastic model through its spread and skill distinguishes, once again, the quality of the mean prediction from the variance of the predictions.

as

$$\text{Spread}(f_\theta) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\mathbb{E}[||f_\theta(\mathbf{x}_i) - \mathbb{E}(f_\theta(\mathbf{x}_i))||^2]}. \tag{4.16}$$

Intuitively, the spread of a stochastic model should be approximately proportional to its skill, as the spread is an indicator of the uncertainty of the model in its mean predicted value. In fact, for a good stochastic model, the spread should be approximately equal to the skill when looking at a high number of predictions. Indeed, as explained in [133], the so-called "groundtruth" is in fact a value that is sampled from the true probability distribution of the data, which we do not have access to and which we seek to approximate through the stochastic model. When averaging over a high number of such samples, one can see that the skill of the groundtruth probability distribution converges to its spread by definition of the variance. Thus, a stochastic model that well approximates the groundtruth probability distribution should also have this property. When the spread of a model is higher than its skill, the model is said to be underconfident as the variance of its predictions exceeds the actual errors that it makes. In contrast, when the spread is lower than the skill, the model is said to be overconfident as it tends to underestimate the errors that it makes.

The idea of the spread-skill plot is to graphically assess the confidence of a stochastic model in its predictions, by plotting the skill of the model as a function of the spread. As mentioned earlier, the spread of the model can only approximate its skill when averaging on many different predictions, i.e. on many pairs $(\mathbf{x}_i, \mathbf{y}_i)$ for a conditional stochastic model. Thus, one first computes a histogram of the values of the spread on a dataset (either a training or a test dataset). Then, one retrieves the mean skill value associated to every bin of this histogram. This gives a set of (spread, skill) pairs, which is visualised in the spread-skill plot. Additionally, an inset plot shows the respective number of samples that each bin contains, in order to get an idea of the relative importance of the corresponding points in the plot. One advantage of this approach is that one can get a sense of the mean value of the skill for differing ranges of the spread. This can be useful as a stochastic model might be underconfident for some spread values and overconfident for some other spread values. When the spread is at least proportional to the skill, one often observes that the model is unable to provide high enough uncertainties for the cases that are hardest to predict, hence resulting in predictions that are actually particularly overconfident when the spread is high.

There are two quantities that leverage the same information as the spread-skill plot in order to give a synthetic indication of the quality of the uncertainty quantification. Those indicators are the spread-skill reliability (SSREL) and the spread-skill ratio (SSRAT). The SSREL is the sum of the absolute distances (in terms of skill) between the 1:1 line and the spread-skill plot, weighted by the number of samples associated to each bin. Thus, the SSREL is positive with a value of zero signifying a perfect uncertainty quantification. However, the SSREL is affected by

the choice of the binning process. In contrast, the SSRAT is independent from the choice of the histogram, as it represents the global ratio between the spread and the skill of a stochastic model. Thus, the SSRAT is a positive quantity, for which a value of 1 characterizes a well calibrated model, while a value smaller or greater than 1 respectively characterizes an overconfident or an underconfident model.

In the next section, we will use the spread-skill plot, the SSREL and the SSRAT to evaluate ensembles of KAE models.

4.2 Ensembles of Koopman autoencoders

In this section, we will perform stochastic forecasting of Sentinel-2 image time series using ensembles of Koopman models. This section is in large part based on our EUSIPCO 2024 article [104]. The basic idea that we develop is to train in parallel several instances of the KAE model from chapter 2.

4.2.1 A brief review of ensembles of models in machine learning

Let us first briefly review the methods based on ensembling machine learning models, and more specifically neural networks. Ensembles of models generally consist in several instances of a same model, which can differ by various factors such as their initial parameterization or the set of data that they have been trained on. For classical models such as decision trees, the ensemble consists in individually simple models and the variability generally comes from the fact that the individual models (i.e. the members of the ensemble) are trained on different subsets of the data. The specific choice of how to design the sets of training data for the different members has been the object of much attention, as it is the heart of renowned methods such as bootstrap aggregating (also known as bagging) [134]. For random forests [135], which are the most popular framework for ensembles of decision trees, an important principle is that the members should not be trained on the same data in order to promote their diversity. However, for ensembles of deep learning models, it has been observed that training the members on different subsets of the data is not always beneficial since the performance of individual members largely depend on the amount of data that they are trained on. Notably, [136] reports significantly better accuracy for ensembles of models trained on all data with different initialisations than for ensembles trained with bagging, on several image classification benchmarks. For this reason, the variability of an ensemble of neural networks generally comes from the differing initialisations of the members rather than from the member's training data [137]. This choice makes sense for neural networks since their objective function usually has multiple local minima, contrarily to a decision tree loss function, which means that the final model largely depends on the initial values of the parameters. Thus, one hopes that the members of an ensemble of neural networks will learn different features

although they are all trained on the same dataset. While ensembles of neural networks were primarily used to boost the deterministic prediction performance by averaging the results of the members (see e.g. [138] for a review), it has been later noticed by [137] that the variance of the member's prediction can be of use for uncertainty quantification. They even showed that the uncertainty estimates produced by a deep ensemble can outperform those of a Bayesian neural network based on either probabilistic backpropagation [139] or Monte Carlo dropout [128]. However, in contrast to our methods that are detailed later, [137] identified the independence of the training process for each member as a key element in training a deep ensemble. The vast majority of subsequent works followed this principle, with the notable exception of [140], which we later compare to our approaches.

4.2.2 Our methods for training an ensemble of Koopman autoencoders

In our case, when training ensembles of KAE models, we will observe that ensembles of independently trained models tend to be highly overconfident. Concretely, the variance of their predictions is usually much lower than the mean squared error of their mean predictions with respect to the groundtruth. In order to increase the spread of our ensembles of KAEs, we will introduce two loss functions that enable to increase the variance between the members of a trained ensemble. One of these loss functions directly favors the spread of the ensemble while the other one is derived from the CRPS from equation (4.14).

As a reminder, an individual KAE, as defined in section 2.2, is composed of 3 components: an encoder ϕ, a decoder ψ and a matrix $\mathbf{K}$. Thus, the set θ of trainable parameters for this model contains the trainable parameters of ϕ and ψ as well as the coefficients of $\mathbf{K}$. Let us now suppose that we are working with a n-dimensional dynamical system and that our training dataset $(\mathbf{x}_{i,t})_{1\leq i\leq N, 0\leq t\leq T}$ is composed of N time series of length $T+1$ resulting from the dynamical system of interest. Note that $\mathbf{x}_{i,t}$ is a n-dimensional vector. In what follows, we may drop the index i to designate any of these time series. The loss function for a single KAE model is composed of the terms:

$$L_{pred}(\theta) = \sum_{i=1}^{N}\sum_{\tau=1}^{T} ||\mathbf{x}_{i,\tau} - \psi(\mathbf{K}^{\tau}\phi(\mathbf{x}_{i,0}))||^2 \tag{4.17}$$

$$L_{ae}(\theta) = \sum_{i=1}^{N}\sum_{t=0}^{T} ||\mathbf{x}_{i,t} - \psi(\phi(\mathbf{x}_{i,t}))||^2 \tag{4.18}$$

$$L_{lin}(\theta) = \sum_{i=1}^{N}\sum_{\tau=1}^{T} ||\phi(\mathbf{x}_{i,\tau}) - \mathbf{K}^{\tau}\phi(\mathbf{x}_{i,0})||^2 \tag{4.19}$$

$$L_{orth}(\theta) = L_{orth}(\mathbf{K}) = ||\mathbf{K}\mathbf{K}^T - \mathbf{I}||_F^2 \tag{4.20}$$

where $||.||_F$ denotes the Frobenius norm. The terms L_{pred}, L_{ae}, L_{lin} and L_{orth} are respectively the

prediction loss, auto-encoding loss, linearity loss and orthogonality loss. We refer to sections 2.1.3 and 2.2.1 for an extensive interpretation of each of these loss terms. All of them can be weighted equally except for the orthogonality loss for which a suitable weight α has to be found, resulting in the global loss function

$$L(\theta) = L_{pred}(\theta) + L_{ae}(\theta) + L_{lin}(\theta) + \alpha L_{orth}(\theta). \tag{4.21}$$

Let us now suppose that we are training an ensemble of M instances of this model. We then denote the parameters of these instances as $\Theta = (\theta_1, ..., \theta_M)$. The instances can be trained in parallel by defining a global loss function which is simply the sum of the loss functions for each of the M instances:

$$\mathcal{L}_{independent}(\Theta) = \frac{1}{M} \sum_{j=1}^{M} L(\theta_j). \tag{4.22}$$

Using this loss function is equivalent to training the M members of the ensemble sequentially and independently. As we will show experimentally, this may lead to a low diversity of the members, since the instances tend to all capture similar features in the data, which is undesirable in ensemble learning.

Given an input state $\mathbf{x}_{i,0} \in \mathbb{R}^n$, the outputs of the members, obtained by equation (2.7), will be denoted as $\hat{\mathbf{x}}_{i,t,j}$ with $0 \leq t \leq T$ denoting time and $1 \leq j \leq M$ denoting the member. Then, we will drop the member index j to denote by $\hat{\mathbf{x}}_{i,t}$ the mean prediction of an ensemble of models, i.e.

$$\hat{\mathbf{x}}_{i,t} = \frac{1}{M} \sum_{j=1}^{M} \hat{\mathbf{x}}_{i,t,j}. \tag{4.23}$$

As previously mentioned when motivating the relevance of spread-skill plots, in an ensemble one wants the spread of the predictions to be approximately equal to its skill when averaging on many initial conditions. However, as we will show later, training an ensemble of Koopman models independently from each other through the loss of equation (4.22) leads to an overconfident model. For this reason, we introduce an additional loss term which acts jointly on each model, promoting the inter-model variance of the predictions:

$$L_{var}(\Theta) = -\frac{1}{M} \sum_{j=1}^{M} \sum_{i=1}^{N} \sum_{t=1}^{T} ||\hat{\mathbf{x}}_{i,t,j} - \hat{\mathbf{x}}_{i,t}||^2. \tag{4.24}$$

Then, this new loss term can be introduced in the global loss function of an ensemble, which results in

$$\mathcal{L}_{var,\lambda}(\Theta) = \mathcal{L}_{independent}(\Theta) + \lambda L_{var}(\Theta), \tag{4.25}$$

where the case $\lambda = 0$ corresponds to equation (4.22). Note that computing this loss function requires the computation and differentiation of the mean prediction of the ensemble's members.

For this reason, it requires the joint optimization of all members, which puts more constraints on the GPU memory than the independent training approach of equation (4.22) since it becomes impossible to train the members sequentially.

We will now describe the range in which the hyperparameter λ from equation (4.25) should be chosen. Let us first note that, for any valid indexes i, t,

$$\frac{1}{M}\sum_{j=1}^{M}||\hat{\mathbf{x}}_{i,t,j} - \mathbf{x}_{i,t}||^2 = \frac{1}{M}\sum_{j=1}^{M}||\hat{\mathbf{x}}_{i,t,j} - \hat{\mathbf{x}}_{i,t}||^2 + ||\hat{\mathbf{x}}_{i,t} - \mathbf{x}_{i,t}||^2 \tag{4.26}$$

This is a well-known statistical result, which is not specific to our case but remains true for any value $\mathbf{x}_{i,t}$ and predictions $\hat{\mathbf{x}}_{i,t,j}$ with mean $\hat{\mathbf{x}}_{i,t}$. See, e.g., the theorem 2 of chapter 6 in [141] for a demonstration. Using this result, we have that

$$\sum_{j=1}^{M}||\hat{\mathbf{x}}_{i,t,j} - \mathbf{x}_{i,t}||^2 - \lambda\sum_{j=1}^{M}||\hat{\mathbf{x}}_{i,t,j} - \hat{\mathbf{x}}_{i,t}||^2 = (1-\lambda)\sum_{j=1}^{M}||\hat{\mathbf{x}}_{i,t,j} - \hat{\mathbf{x}}_{i,t}||^2 + M||\hat{\mathbf{x}}_{i,t} - \mathbf{x}_{i,t}||^2 \tag{4.27}$$

The first and second term of the left member of this equation are respectively the prediction loss and the variance loss associated to the training sample $\mathbf{x}_{i,t}$. From this equation, one can first notice that, if $\lambda \leq 1$, then the expression is trivially positive and can therefore be interpreted as a valid loss function. However, when $\lambda > 1$, this expression is not negatively bounded. Indeed, since $\mathbf{x}_{i,t}$ is a constant vector, one can simply choose arbitrarily large member predictions $\hat{\mathbf{x}}_{i,t,j}$ satisfying $\hat{\mathbf{x}}_{i,t} = 0$ through equation (4.23) (e.g. the members can be arranged in opposite pairs) in order for the expression (4.27) to become arbitrarily low. As this analysis remains true for any indexes $1 \leq i \leq N$ and $1 \leq t \leq T$, the prediction loss term (4.17) can counterbalance the variance loss term (4.24) as long as $0 \leq \lambda \leq 1$ in equation (4.25). Qualitatively, the case $\lambda = 1$ means that the individual errors made by each member are replaced by the squared error of the mean of the members in the loss function. Thus, although this case does not lead to a negatively unbounded loss, it corresponds to a situation where the prediction loss term for the individual members is somehow negated and the ensemble is only expected to produce an accurate mean prediction. From our observation in section 2.2.2.b that each of the four loss terms is important when training a KAE model, this does not look to be a desirable situation. We will show in our experimental results that the resulting ensemble is indeed underconfident in its predictions in this case.

We emphasize that the previous work of [140] was, to our knowledge, the only one that advocated for directly including the negative variance in the loss function for a deep ensemble. However, they work in a setup where the predictions of the models are all bounded by design, which eliminates the risk of obtaining an ensemble of infinite variance. Therefore, the above

analysis for the acceptable values of λ is novel, and corresponds to a case where the member predictions are not bounded.

We now introduce an alternative loss function for an ensemble of KAEs, this time inspired from the CRPS. While easy to compute for small ensembles, the formulation of equation (4.14) gets very costly to compute as the number M of members gets higher because of the pairwise differences in the second term. Concretely, the pairwise differences make the number of terms grow quadratically with the number M of members, while the number of terms to compute in equation (4.24) only grows linearly with the size of the ensemble. This is not an important issue for evaluation since one may simply decompose the computation into any needed number of steps. However, training an ensemble of deep learning models with the CRPS requires computing (and differentiating) this metric multiple times, which can become extremely slow as the number of batches has to be largely increased to fit the GPU memory. Therefore, we propose to replace the mean of the pairwise absolute differences by the mean of the absolute differences to the mean prediction. This comes in the form of the new loss term

$$L_{abs}(\Theta) = -\frac{1}{2}\frac{1}{M}\sum_{j=1}^{M}\sum_{i=1}^{N}\sum_{\tau=1}^{T}|\hat{\mathbf{x}}_{i,t,j} - \hat{\mathbf{x}}_{i,t}|. \tag{4.28}$$

One can see that this new loss term is very similar to the variance-promoting loss term of equation (4.24), although it uses absolute differences instead of the squared L^2 norm. Using this term as a proxy to the CRPS can be justified by the following theorem:

Theorem 2. *Let $(\hat{\mathbf{x}}_j)_{j=1}^M$ a set of predictions performed by an ensemble of M models and $\hat{\mathbf{x}}$ the mean of these predictions. Then,*

$$\frac{1}{M}\sum_{j=1}^{M}|\hat{\mathbf{x}}_j - \hat{\mathbf{x}}| \leq \frac{1}{M^2}\sum_{j=1}^{M}\sum_{k=1}^{M}|\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_k| \leq \frac{2}{M}\sum_{j=1}^{M}|\hat{\mathbf{x}}_j - \hat{\mathbf{x}}|. \tag{4.29}$$

Proof. Let us first prove the first inequality. This is done by decomposing $\hat{\mathbf{x}}$ into the sum that defines it, and then using a triangular inequality, as detailed in the following development:

$$\begin{aligned}
\frac{1}{M}\sum_{j=1}^{M}|\hat{\mathbf{x}}_j - \hat{\mathbf{x}}| &= \frac{1}{M}\sum_{j=1}^{M}|\hat{\mathbf{x}}_j - \frac{1}{M}\sum_{k=1}^{M}\hat{\mathbf{x}}_k| \\
&= \frac{1}{M^2}\sum_{j=1}^{M}|M\hat{\mathbf{x}}_j - \sum_{k=1}^{M}\hat{\mathbf{x}}_k| \\
&= \frac{1}{M^2}\sum_{j=1}^{M}|\sum_{k=1}^{M}(\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_k)| \\
&\leq \frac{1}{M^2}\sum_{j=1}^{M}\sum_{k=1}^{M}|\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_k|
\end{aligned} \tag{4.30}$$

We indeed obtain the desired result.

As for the second inequality of the theorem, it can be obtained by using a triangular inequality in order to make $\hat{\mathbf{x}}$ appear and then rearranging the terms, as follows:

$$\begin{aligned}\frac{1}{M^2}\sum_{j=1}^{M}\sum_{k=1}^{M}|\hat{\mathbf{x}}_j-\hat{\mathbf{x}}_k| &\leq \frac{1}{M^2}\sum_{j=1}^{M}\sum_{k=1}^{M}(|\hat{\mathbf{x}}_j-\hat{\mathbf{x}}|+|\hat{\mathbf{x}}_k-\hat{\mathbf{x}}|)\\ &= \frac{1}{M^2}(M\sum_{j=1}^{M}|\hat{\mathbf{x}}_j-\hat{\mathbf{x}}|+M\sum_{k=1}^{M}|\hat{\mathbf{x}}_k-\hat{\mathbf{x}}|) \qquad (4.31)\\ &= \frac{2}{M}\sum_{j=1}^{M}|\hat{\mathbf{x}}_j-\hat{\mathbf{x}}|.\end{aligned}$$

□

Thus, the loss from equation (4.28) is a lower bound (in absolute value) to the second term of the expression of the CRPS in equation (4.14). Thus, one can design a loss function that leverages it to take inspiration from the CRPS. To do so, we simply use variants of the basic loss terms in equations (4.17) to (4.20) where the squared L^2 norms are replaced by L1 norms. This substitution is performed in order for all loss terms to be consistent with the L_{abs} loss term from equation (4.28). Formally, we use:

$$L_{pred,1}(\theta)=\sum_{i=1}^{N}\sum_{\tau=1}^{T}|\mathbf{x}_{i,\tau}-\psi(\mathbf{K}^{\tau}\phi(\mathbf{x}_{i,0}))| \qquad (4.32)$$

$$L_{ae,1}(\theta)=\sum_{i=1}^{N}\sum_{t=0}^{T}|\mathbf{x}_{i,t}-\psi(\phi(\mathbf{x}_{i,t}))| \qquad (4.33)$$

$$L_{lin,1}(\theta)=\sum_{i=1}^{N}\sum_{\tau=1}^{T}|\phi(\mathbf{x}_{i,\tau})-\mathbf{K}^{\tau}\phi(\mathbf{x}_{i,0})| \qquad (4.34)$$

$$L_{orth,1}(\theta)=L_{orth}(\mathbf{K})=||\mathbf{K}\mathbf{K}^{T}-\mathbf{I}||_{1,1}, \qquad (4.35)$$

where $||.||_{1,1}$ is the sum of the absolute values of the coefficients of a matrix. Note that, when averaged over a set of members, the prediction loss (4.32) corresponds to the first term of the expression of the CRPS in equation (4.14).

Then, one can introduce an equivalent of the memberwise loss of equation (4.21) as

$$L_1(\theta)=L_{pred,1}(\theta)+L_{ae,1}(\theta)+L_{lin,1}(\theta)+\alpha L_{orth,1}(\theta). \qquad (4.36)$$

Finally, for an ensemble of M members with parameters $\Theta=(\theta_1,...,\theta_M)$, one can use the following

loss function:

$$\mathcal{L}_{CRPS}(\Theta) = \sum_j L_1(\theta_j) + \lambda L_{abs}(\Theta), \tag{4.37}$$

where λ is usually set to 1. It should be noted that this is not the true CRPS loss since the pairwise differences between the predictions are only approximated, and there are auxiliary loss terms.

4.2.3 Experiments on Sentinel-2 data

We now apply the methods introduced in the previous section for the task of forecasting time series of satellite images. We train several ensembles of 8 members, with the members all having the same architecture as for the Sentinel-2 experiments of chapters 2 and 3. Between the different ensembles, only the training loss functions differ. We use the Fontainebleau and Orléans time series that were introduced in chapter 1 and which were already used for the experiments of chapters 2 and 3. For this experiment, we use the interpolated versions of the time series only, and we do not use the assimilation frameworks introduced in chapter 3. The ensembles are trained on the Fontainebleau area, and then tested on two tasks: 1) extrapolating on the training Fontainebleau area to times unseen during training 2) predicting from an initial time on the test Orléans area. Thus, task 1) is used to test temporal extrapolation while task 2) tests the ability to transfer the knowledge to a new area with a distribution shift. Thus, the experimental setup is kept quite simple, and the goal of the experiment is to assess the impact on the quantification of the uncertainty when using the diversity-promoting loss functions of equations (4.25) and (4.37), compared to the basic approach of training the members independently from each other with equation (4.22). In order to perform qualitative and quantitative comparisons between the loss functions, we will use the spread-skill plot with its associated SSRAT and SSREL, as well as the CRPS (which, as mentioned in section 4.1, is primarily a metric for evaluating stochastic models).

First, we demonstrate through an example our claim from section 4.2.2 that ensembles of independently trained KAEs tend to perform overconfident predictions. We intentionally pick a pixel for which most models struggle to perform good predictions, and we plot on figure 4.2 its groundtruth trajectory along with the predictions performed by 2 ensembles of models. These ensembles are trained with the loss function of equation (4.25), with a value of λ equal to either 0 (i.e. independently trained members) or 0.5. One can see that, although both ensembles are highly biased, the ensemble trained with a variance-promoting loss term has a higher inter-model variance. This means that this ensemble is less confident in its prediction, which is desirable here because the mean squared error is high.

In order to get a more comprehensive idea corresponding to this intuition, we show in figure 4.3 the spread-skill plots of several ensembles for the two tasks corresponding to the two identified

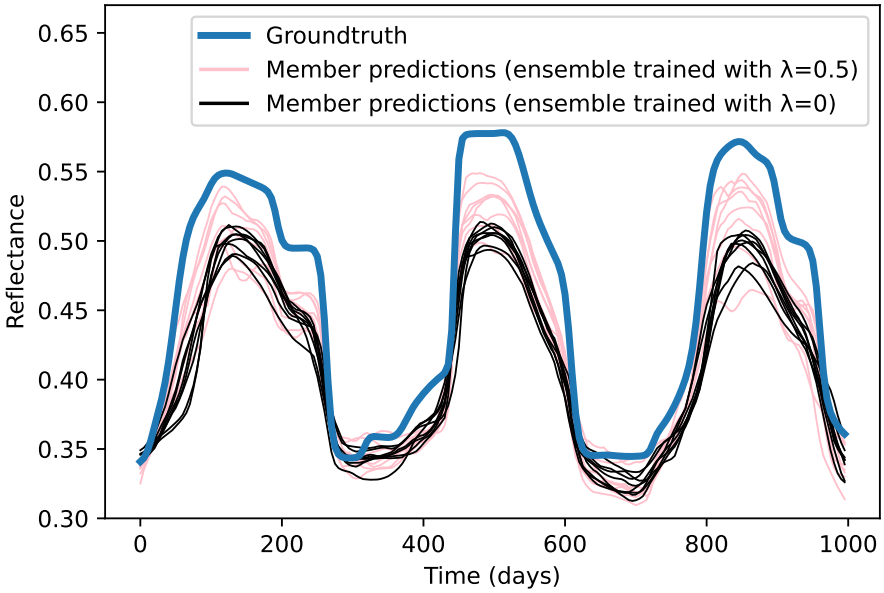

Figure 4.2 – Forecasting from time 0 by two ensembles for the reflectance of the B7 band (in near infrared) for a Fontainebleau pixel. Here, both ensembles are biased, but the ensemble trained with a variance-promoting loss term ($\lambda = 0.5$) yields a higher inter-member variance, and hence a better uncertainty estimate, than the ensemble of independently trained models ($\lambda = 0$).

spatial areas. We test several values of λ from 0 to 1 for training with the variance-promoting loss function of equation (4.25), and we also test a model trained with the loss that approximates the CRPS in equation (4.28) with $\lambda = 1$.

Several conclusions can be drawn from figure 4.3. First, the ensemble with independently trained models (corresponding to $\lambda = 0$) is highly overconfident, which quantitatively confirms the intuition gained from figure 4.2. Then, one can clearly see that the ensembles get less confident as the value of λ increases. The case of $\lambda = 1$ is a limit case, and results in the only model that is severely underconfident on both the training and the test areas. The value $\lambda = 0.99$ yields the best spread-skill ratios, and the model trained with a proxy to the CRPS lies in between $\lambda = 0.5$ and $\lambda = 0.9$.

Finally, we show in figure 4.4 the CRPS of the ensembles as a function of the value of λ used in their training function (4.25). Note that lower values are better for the CRPS metric. Again, one can see that a well-chosen value of λ can significantly improve the performance compared to an ensemble of independently trained members ($\lambda = 0$). The values $\lambda = 0.5$ and $\lambda = 0.9$ seem to be good compromises between the CRPS on the two tasks, while the model trained with a loss function similar to the CRPS also performs well on both. It should be noted that, as could be expected, stochastic models trained with the (real) CRPS as their objective function usually perform better according to the CRPS than models trained with any other criterion.

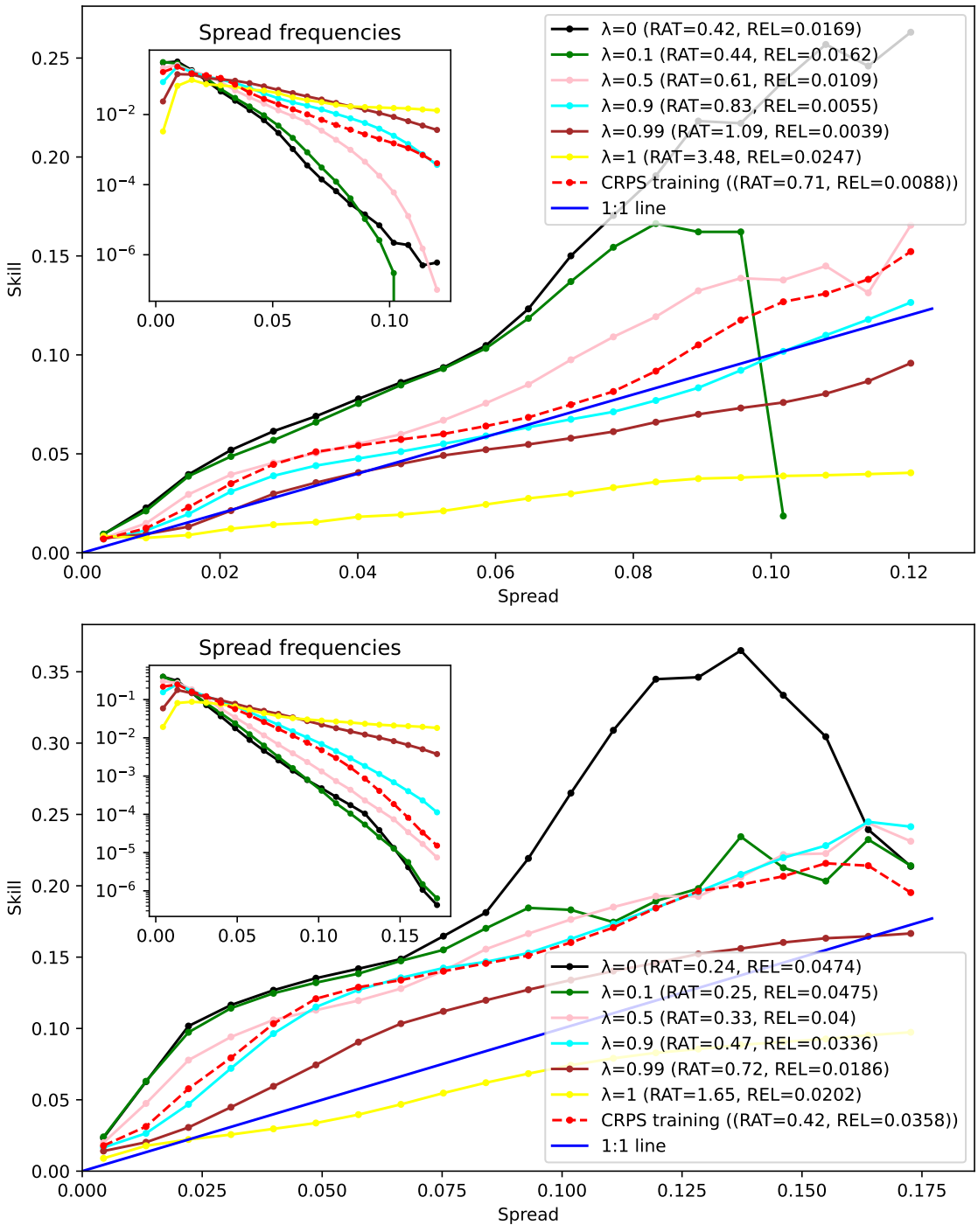

Figure 4.3 – Spread-skill plots for two different datasets. Top: spread-skill plot of extrapolation on the training Fontainebleau area. Bottom: spread-skill plot of predictions from time 0 on the test Orléans area.

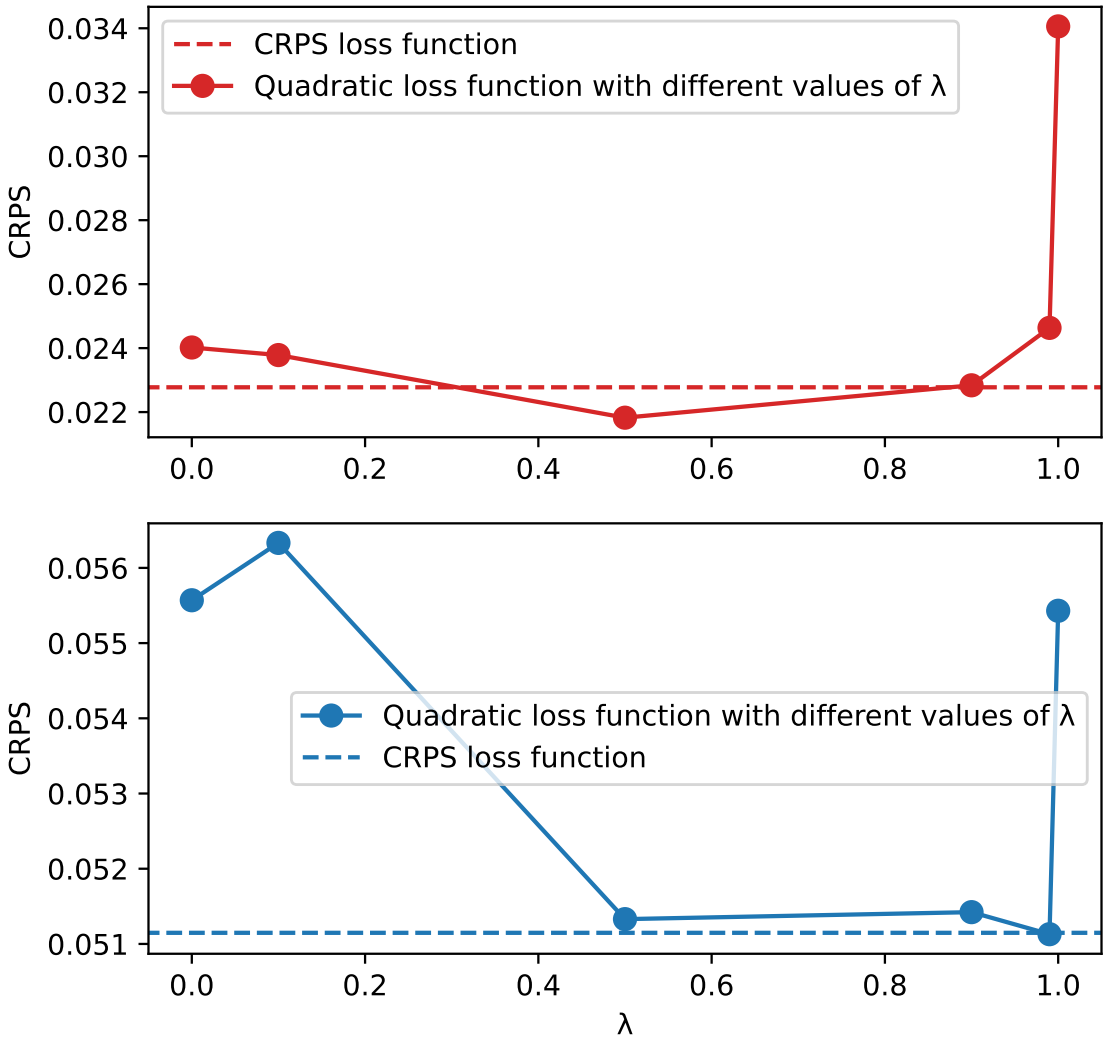

Figure 4.4 – CRPS of ensembles of KAEs according to the weight λ of their variance-promoting loss term during training. Top: extrapolation on training Fontainebleau area. Bottom: transfer to test Orléans area. The represented values of λ are $0, 0.1, 0.5, 0.9, 0.99, 1$.

To sum up, after noticing that ensembles of KAEs tend to be very overconfident when their members are trained independently, we introduced a variance-promoting loss term which encourages the members of an ensemble to produce more diverse forecasts. We studied, both analytically and empirically, the influence of this term on the trained ensembles according to its weight relatively to the other loss terms. We found that, according to several metrics, the quality of the uncertainties produced by the ensembles improves as the weight of the variance-promoting loss term gets closer to its theoretical limit of 1.

While ensembles of deep models only allow for a rudimentary stochastic prediction corresponding to a sum of M Dirac distributions, in the next section we will study a new method based on KAEs which enables to produce Gaussian predictions in the latent space of its encoder, corresponding to arbitrarily complex probability distributions in the state space.

4.3 Augmented Variational Invertible Koopman AutoEncoder

In this section, we present a new Koopman-inspired neural network architecture, called the Augmented Variational Invertible Koopman AutoEncoder (AVIKAE). This architecture has been developed in late stages of the thesis work and has not been presented in a scientific article yet.

We will first clarify the acronym of our model letter by letter, starting from the Koopman AutoEncoder (KAE) framework described in chapter 2. Then, we will perform an analysis of the statistical properties of this model and of the possibilities that it offers for probabilistic processing (forecasting, interpolation, denoising) of time series. Finally, we will present experiments on Sentinel-2 time series, where we demonstrate some use cases for the AVIKAE and compare it to the ensemble method of section 4.2 and to simpler variants of stochastic Koopman autoencoder models.

4.3.1 Description of the AVIKAE architecture

In chapter 2, we described the KAE model which, in a few words, consists in an autoencoder (ϕ, ψ) and a matrix $\mathbf{K}$, where the autoencoder goes to and from a Koopman Invariant Subspace (KIS) and the matrix $\mathbf{K}$ enables to advance a latent state vector from this space by one time step. We briefly mentioned that some recent works [50, 49] proposed to replace the classical neural network architectures that are used for (ϕ, ψ) (i.e. generally multi-layer perceptrons) by a model from which the inverse is known up to machine precision. Thus, instead of training an encoder ϕ and a decoder ψ from which the composition is approximately the identity function, one can train a single (invertible) encoder neural network ϕ, and use its known inverse ϕ^{-1} as the associated decoder. We show in figure 4.5 a schematic view of such a model, which we call an Invertible Koopman AutoEncoder (IKAE). This approach has several advantages over the classical autoencoder approach. First, the reconstruction of a state from its corresponding latent

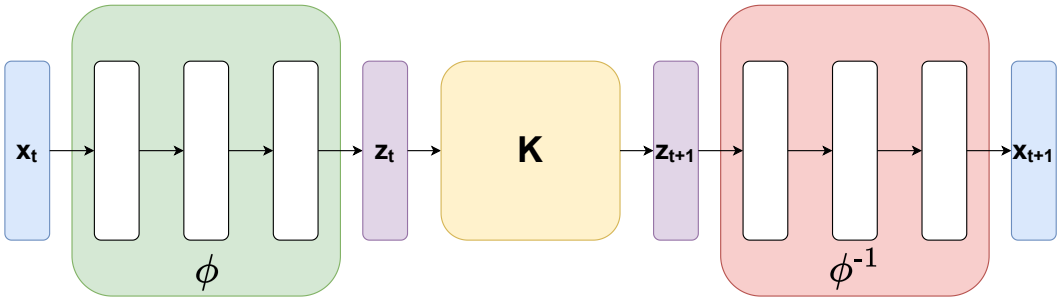

Figure 4.5 – Graphical representation of the IKAE framework. Note that, compared to the KAE framework represented in figure 2.1, the sizes of the intermediary states are always the same, in order to keep the invertibility of the encoder. Besides, ϕ^{-1} is the exact inverse of ϕ rather than a second neural network with its own trainable parameters.

vector is now exact. In practice, in addition to removing a potential source of error, this means that there is no necessity for using the reconstruction loss from equation (2.12). Thus, the global loss landscape is likely to become less accidented, and the training procedure also gets simpler in practice since there is one less training hyperparameter to tune.

The invertible encoding function that is used in [50, 49] is based on a normalizing flow with coupling layers [51, 52]. As mentioned in section 4.1.2, a disadvantage of these models is that they require the preservation of the dimension of the input state in order for the density computations to be tractable. While this is usually an issue when processing large-dimensional inputs, we also mentioned in section 2.1 that the Koopman autoencoders usually have a latent space of higher dimension than their input space. This is due to the fact that the state space often has to be augmented, rather than compressed, in order to find a suitable Koopman invariant subspace that can provide a reasonably accurate linear approximation of the dynamical system. For this reason, the preservation of the dimension might be problematic when resorting to an invertible encoder in a KAE. In order to alleviate this potential issue, it has been proposed to augment the state with zeros along new dimensions, either before [49] or in between [50] the coupling layers. This architectural change keeps the invertibility property.

In [50], the introduced IKAE is used to provide embeddings for a Transformer model [142]. The Transformer then performs predictions in the latent space of the IKAE, which are then decoded by this model. This strategy, which was first sketched in [143], was shown to obtain better results than directly forecasting with a KAE through equation (2.7), at the cost of less flexibility and an overall more complex model. The authors of [50] report better forecasting performance on chaotic dynamical systems such as the Lorenz-63 system [19] when providing embeddings for a Transformer with their IKAE model rather than with a classical KAE model, which may be explained by the increased robustness of the analytically invertible encoding of the IKAE. In [49], the authors show that their IKAE model performs better when forecasting

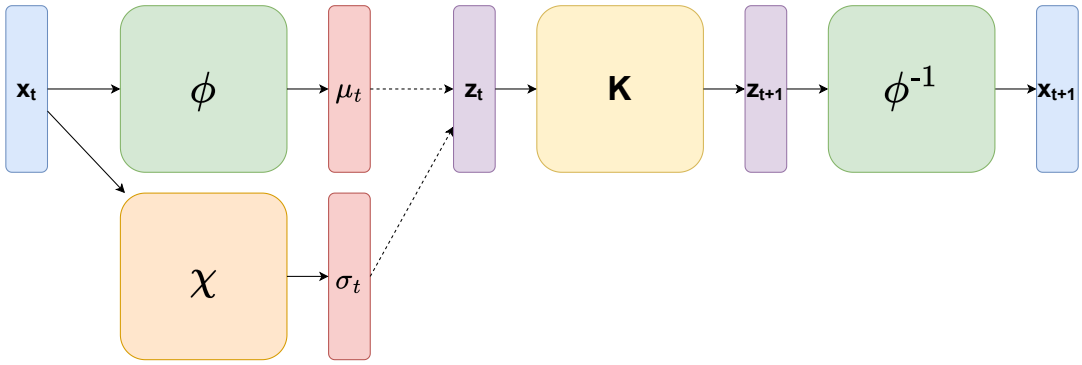

Figure 4.6 – Graphical representation of the VIKAE framework. In comparison to the IKAE from figure 4.5, a new variance encoder χ is introduced, so that the latent state $\mathbf{z}_t$ is now sampled from a Gaussian distribution rather than deterministically obtained from ϕ. Dashed lines represent a sampling operation while the solid lines represent deterministic operations.

low-dimensional dynamical systems driven by the Burgers equation or the Allen-Cahn equation than a classical KAE.

We now move on to variational invertible Koopman autoencoders (VIKAE). To the best of our knowledge, there are no published works that present such models yet. Thus, from now on until the end of the chapter, the developments that we present are novel.

The idea for VIKAE is to add a stochastic component to the model, in the form of a variational encoding. Thus, this corresponds to a similar approach to the one of VAEs: instead of providing a single encoding, the encoder returns the mean and standard deviation of the distribution of possible encodings associated to a given input. Thus, compared to the IKAE framework, a new encoder should be trained to produce a (diagonal) variance along with the invertible encoder, which now computes the mean of the distribution of encodings. Apart from this change, the other components of the Koopman autoencoder remain unchanged. Thus, to sum up, the components of a VIKAE, which we graphically represent in figure 4.6, are:

— the invertible encoder $\phi : \mathbb{R}^n \to \mathbb{R}^n$, with its analytical inverse denoted ϕ^{-1},

— the variance encoder $\chi : \mathbb{R}^n \to \mathbb{R}^n$, which computes a (diagonal) variance associated to the mean vector returned by the invertible encoder ϕ,

— the Koopman matrix $\mathbf{K}$, which is meant to be multiplied to a latent state sampled from the Gaussian distribution parameterized by the outputs of ϕ and χ.

With this architectural change, one now has a stochastic model for the evolution of the state, either in the latent space or in the observation space. In the latent space, the distribution of the state always remains Gaussian since it is, at any time, a linear transformation of the Gaussian distribution of the initial latent state. Indeed, the matrix $\mathbf{K}$ that guides the latent evolution is still a deterministic component, which means that all of the stochasticity of the predictions lies in the encoding of the initial state. We will come back to this analysis later for the AVIKAE model.

In the input space however, the distribution of the state is a nonlinear transformation (through ϕ^{-1}) of this Gaussian distribution. Thus, the distribution in the input space is rather difficult to manipulate, yet it can easily be sampled from by decoding states sampled from the latent distribution. Besides from enabling to obtain stochastic predictions, variational autoencoders, which were discussed at length in section 4.1.2, are known for having more structured latent spaces than classical autoencoders. Thus, the introduction of a variational component might also improve the structure of the analytically invertible encoders.

For the sake of completeness, we briefly mention the recent works that introduced frameworks for variational KAEs. In [144], a Koopman-inspired model is introduced for time series generation. This model comprises a variational encoder, yet it deviates from the KAE framework in the sense that its encoder and decoder contain gated recurrent units (GRU) [145], which process the information sequentially instead of encoding the state vectors independently from each other. In addition, the matrix that governs the latent evolution is re-computed for every latent trajectory as the best linear fit for this particular sequence, instead of being a trainable component of the model. The authors of [146] also propose a similar approach to our V(I)KAE where the encoder learns the parameters of a latent Gaussian distribution instead of a single encoded state. However, in their architecture, the decoding is performed by multiplicating an encoded state by an observation matrix $\mathbf{C}$ rather than by a neural decoder, which means that this approach does not fall under the KAE framework.

Finally, we now discuss the Augmented Variational Invertible Koopman AutoEncoder (AVIKAE) framework, which, to the best of our knowledge, does not correspond to any prior work. Compared to the previous VIKAE framework, our motivation for introducing this architecture is that we would like to manipulate higher-dimensional latent vectors while still benefiting from the important invertibility and density estimation properties allowed by the normalizing flows with coupling layers. In order to attain this objective, we introduce a third encoder neural network, which we call the augmentation encoder. As its name suggests, the goal of this component is to augment the invertible latent vector in order to be able to represent a higher-dimensional KIS. A sample encoding associated to a given input now has two parts: an invertible part and an augmentation part. Its distribution is parameterized by the outputs of three encoders, as follows:

- the invertible encoder $\phi : \mathbb{R}^n \to \mathbb{R}^n$, returns the mean of the invertible part of the encoding,
- the augmentation encoder $\xi : \mathbb{R}^n \to \mathbb{R}^m$, returns the mean of the augmentation part of the encoding,
- the variance encoder $\chi : \mathbb{R}^n \to \mathbb{R}^{n+m}$, returns the (diagonal) variance for both parts of the encoding.

There are two remaining components for the AVIKAE model: its advancement matrix $\mathbf{K} \in \mathbb{R}^{d \times d}$ (where $d = n + m$), and its decoder $\phi^{-1} : \mathbb{R}^n \to \mathbb{R}^n$. A schematic view of the whole architecture

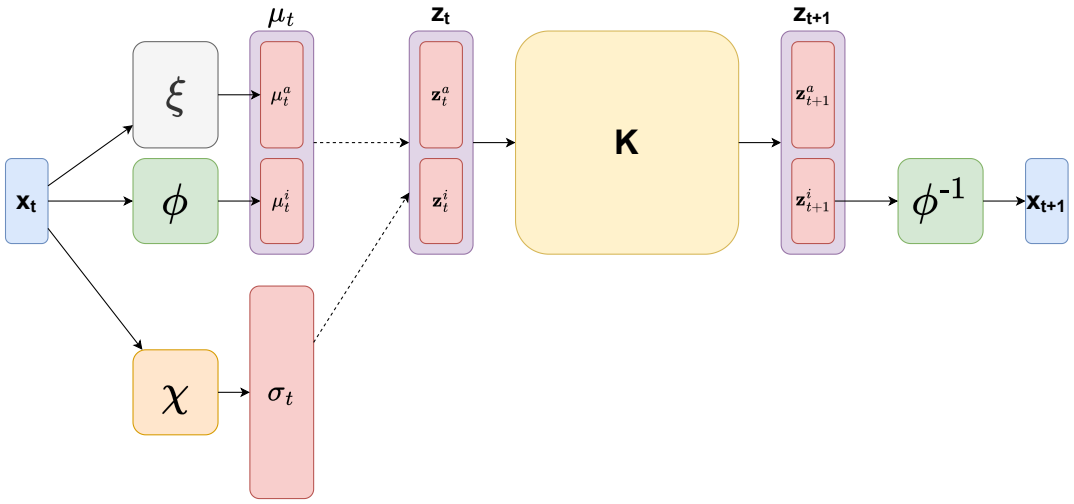

Figure 4.7 – Graphical representation of the AVIKAE framework. In comparison to the VIKAE from figure 4.5, an augmentation encoder ξ is introduced. The mean $\boldsymbol{\mu}_t$ of $\mathbf{z}_t$ is now obtained by concatenating $\boldsymbol{\mu}_t^i = \phi(\mathbf{x}_t)$ and $\boldsymbol{\mu}_t^a = \xi(\mathbf{x}_t)$, and the size of the variance vector $\boldsymbol{\sigma}_t = \chi(\mathbf{x}_t)$ has to be increased accordingly. Again, dashed lines represent a sampling operation while the solid lines represent deterministic operations.

is shown in figure 4.7. We emphasize that only ϕ has to be invertible by design, while the other encoders ξ and χ do not have this constraint and can therefore be implemented as classical neural architectures, such as MLP networks.

Importantly, ϕ^{-1} is still the analytical inverse of the invertible encoder ϕ. As such, it only takes the invertible part, rather than the full encoding, as its input. With this architectural choice, one might think that the augmentation part of the encoding plays no role in the final solution, yet it actually has an influence on the invertible part through the multiplications by $\mathbf{K}$. Thus, one might view the invertible part of the latent state as containing the "static" features of the state, i.e. the features that directly enable the reconstruction of the associated state space. In contrast, the augmentation part of the encoding can be seen as containing the "dynamic" features of the state as it has no influence on the current state vector but only on the subsequent ones in the predictions. In the following subsection, we will build a statistical model in order to enable a better understanding of the stochastic predictions that are produced by the AVIKAE architecture.

4.3.2 Statistical model associated to the AVIKAE architecture

After having discussed the neural AVIKAE architecture in the previous subsection, we now detail a statistical model that will be used along with this architecture in order to perform

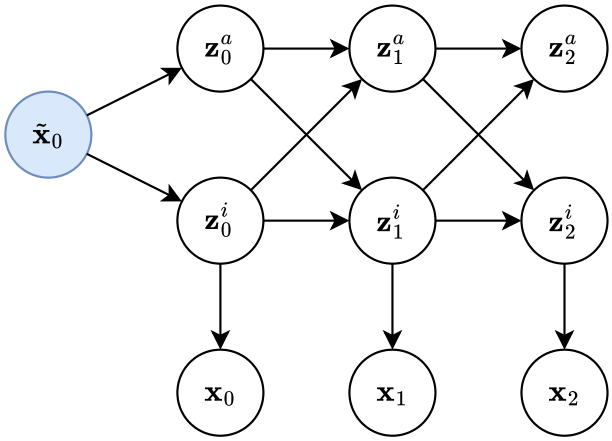

Figure 4.8 – Directed graph representing the probabilistic model for predicting the next 2 time steps from an observed initial state $\tilde{\mathbf{x}}_0$, through the AVIKAE architecture.

stochastic predictions with this model. In particular, we provide a graphical probabilistic model and we analytically describe the probabilities that can be computed from this model.

First, let us assume that the input data from which the probabilistic predictions are computed is an observation $\tilde{\mathbf{x}}_0$, which is obtained at time 0. The "tilde" denotes the fact that this observation is not necessarily assumed to be a completely trustworthy information about the actual state of the dynamical system at time 0. Indeed, $\tilde{\mathbf{x}}_0$ can be assumed to be a noisy version of the true state $\bar{\mathbf{x}}_0$, where the noise would typically follow a centered Gaussian distribution. Thus, the probability distribution of the initial state will itself be inferred by the model.

We first introduce some notations for the random variables that will be manipulated. Remember that, with the AVIKAE architecture, a latent vector is composed of an invertible part of dimension n and of an augmentation part of dimension m, which together form an encoding of dimension $d = n + m$. By convention, one can consider that for a latent vector $\mathbf{z}_t \in \mathbb{R}^d$, the elements indexed from 1 to n compose the invertible part while the elements from $n+1$ to d compose the augmentation part. We will use the notations $\mathbf{z}_t^i$ and $\mathbf{z}_t^a$ to respectively denote these two parts. Importantly, from now on the states $\mathbf{x}_t$ and latent states $\mathbf{z}_t$, as well as their two parts $\mathbf{z}_t^i$ and $\mathbf{z}_t^a$, are all considered to be random variables. We will use notations $\mathbf{x}$ and $\mathbf{z}$ to refer to particular values that these variables could take. In addition, since the variables $\mathbf{z}_t^i$ and $\mathbf{z}_t^a$ given $\tilde{\mathbf{x}}_0$ will all follow Gaussian distributions, the mean vectors and covariance matrices for $\mathbf{z}_t$, $\mathbf{z}_t^i$ and $\mathbf{z}_t^a$ given $\tilde{\mathbf{x}}_0$ will respectively be denoted as:

— $\boldsymbol{\mu}_t \in \mathbb{R}^d, \boldsymbol{\Sigma}_t \in \mathbb{R}^{d \times d}$
— $\boldsymbol{\mu}_t^i \in \mathbb{R}^n, \boldsymbol{\Sigma}_t^i \in \mathbb{R}^{n \times n}$
— $\boldsymbol{\mu}_t^a \in \mathbb{R}^m, \boldsymbol{\Sigma}_t^a \in \mathbb{R}^{m \times m}$

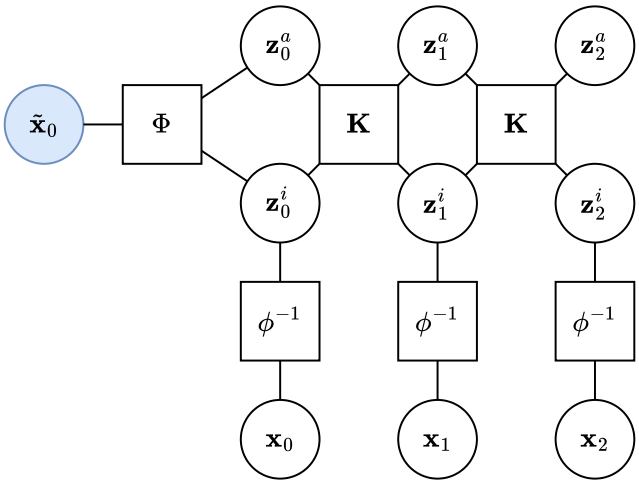

Figure 4.9 – Factor graph representing the probabilistic model for predicting the next 2 time steps from an observed initial state $\tilde{\mathbf{x}}_0$, through the AVIKAE architecture. Circles denote random variables while squares denote relationships between these variables. We use the symbol Φ to denote the sampling operations involving ϕ, χ and ξ.

On figure 4.8, we show a directed graph representing our graphical model. We refer to the chapter 8 of [106] for a review of probabilistic graphical models and the associated mathematical manipulations. As can be seen from this graph, the information flows from $\tilde{\mathbf{x}}_0$ to $\mathbf{z}_0$ and then autoregressively to the subsequent latent vectors $\mathbf{z}_t$. A corresponding factor graph, with a more explicit account of the relationships between the variables, is represented in figure 4.9.

Let us first discuss the probability distribution for the initial latent state $\mathbf{z}_0$ given an observation $\tilde{\mathbf{x}}_0$ at time 0. This distribution is a Gaussian with a diagonal covariance matrix, whose $2d$ parameters are obtained from the 3 encoders ϕ, χ, ξ described in subsection 4.3.1, as follows:

$$p(\mathbf{z}_0|\tilde{\mathbf{x}}_0) \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \tag{4.38}$$

where $\boldsymbol{\mu}_0 = (\phi(\tilde{\mathbf{x}}_0), \xi(\tilde{\mathbf{x}}_0))^\intercal$ and $\boldsymbol{\Sigma}_0 = \mathrm{diag}(\chi(\tilde{\mathbf{x}}_0))$. One can also write corresponding equations for the two parts $\mathbf{z}_0^i$ and $\mathbf{z}_0^a$:

$$\begin{aligned} p(\mathbf{z}_0^i|\tilde{\mathbf{x}}_0) &\sim \mathcal{N}(\boldsymbol{\mu}_0^i, \boldsymbol{\Sigma}_0^i) \\ p(\mathbf{z}_0^a|\tilde{\mathbf{x}}_0) &\sim \mathcal{N}(\boldsymbol{\mu}_0^a, \boldsymbol{\Sigma}_0^a). \end{aligned} \tag{4.39}$$

where the Gaussian distributions are parameterized by

— $\boldsymbol{\mu}_0^i = \phi(\tilde{\mathbf{x}}_0)$,
— $\boldsymbol{\Sigma}_0^i = \mathrm{diag}((\chi(\tilde{\mathbf{x}}_0))_{1:n})$
— $\boldsymbol{\mu}_0^a = \xi(\tilde{\mathbf{x}}_0)$

— $\mathbf{\Sigma}_0^a = \text{diag}((\chi(\tilde{\mathbf{x}}_0))_{n+1:d})$

Note that, although $\mathbf{z}_0^i$ and $\mathbf{z}_0^a$ seem to be jointly Gaussian when looking at equation (4.38), they are actually independent from each other since $\mathbf{\Sigma}_0$ is diagonal. Now, one can obtain an estimated probability distribution for the actual initial state $\mathbf{x}_0$ given $\tilde{\mathbf{x}}_0$ by leveraging the bijection ϕ between the state space and the invertible part of the latent space, which is expressed by the following deterministic relationship:

$$\mathbf{x}_0 = \phi^{-1}(\mathbf{z}_0^i). \tag{4.40}$$

Thus, the probability distribution of $\mathbf{x}_0$ to take a value $\mathbf{x} \in \mathbb{R}^n$ knowing $\mathbf{z}_0^i$ is simply a Dirac distribution on $\phi^{-1}(\mathbf{z}_0^i)$, located on the value $\mathbf{x}$:

$$p(\mathbf{x}_0 = \mathbf{x}|\mathbf{z}_0^i) = \delta_{\mathbf{x}}(\phi^{-1}(\mathbf{z}_0^i)). \tag{4.41}$$

As a quick reminder, $\delta_{\mathbf{x}}(.)$ is a function that has an integral of 1 and whose mass is located on the value $\mathbf{x}$. Besides, one can see from the graphical model of figure 4.8 that $\mathbf{x}_0$ is conditionally independent on any other variable of the graph given $\mathbf{z}_0^i$. Concretely, this means that the probability distribution of $\mathbf{x}_0$ given the values of $\mathbf{z}_0^i$ and of any other variables of the graph is the same as the probability distribution of $\mathbf{x}_0$ given $\mathbf{z}_0^i$ only. More generally, for any time t, $\mathbf{x}_t$ is conditionally independent on any other variable of the graph given $\mathbf{z}_t^i$. Thus, one can now compute

$$\begin{aligned} p(\mathbf{x}_0|\tilde{\mathbf{x}}_0) &= \int_{\mathbb{R}^n} p(\mathbf{x}_0, \mathbf{z}_0^i|\tilde{\mathbf{x}}_0) d\mathbf{z}_0^i \\ &= \int_{\mathbb{R}^n} p(\mathbf{z}_0^i|\tilde{\mathbf{x}}_0) p(\mathbf{x}_0|\mathbf{z}_0^i, \tilde{\mathbf{x}}_0) d\mathbf{z}_0^i \\ &= \int_{\mathbb{R}^n} p(\mathbf{z}_0^i|\tilde{\mathbf{x}}_0) p(\mathbf{x}_0|\mathbf{z}_0^i) d\mathbf{z}_0^i \\ &= \int_{\mathbb{R}^n} p(\mathbf{z}_0^i|\tilde{\mathbf{x}}_0) \delta_{\mathbf{x}_0}(\phi^{-1}(\mathbf{z}_0^i)) d\mathbf{z}_0^i. \end{aligned} \tag{4.42}$$

We first made use of the sum rule, then the product rule (see e.g. [106] equations 1.10 and 1.11), then of the conditional independence of $\mathbf{x}_0$ on $\tilde{\mathbf{x}}_0$ given $\mathbf{z}_0$ and finally of equation (4.41). In particular, for a given value $\mathbf{x}$, by writing $p(\mathbf{z}_0|\tilde{\mathbf{x}}_0)$ more explicitly through equation (4.39), we obtain

$$p(\mathbf{x}_0 = \mathbf{x}|\tilde{\mathbf{x}}_0) = \int_{\mathbb{R}^n} \mathcal{N}(\mathbf{z}_0^i; \boldsymbol{\mu}_0^i, \mathbf{\Sigma}_0^i) \delta_{\mathbf{x}}(\phi^{-1}(\mathbf{z}_0^i)) d\mathbf{z}_0^i. \tag{4.43}$$

Finally, we make the change of variables $\mathbf{x}_0 = \phi^{-1}(\mathbf{z}_0^i)$, or equivalently $\mathbf{z}_0^i = \phi(\mathbf{x}_0)$. The corresponding infinitesimal variation is $d\mathbf{x}_0 = \left|\det \frac{\partial \phi(\mathbf{x}_0)}{\partial \mathbf{x}_0}\right| d\mathbf{z}_0^i$, which leads to integrating on $\mathbf{x}_0$

as follows:

$$\begin{aligned} p(\mathbf{x}_0 = \mathbf{x}|\tilde{\mathbf{x}}_0) &= \int_{\mathbb{R}^n} \mathcal{N}(\mathbf{z}_0^i; \boldsymbol{\mu}_0^i, \boldsymbol{\Sigma}_0^i)\delta_{\mathbf{x}}(\phi^{-1}(\mathbf{z}_0^i))d\mathbf{z}_0^i \\ &= \int_{\mathbb{R}^n} \mathcal{N}(\phi(\mathbf{x}_0); \boldsymbol{\mu}_0^i, \boldsymbol{\Sigma}_0^i)\delta_{\mathbf{x}}(\mathbf{x}_0) \left|\det \frac{\partial \phi(\mathbf{x}_0)}{\partial \mathbf{x}_0}\right| d\mathbf{x}_0 \\ &= \mathcal{N}(\phi(\mathbf{x}); \boldsymbol{\mu}_0^i, \boldsymbol{\Sigma}_0^i) \left|\det \frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}}\right| \end{aligned} \tag{4.44}$$

Note that we have resorted to a change of variable for a multiple integral, which is rather unusual. We refer to, e.g., the theorem 10.9 of [147] for a theoretical justification.

This final expression of equation (4.44) is easy to compute for a particular value $\mathbf{x}$ of $\mathbf{x}_0$ since the parameters $\boldsymbol{\mu}_0^i, \boldsymbol{\Sigma}_0^i$ of the Gaussian distribution are directly obtained from equation (4.39) while the Jacobian $\left|\det \frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}}\right|$ can be efficiently computed from the parameters of the normalizing flow ϕ. Thus, we are able to compute the likelihood of a particular initial value given an observed initial state. This ability can be used for training a model or for interpreting its predictions.

However, the distribution $p(\mathbf{x}_0|\tilde{\mathbf{x}}_0)$ is a nonlinear transformation of this Gaussian distribution, for which one cannot give an analytical expression, as is usually the case for distributions parameterized by a normalizing flow (or by another neural generative model).

Let us now examine the distribution of the states $\mathbf{z}_t$ and $\mathbf{x}_t$ for subsequent time indexes $t > 0$. For any time $t > 0$, the relationship between $\mathbf{z}_t$ and $\mathbf{z}_{t-1}$ is the matrix-vector product by $\mathbf{K}$:

$$\mathbf{z}_t = \mathbf{K}\mathbf{z}_{t-1}. \tag{4.45}$$

As a direct consequence of this, we have that $\mathbf{z}_t$ can be deterministically obtained from $\mathbf{z}_0$ through:

$$\mathbf{z}_t = \mathbf{K}^t\mathbf{z}_0. \tag{4.46}$$

This may seem surprising to a reader familiar with autoregressive stochastic models such as the Kalman filter or dynamical variational autoencoders [148]. Indeed, in such systems, the probability distribution $p(\mathbf{z}_t|\mathbf{z}_{t-1})$ is generally a nontrivial distribution such as a Gaussian, while it is in our case a Dirac (deterministic) distribution, which does not seem to allow for the propagation of the uncertainty in the latent space. However, it is important to remember that in the AVIKAE architecture, the inferred state vector is obtained from only the invertible part $\mathbf{z}_t^i$ rather than from the whole latent vector $\mathbf{z}_t$. For this reason, one can see that the probability distribution $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$ is actually nontrivial since (informally) the knowledge of $\mathbf{x}_t$ gives us information on $\mathbf{z}_t^i$ but not directly on $\mathbf{z}_t^a$, which also has influence on the value of $\mathbf{x}_{t+1}$ through $\mathbf{z}_{t+1}^i$. Still, the AVIKAE architecture is designed in such a way that all of the stochasticity of the predictions comes from the encoding of the initial latent state $\mathbf{z}_0$ given an observation $\tilde{\mathbf{x}}_0$, which limits the expressive power of the model. Although we will show in the associated experiments

that the model still enables to produce convincing probabilistic forecasts, the natural perspectives of this work include changing the deterministic latent dynamic evolution to a stochastic one, i.e. substituting equation (4.45) by a stochastic variant such as [5]:

$$\mathbf{z}_t = \mathbf{K}\mathbf{z}_{t-1} + \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}^2). \tag{4.47}$$

For now, we put these considerations aside and stick to the deterministic latent evolution from equation (4.45). In this context, we have seen that the distribution $p(\mathbf{z}_t|\mathbf{z}_0 = \mathbf{z})$ (for any possible value $\mathbf{z}$) is a trivial Dirac distribution. As for the distribution of $p(\mathbf{z}_t|\tilde{\mathbf{x}}_0)$, one can see from equation (4.46) that it is a linear transformation of the Gaussian distribution $p(\mathbf{z}_0|\tilde{\mathbf{x}}_0)$. It is a well known fact that such a transformation results in a new Gaussian distribution. More precisely, for a given p-dimensional random variable $\mathbf{X}$ following the Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, we have that, for any deterministic matrix $\mathbf{A} \in \mathbb{R}^{p\times p}$ and vector $\mathbf{b} \in \mathbb{R}^p$,

$$\mathbf{A}\mathbf{X} + \mathbf{b} \sim \mathcal{N}(\mathbf{b} + \mathbf{A}\boldsymbol{\mu}, \mathbf{A}\mathbf{\Sigma}\mathbf{A}^\intercal). \tag{4.48}$$

In order to avoid potential confusions, we emphasize that $.^\intercal$ denotes the transpose of a matrix, and not a matrix power. Using this formula along with the known parameters $\boldsymbol{\mu}_0$ and $\mathbf{\Sigma}_0$ of the distribution $p(\mathbf{z}_0|\tilde{\mathbf{x}}_0)$, one can then say that

$$p(\mathbf{z}_t|\tilde{\mathbf{x}}_0) \sim \mathcal{N}(\mathbf{K}^t\boldsymbol{\mu}_0, \mathbf{K}^t\mathbf{\Sigma}_0(\mathbf{K}^\intercal)^t). \tag{4.49}$$

Thus, this distribution can be easily sampled from in practice for any time step t. As mentioned earlier, we now denote $\boldsymbol{\mu}_t = \mathbf{K}^t\boldsymbol{\mu}_0$ and $\mathbf{\Sigma}_t = \mathbf{K}^t\mathbf{\Sigma}_0(\mathbf{K}^\intercal)^t$ the mean and variance of $p(\mathbf{z}_t|\tilde{\mathbf{x}}_0)$. The distribution of $p(\mathbf{z}_t^i|\tilde{\mathbf{x}}_0)$ is obtained by marginalizing this Gaussian distribution over its n first variables, which gives

$$p(\mathbf{z}_t^i|\tilde{\mathbf{x}}_0) \sim \mathcal{N}(\boldsymbol{\mu}_t^i, \mathbf{\Sigma}_t^i), \tag{4.50}$$

where $\boldsymbol{\mu}_t^i$ and $\mathbf{\Sigma}_t^i$ are respectively the first n components of $\boldsymbol{\mu}_t$ and the $n \times n$ upper left block of $\mathbf{\Sigma}_t$.

In addition, with a similar development to the one of equation (4.42) [6], one can obtain that, for a potential value $\mathbf{x} \in \mathbb{R}^n$,

$$p(\mathbf{x}_t = \mathbf{x}|\tilde{\mathbf{x}}_0) = \mathcal{N}(\phi(\mathbf{x}); \boldsymbol{\mu}_t^i, \mathbf{\Sigma}_t^i)) \left|\det \frac{\partial\phi(\mathbf{x})}{\partial\mathbf{x}}\right|. \tag{4.51}$$

5. The proposed equation corresponds to an additive noise. Another potentially interesting possibility would be to consider a multiplicative noise instead, i.e. a version where one would left-multiply the latent state by a stochastic, rather than fixed, matrix $\mathbf{K}$.

6. In short, one can start from $p(\mathbf{x}_t|\tilde{\mathbf{x}}_0) = \int_{\mathbb{R}^n} p(\mathbf{x}_t, \mathbf{z}_t^i|\tilde{\mathbf{x}}_0)d\mathbf{z}_t^i$ and from there follow the exact same steps as in equation (4.42), with $\mathbf{z}_t$ playing a role analogous to $\mathbf{z}_0$.

Again, this expression is easy to compute in practice using the known values of $\boldsymbol{\mu}_t^i$ and $\boldsymbol{\Sigma}_t^i$ along with the relatively simple Jacobian of ϕ.

To sum up, we have shown that one can compute the likelihood of obtaining a given (latent or observed) state at any time $t \geq 0$ when conditioning our model on an initial observation $\tilde{\mathbf{x}}_0$. This capacity can be used at testing time in order to perform density estimation, but also at training time. Indeed, when training on a time series $(\bar{\mathbf{x}}_t)_{0 \leq t \leq T}$ where $\bar{\mathbf{x}}_t \in \mathbb{R}^n$ is considered to be a groundtruth observed state for time t, one can let $\tilde{\mathbf{x}}_0 = \bar{\mathbf{x}}_0$ be the input to the trained AVIKAE model. Then, one can seek the parameters of ϕ, χ, ξ and $\mathbf{K}$ that maximise the likelihood of the data, i.e.

$$\arg\max_{\phi,\chi,\xi,\mathbf{K}} \quad \sum_{t=0}^{T} p(\mathbf{x}_t = \bar{\mathbf{x}}_t | \tilde{\mathbf{x}}_0). \tag{4.52}$$

In practice, one can solve this problem by using the negative log-likelihood of the data as a loss function for the model.

Importantly, this approach only consists in maximizing a compromise between the individual likelihoods of each of the observations at time t. It would be more interesting but more complex to instead maximize the likelihood of the trajectory as a whole, i.e. substitute the problem of equation (4.52) by

$$\arg\max_{\phi,\chi,\xi,\mathbf{K}} \quad p((\mathbf{x}_0, ..., \mathbf{x}_T) = (\bar{\mathbf{x}}_0, ..., \bar{\mathbf{x}}_T) | \tilde{\mathbf{x}}_0). \tag{4.53}$$

Should the variables $\mathbf{x}_0, ..., \mathbf{x}_T$ be independent from each other given $\tilde{\mathbf{x}}_0$, the two problems would be equivalent since the joint likelihood of the trajectory would be equal to the product of the marginal probabilities of each state. However, this is not true in general since the knowledge that the variable $\mathbf{x}_t$ takes the value $\bar{\mathbf{x}}_t$ influences the probability distribution of $\mathbf{x}_{t+1}$ (and of all subsequent states). Formally, we have that

$$p(\mathbf{x}_{t+1} | \mathbf{x}_t, \tilde{\mathbf{x}}_0) \neq p(\mathbf{x}_{t+1} | \tilde{\mathbf{x}}_0). \tag{4.54}$$

This can be concretely illustrated by examining the probability distribution $p(\mathbf{z}_1 | \mathbf{x}_0, \tilde{\mathbf{x}}_0)$. For this purpose, we introduce the following block decomposition of $\mathbf{K}$:

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22}, \end{pmatrix} \tag{4.55}$$

with $\mathbf{K}_{11} \in \mathbb{R}^{n \times n}$, $\mathbf{K}_{12} \in \mathbb{R}^{m \times n}$, $\mathbf{K}_{21} \in \mathbb{R}^{n \times m}$, $\mathbf{K}_{22} \in \mathbb{R}^{m \times m}$. With these notations we have as a direct consequence from equation (4.45) that, for any $t \geq 0$,

$$\begin{aligned} \mathbf{z}_{t+1}^i &= \mathbf{K}_{11} \mathbf{z}_t^i + \mathbf{K}_{12} \mathbf{z}_t^a \\ \mathbf{z}_{t+1}^a &= \mathbf{K}_{21} \mathbf{z}_t^i + \mathbf{K}_{22} \mathbf{z}_t^a. \end{aligned} \tag{4.56}$$

One can then write

$$\begin{aligned} p(\mathbf{z}_1^i|\mathbf{x}_0, \tilde{\mathbf{x}}_0) &= p(\mathbf{K}_{11}\mathbf{z}_0^i + \mathbf{K}_{12}\mathbf{z}_0^a|\mathbf{x}_0, \tilde{\mathbf{x}}_0) \\ &= p(\mathbf{K}_{11}\phi(\mathbf{x}_0) + \mathbf{K}_{12}\mathbf{z}_0^a|\mathbf{x}_0, \tilde{\mathbf{x}}_0). \end{aligned} \tag{4.57}$$

Since $\mathbf{x}_0$ is known, the term $\mathbf{K}_{11}\phi(\mathbf{x}_0)$ in the conditional probability is fixed and acts as a bias term. Besides, using the conditional independence of $\mathbf{z}_0^a$ on $\mathbf{x}_0$ given $\tilde{\mathbf{x}}_0$, we have that

$$p(\mathbf{z}_0^a|\mathbf{x}_0, \tilde{\mathbf{x}}_0) = p(\mathbf{z}_0^a|\tilde{\mathbf{x}}_0) \sim \mathcal{N}(\boldsymbol{\mu}_0^a, \boldsymbol{\Sigma}_0^a). \tag{4.58}$$

Thus, using equation (4.48), we obtain

$$p(\mathbf{z}_1^i|\mathbf{x}_0, \tilde{\mathbf{x}}_0) \sim \mathcal{N}(\mathbf{K}_{12}\boldsymbol{\mu}_0^a + \mathbf{K}_{11}\phi(\mathbf{x}_0), \mathbf{K}_{12}\boldsymbol{\Sigma}_0^a\mathbf{K}_{12}^\intercal). \tag{4.59}$$

In contrast, when $\mathbf{z}_1^i$ is conditioned only on $\tilde{\mathbf{x}}_0$, one can marginalize the Gaussian distribution of equation (4.49) on its first n variables to obtain

$$p(\mathbf{z}_1^i|\tilde{\mathbf{x}}_0) \sim \mathcal{N}(\mathbf{K}_{12}\boldsymbol{\mu}_0^a + \mathbf{K}_{11}\boldsymbol{\mu}_0^i, \mathbf{K}_{12}\boldsymbol{\Sigma}_0^a\mathbf{K}_{12}^\intercal + \mathbf{K}_{11}\boldsymbol{\Sigma}_0^i\mathbf{K}_{11}^\intercal). \tag{4.60}$$

By comparing equations (4.59) and (4.60), one can intuitively see that the additional conditioning on $\mathbf{x}_0$ results in losing the variance that comes from $\mathbf{z}_0$ and in a potential bias when $\phi(\mathbf{x}_0)$ is not equal to its mean $\boldsymbol{\mu}_0^i$ given $\tilde{\mathbf{x}}_0$.

Thus, we have proved that the likelihood $p(\mathbf{x}_0, ..., \mathbf{x}_T|\tilde{\mathbf{x}}_0)$ of a predicted trajectory is different from the product of the marginal likelihoods of each point in the trajectory. Although we do not detail the computations, we conjecture that the likelihood of almost every trajectory $(\bar{\mathbf{x}}_0, ..., \bar{\mathbf{x}}_T)$ given $\tilde{\mathbf{x}}_0$ will be exactly 0 for a sufficiently large trajectory length $T+1$, which means that the model is unable to produce a prediction that corresponds to these trajectories. This can be intuited by trying to solve the problem from a linear algebra perspective, i.e. characterize the set of initial latent conditions $\mathbf{z}_0$ that, when propagated through equation (4.46), verify

$$\begin{cases} \mathbf{z}_0^i = \phi(\bar{\mathbf{x}}_0) \\ \mathbf{z}_1^i = \phi(\bar{\mathbf{x}}_1) \\ \vdots \\ \mathbf{z}_T^i = \phi(\bar{\mathbf{x}}_T) \end{cases} \tag{4.61}$$

Remember that, from equation (4.46), the variables $\mathbf{z}_1^i, ..., \mathbf{z}_T^i$ are all related linearly to $\mathbf{z}_0$. Besides, each of the above equations is a vector equation, which is in fact equivalent to a system of n scalar equations. Thus, supposing that all of these equations are linearly independent, we find ourselves with a system of $n(T+1)$ linear equations on the d variables of $\mathbf{z}_0$. This system can

only admit solutions if $d \geq n(T+1)$. Remember that d is the sum of the sizes of the invertible encoding and of the augmentation encoding, i.e. $d = n + m$. Therefore, one can see that, for any chosen size m of the augmentation encoding, there is a strict limit to the trajectory length up to which the model will be able to produce any possible trajectory (i.e. any trajectory is associated to a nonzero probability). When this limit is exceeded, only some specific trajectories can still be produced (i.e. almost all trajectories have a likelihood of 0).

To sum up, the preceding discussion suggests that the AVIKAE model is not able to provide nonzero trajectory likelihoods for almost all of potential trajectories starting from a length that depends on the size of the augmentation part of the encoding. Again, this advocates for introducing some noise in the latent dynamics, e.g. through equation (4.47). Indeed, by introducing stochasticity at each step of the latent dynamics, one would significantly improve the model's generative abilities, and therefore be able to provide a nonzero likelihood for any trajectory of any length. We keep this as a perspective for future works, and we now move on to reporting experiments performed with the current AVIKAE model driven by equation (4.46).

4.3.3 Experiments with the AVIKAE architecture on Sentinel-2 time series

We now illustrate the capabilities offered by the AVIKAE model on the Sentinel-2 time series that were introduced in chapter 1 and used in the experiments of chapters 2 and 3. As in these previous experiments, we train a model on a set of data from the forest of Fontainebleau, and then test this model on both the Fontainebleau and the Orléans areas. Thus, on the Fontainebleau area, we primarily evaluate the capacity of the model to extrapolate in time, while on the Orléans area the ability of the model to transfer its knowledge to a different spatial area is also examined.

We adopt the experimental setup of section 2.2.2.b where the training is performed on an incomplete version of the Fontainebleau time series. The advantage of this setup is that one does not have to interpolate the available data, which means that our model only learns from the true distribution of the collected data rather than learning a chosen (imperfect) interpolation scheme along with the true data. As discussed in section 2.2.2.b, the KAE framework enables to deal with irregularly-sampled time series with relatively little effort, and this possibility carries on to the more specific AVIKAE model.

In order to leverage the capability of the model to perform stochastic predictions, one should adapt the loss function that we introduced in section 2.2. First, as previously mentioned, since the decoder in an AVIKAE model is the analytical inverse of the encoder ϕ, the reconstructions through the autoencoder are now exact, which enables to remove the reconstruction loss term introduced in equation (2.12). The orthogonality loss term from equation (2.34) is not affected by the change of the architecture, and it therefore remains unchanged in comparison to the KAE architecture from section 2.2. As for the prediction and the linearity loss terms from equations (4.17) and (4.19), they have to be slightly adapted to the new architecture. Indeed, as

a reminder, in a classical KAE model, the predictions are obtained deterministically from a given initial condition $\tilde{\mathbf{x}}_0$ by equation (2.7). In contrast, for the AVIKAE model (or more generally for any variational KAE) the initial latent state $\mathbf{z}_0$ is a random variable from which the parameters are given by the output of the model's encoders when fed with $\tilde{\mathbf{x}}_0$. In practice, although one wants to allow for an appropriate amount of variability in the associated predictions, we still want any sample prediction to provide a reasonably good approximation of the "groundtruth" subsequent states. For this reason, instead of computing the prediction and linearity loss terms on the only (or, in the case of a stochastic model, the mean) prediction, we first sample latent conditions in $p(\mathbf{z}_0|\tilde{\mathbf{x}}_0)$, and then produce associated predictions on which the prediction and linearity loss terms are computed. In order to be able to differentiate these terms, one uses the well-known reparameterization trick that was introduced by [102].

Without the introduction of a new loss term, this loss function corresponds to a (regularized) form of empirical risk minimization, and a model trained in such a way would tend to only capture the mean distribution of $p(\mathbf{x}_t|\tilde{\mathbf{x}}_0)$. Thus, the variance encoder χ would likely tend to produce very low variance vectors because it has no particular interest in introducing variability in the predictions. In practice, this would mean that the predictions would be highly overconfident or even that the model would tend to perform deterministic predictions from a practical point of view. This problem is similar to the one that occured for ensembles of Koopman models in section 4.2, where the members tended to all produce similar predictions, and we had to introduce variance-promoting loss terms to make the ensembles less confident. We could use a similar strategy here, yet since we have access to the PDF of the predictions, one can instead directly maximise the likelihood of the data. Therefore, we introduce a new loss term which is linked to the likelihood of the observed data given the probability distributions produced by the model. This loss term corresponds to a natural choice for training generative models such as normalizing flows.

As explained in subsection 4.3.2, with the AVIKAE model, the predictions in the latent state at time t given an observed initial condition $\tilde{\mathbf{x}}_0$ always follow a Gaussian distribution, i.e. $p(\mathbf{z}_t|\tilde{\mathbf{x}}_0) \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. In addition, given an initial observation, one can easily compute the likelihood that the stochastic prediction $\mathbf{x}_t$ produced by the model at time t takes a given value $\mathbf{x}$. Thus, we are able to produce - in a sense - a probabilistic version of the prediction loss term, where instead of computing the difference between a sample prediction and the groundtruth we evaluate the probability of obtaining the groundtruth according to the distribution of the predictions. Concretely, let us assume (as we did in section 2.1.3) that the training dataset consists of N regularly-sampled trajectories of length $T+1$, which are denoted as $(\tilde{\mathbf{x}}_{i,t})_{1\leq i\leq N, 0\leq t\leq T}$. We denote as θ the parameters of the AVIKAE model, i.e. the parameters of the components ϕ, χ, ξ

and the coefficients of $\mathbf{K}$. Then, the new likelihood loss term L_{lkl} can be written as

$$L_{lkl}(\theta) = \sum_{i=1}^{N} \sum_{t=0}^{T} -\log p(\mathbf{x}_{i,t} = \tilde{\mathbf{x}}_{i,t} | \tilde{\mathbf{x}}_{i,0}). \tag{4.62}$$

This expression can be computed through equation (4.51). For a given sample corresponding to index i and time t, it is an addition of two components: the Gaussian negative log-likelihood of the latent state $\phi(\tilde{\mathbf{x}}_{i,t})$ and the negative log-Jacobian of the encoder ϕ evaluated on $\tilde{\mathbf{x}}_{i,t}$. For the first part, the parameters of the Gaussian are obtained from $\phi(\tilde{\mathbf{x}}_0)$, $\xi(\tilde{\mathbf{x}}_0)$ $\chi(\tilde{\mathbf{x}}_0)$, and $\mathbf{K}$, hence they are indeed differentiable according to θ. For the second part, the Jacobian of ϕ can be easily computed since ϕ is a normalizing flow. Thus, the likelihood term of equation (4.62) can indeed be computed using an automatic differentiation framework. Interestingly, if the Jacobian term is not included in the loss term, then the loss can be understood as the likelihood of the associated latent state. Thus, one can roughly see it as a probabilistic version of the linearity loss term from equation (2.13).

In fact, the likelihood term could theoretically be the only term in the loss function for training a stochastic model. However, in our experiments, we did not manage to make a model converge using only the likelihood loss from equation (4.62). We conjecture that this might be due to a badly behaved gradient, which notably contains the determinants of covariance matrices. However, we will show that, when used as an auxiliary loss term, L_{lkl} enables to significantly improve the soundness of the probabilistic forecasts, both visually and quantitatively. To sum up, our loss function for the AVIKAE model is the following:

$$L_{AVIKAE}(\theta) = L_{pred}(\theta) + \alpha L_{lin}(\theta) + \beta L_{lkl}(\theta) + \gamma L_{orth}(\theta). \tag{4.63}$$

When comparing to the loss function from equation (2.14) that we used in chapter 2, one can see that the auto-encoding loss from equation (2.12) was removed while the likelihood loss from equation (4.62) was added. Thus, the loss function still contains 4 terms, with the prediction loss term being considered as the main one and the parameters α, β, γ used to determine their relative weights in the expression of the loss.

For this experiment, as in previous experiments on Sentinel-2 time series, we train our models on the Fontainebleau area, on a temporal span of 1440 days. We place ourselves in a harder setup than in section 2.2.2.b in the sense that no temporal pre-processing is applied on the training data, which are irregularly sampled in time. In addition, the area of the training data is extended: while we trained our models on an area of 150×150 pixels (i.e. 2.25 square kilometers) in the experiments of chapters 2 and 3, the models are now trained on an area of 300×300 pixels (i.e. 9 square kilometers). We will see in the quantitative results that this significantly larger training dataset is, as expected, harder to fit, but enables to obtain better generalisation performance

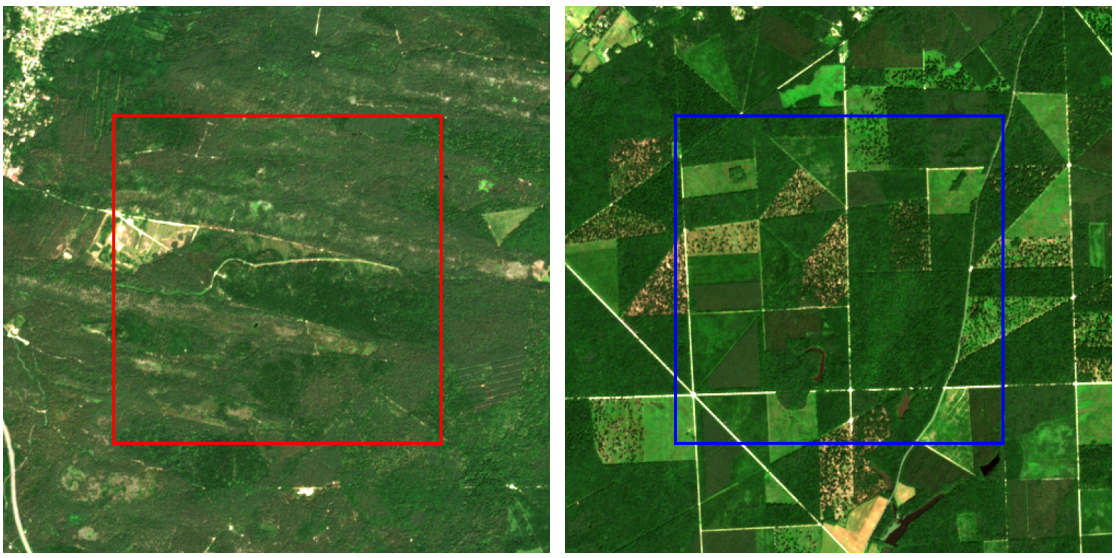

Figure 4.10 – Left: an image from the forest of Fontainebleau. Right: an image from the forest of Orléans. The date for both images is 20/06/2018. Those are RGB compositions with saturated colors. The red square is the 300×300 pixel training area and the blue square is the 300×300 pixel test area.

on the forest of Orléans. The testing Orléans area is also extended from 150×150 to 300×300 pixels. In figure 4.10, we show sample images of the datasets and highlight the areas that we use in this section for training and testing the models. Interestingly, it is in large part thanks to the training on irregular (rather than interpolated) data that training on this larger scale is made possible. Indeed, the computational bottleneck of training a Koopman autoencoder on long-term forecasting is the decoder since, after cheaply multiplying by $\mathbf{K}$ to predict in the latent space, all latent vectors have to be decoded in order to perform a comparison to the groundtruth. When training on sparsely sampled data, we only decode the vectors for which a groundtruth point is available, which significantly reduces the computational burden.

In order to assess the interest of the different components of the AVIKAE model for probabilistic forecasting, we will train several variants of Koopman autoencoder models:

— The complete AVIKAE model, trained with the loss function from equation (4.63),

— A (non-augmented) variational invertible Koopman autoencoder, with a loss function similar to the AVIKAE model since this model also does not require a reconstruction loss term and enables to compute a likelihood loss term. This model will be referred to as "VIKAE",

— A variational Koopman autoencoder model, with a non-invertible encoder. Its loss function includes a prediction term, a reconstruction term, a linearity term and a likelihood term on the latent state. Here, since the autoencoder is not a normalizing flow model, the

Jacobian of the decoder cannot be easily computed, hence it is simply ignored, which indeed amounts to only computing the likelihood of the latent state. This model will be referred to as "VKAE",

— An ensemble of 16 classical Koopman autoencoders, which we refer to as "KAE ensemble". The ensemble is trained with the loss function from equation (4.25), with the value $\lambda = 0.5$ which was observed to perform well on both the Fontainebleau and Orléans areas in section 4.2.

Table 4.1 – Forecasting CRPS for different methods and areas

	CRPS on the Fontainebleau area	CRPS on the Orléans area
AVIKAE	**0.0241**	**0.0473**
VIKAE	0.0304	0.0567
VKAE	0.0276	0.0540
KAE ensemble	0.0258	0.0530

Like in section 4.2, our primary metric for quantitatively evaluating the models will be the CRPS on the test data. In table 4.1, we report the results of all of our models trained on the Fontainebleau area and tested for extrapolation on the same Fontainebleau area as well as for transferring on the Orléans area. From this table, one can see that the AVIKAE model performs best on both datasets. In particular, it is the only tested method which is able to beat the ensemble of Koopman autoencoders presented in section 4.2, although it has far fewer parameters. In addition, we provide visual results for the predictions made by the AVIKAE model on the Fontainebleau area in figure 4.11. In this figure, for each randomly chosen pixel, in addition to the central prediction of the model (i.e. $\mathbf{z}_0 = \boldsymbol{\mu}_0$), we sample 100 trajectories in the distribution of predictions conditioned on the initial observed datapoint. One can see that, although the central prediction of the model is usually imperfect, the observed datapoints most often appear to be likely with regards to the empirical distribution of the predictions. In addition, this figure illustrates the complexity of the output probability distribution of the model, and in particular the possibility that it brings to sample as many trajectories as desired. In contrast, the ensembles of deterministic models are always limited to producing a fixed number of trajectories, as illustrated in figure 4.2. Thus, the probability distributions produced by the AVIKAE model are significantly more complex, which is usually a desirable feature for generative models.

Interestingly, one can see that the VIKAE is the worst performing of the tested methods, having significantly worse results than the non-invertible VKAE. We conjecture that this bad performance is due to the limitation of the size of the latent space that is imposed by the invertibility property. Indeed, the size of the latent space of the VIKAE model is only $d = 10$, while it is $d = 26$, with an augmentation encoding of size 16, for the AVIKAE model. As discussed throughout the manuscript, this constraint might be very restrictive in practice since it might not

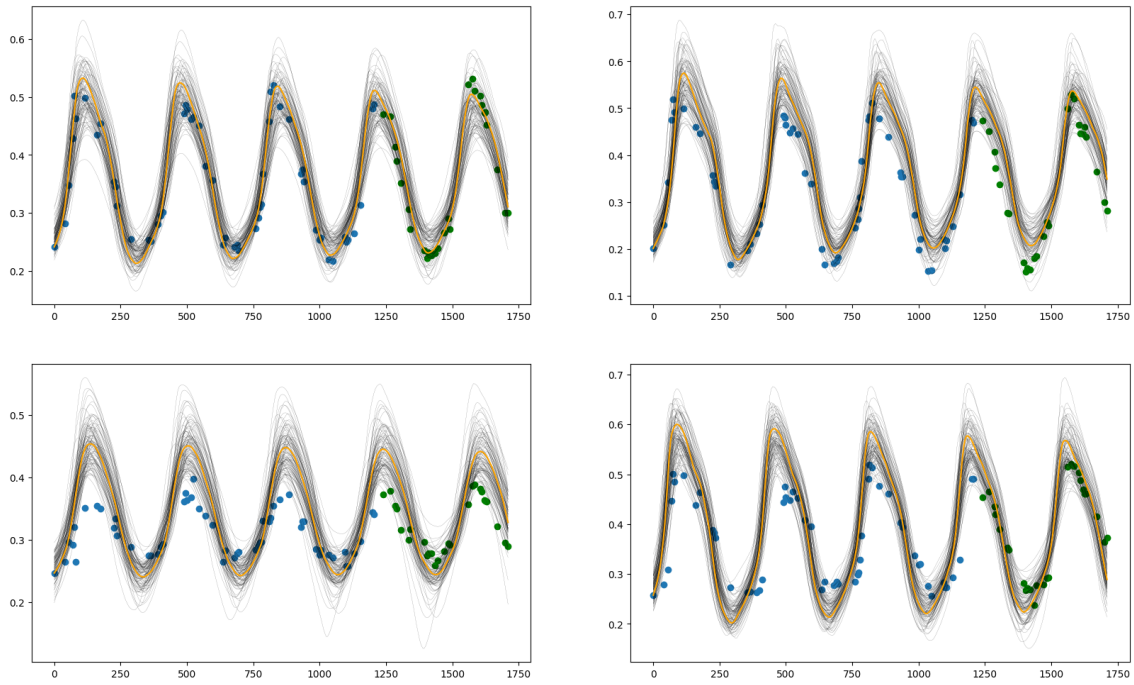

Figure 4.11 – Sample predictions for forecasting from an initial condition at time 0 with an AVIKAE model on the Fontainebleau training area. We represent the B7 spectral band, which is the most energetic one in the data. On each plot, the horizontal scale is in days, the blue dots are training data, the green dots are test data, the orange plot is the central prediction and the black plots are samples from the distribution of predictions.

be possible to design a good linear approximation $\mathbf{K} \in \mathbb{R}^{d \times d}$ with such a low latent dimension d. Thus, although a VIKAE with preservation of the input dimension might perform well on some datasets thanks to the robustness of its invertible encoding, it does not perform as well as a VKAE here. However, the results show that inflating the latent space with an additional augmentation encoding, as done by the AVIKAE model, can overcome this limitation to the point of beating the VKAE and the KAE ensemble.

Now that the good performance of the AVIKAE architecture for the Sentinel-2 datasets has been demonstrated, we leverage the pre-trained AVIKAE model that gave the results of table 4.1 to perform a series of experiments involving different inference techniques, mainly based on variational data assimilation, with the same self-supervised learning approach as in chapter 3.

First, we show that one can very easily account for the fact that an initial observation is assumed to be exact or noisy. To do so, we leverage the fact that the decoder of the AVIKAE model performs an exact reconstruction of its encoding and that the augmentation encoding has no immediate influence on the decoding but has an influence on its evolution through time. Thus, one can constrain the invertible part of the encoding to be equal to its mean value in order to ensure that the initial decoding is exactly equal to the input initial state. This constraint still enables to obtain a distribution of probability for the predictions instead of a single predicted trajectory, as long as the invertible part of the encoding is sampled in its initial Gaussian distribution rather than fixed.

Formally, we have that the classical way of performing predictions from an initial condition $\tilde{\mathbf{x}}_0$ is as follows:

$$\begin{aligned} \mathbf{z}_0 &\sim \mathcal{N}\left(\begin{pmatrix}\phi(\tilde{\mathbf{x}}_0)\\ \xi(\tilde{\mathbf{x}}_0)\end{pmatrix}, \operatorname{diag}(\chi(\tilde{\mathbf{x}}_0))\right), \\ \mathbf{z}_t &= \mathbf{K}^t \mathbf{z}_0. \end{aligned} \tag{4.64}$$

We argue that, when the observed initial state $\tilde{\mathbf{x}}_0$ is assumed to be a perfect representation of the state, one can use the following alternative procedure:

$$\begin{aligned} &\mathbf{z}_0^i = \phi(\tilde{\mathbf{x}}_0), \quad \mathbf{z}_0^a \sim \mathcal{N}(\xi(\tilde{\mathbf{x}}_0), \operatorname{diag}(\chi(\tilde{\mathbf{x}}_0)_{n+1:d})), \\ &\mathbf{z}_t = \mathbf{K}^t \mathbf{z}_0. \end{aligned} \tag{4.65}$$

In figure 4.12, we show an example of predicting through the procedures of equation (4.64) or (4.65) with the same pre-trained model and from the same observed initial condition. As previously mentioned, the second procedure enables to retain some variability in the predicted trajectories although it imposes its initial state to match the observed one. Importantly, one can clearly see that the empirical variance of the predictions increases with time, which corresponds to an expected behaviour. Although the base setup seems to give more convincing results here,

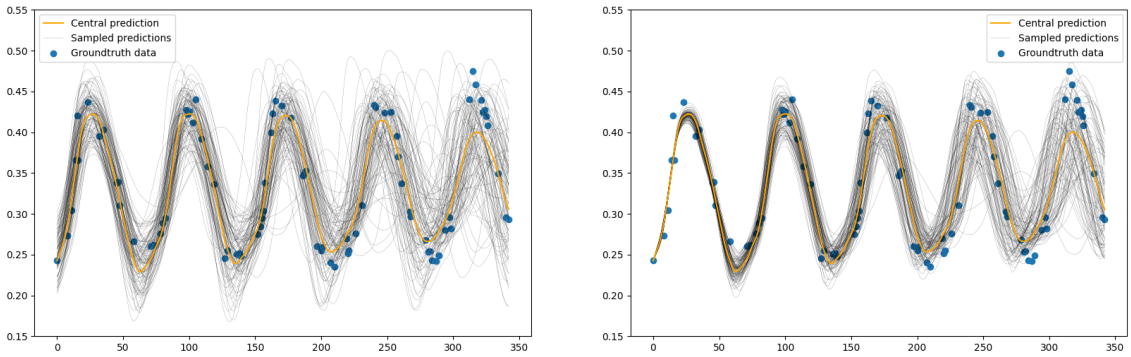

Figure 4.12 – Examples of AVIKAE predictions with two different setups. Left: the setup corresponding to the training of the model and to the plots of figure 4.11, where the invertible encoding is allowed to vary from the observed state when sampling. Right: a variant where the invertible part of the initial encoding is fixed and only the augmentation part is stochastic. In this case, one can see that the uncertainty of the predictions gradually increases but still remains much lower than in the left case.

there might be situations in which there is high certainty about an initial condition and we want to have less variability in the short-term predictions.

Interestingly, one could choose to design a variant of the AVIKAE that always predicts in a way that matches the second procedure by design. Indeed, one would simply need to redefine the variance encoder χ so that it produces variance vectors of size m, corresponding to the augmentation encoding, while keeping the invertible encoding deterministic. We chose to prioritize the variant of the AVIKAE model where the full latent state follows a Gaussian distribution since, as demonstrated, one can always impose a deterministic invertible encoding during inference.

Let us now discuss the adaptation of the variational data assimilation methods introduced in chapter 3 to the AVIKAE model. We will focus on the constrained assimilation methods here.

Adapting the constrained assimilation techniques of chapter 3 to a stochastic model (in particular a variational model) comes with several additional questions. Notably, one can wonder whether the goal of the optimization is still to find the best initial latent state or to find parameters for an "optimal" distribution of predictions. The latter problem is significantly more difficult to solve since metrics for comparing a distribution of predictions to a groundtruth (typically the CRPS) require heavy computation, and often have to be approximated when the distribution of predictions is not as simple as a finite ensemble of members. For this reason, we will only solve assimilation costs where the mean (or, more exactly, the central) prediction of the distribution is compared to the groundtruth. Thus, the procedure for the optimization will be the exact same as in chapter 3: we seek the initial latent state $\mathbf{z}_0^* \in \mathbb{R}^d$ which, after propagation through the latent dynamics and decoding of the invertible part, yields the closest trajectory to the groundtruth in terms of the mean squared error. Formally, to fit a (possibly incomplete)

trajectory of observations $(\tilde{\mathbf{x}}_t)_{t\in H}$, we solve

$$\mathbf{z}_0^* = \arg\min_{\mathbf{z}_0} \sum_{t\in H} ||\phi^{-1}(\mathbf{K}^t\mathbf{z}_0) - \tilde{\mathbf{x}}_t||^2. \tag{4.66}$$

Again, one can leverage the encoding $\phi(\tilde{\mathbf{x}}_0), \xi(\tilde{\mathbf{x}}_0)$ of the initial observation in order to provide a good starting point for this problem.

Since only the encoders of the AVIKAE model are stochastic, this results in only one output trajectory. Thus, in order to introduce stochasticity in the predictions after computing $\mathbf{z}_0^*$, one can simply re-encode it by computing

$$\mathbf{\Sigma}_0 = \text{diag}(\chi(\phi^{-1}(\mathbf{z}_0^*))). \tag{4.67}$$

Thus, one can then sample an initial condition $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{z}_0^*, \mathbf{\Sigma}_0)$, propagate it with $\mathbf{K}$ and decode it with ϕ^{-1} in order to obtain a corresponding sampled trajectory. One should note that this procedure is likely to be sub-optimal since χ is trained to produce a variance vector knowing only an initial condition $\tilde{\mathbf{x}}_0$ rather than knowing an entire trajectory. However, as we will show, this strategy still enables to get satisfying results, both qualitatively and quantitatively.

Table 4.2 – Forecasting CRPS and MSE, on the training Fontainebleau area and the test Orléans area. The Fontainebleau CRPS of the first column match the ones of table 4.1 but the Orléans CRPS of the first column slightly differs since it is now computed on a smaller range of times in order to assimilate on the rest of the available times.

	Prediction from the initial state	Assimilation of the trajectory
CRPS on Fontainebleau	0.0241	**0.0153**
MSE on Fontainebleau	0.00219	**0.000739**
CRPS on Orléans	0.0457	**0.0273**
MSE on Orléans	0.00738	**0.00236**

Let us now discuss the results of the aforementioned resolution of an assimilation cost. We use the same crops of the Fontainebleau and Orléans areas as in table 4.1, and solve equation (4.66) on the training period by automatic differentiation using the Adam algorithm [91] for 200 iterations with a learning rate of 10^{-2}. We then extend the obtained predictions in order to forecast beyond the training period. The mean squared error and the CRPS are reported in table 4.2 along with those corresponding to a direct inference from the initial condition as in table 4.1. One can see that, similarly to what we showed in chapter 3, taking into account several datapoints rather than only an initial observation enables to obtain significantly better results. In figures 4.13 and 4.14, we respectively show some visual results on random pixels of the Fontainebleau and Orléans areas, in order to give a qualitative assessment of the relevance of the predictions. In particular, by comparing figure 4.13 to figure 4.11, one can see that the data assimilation method

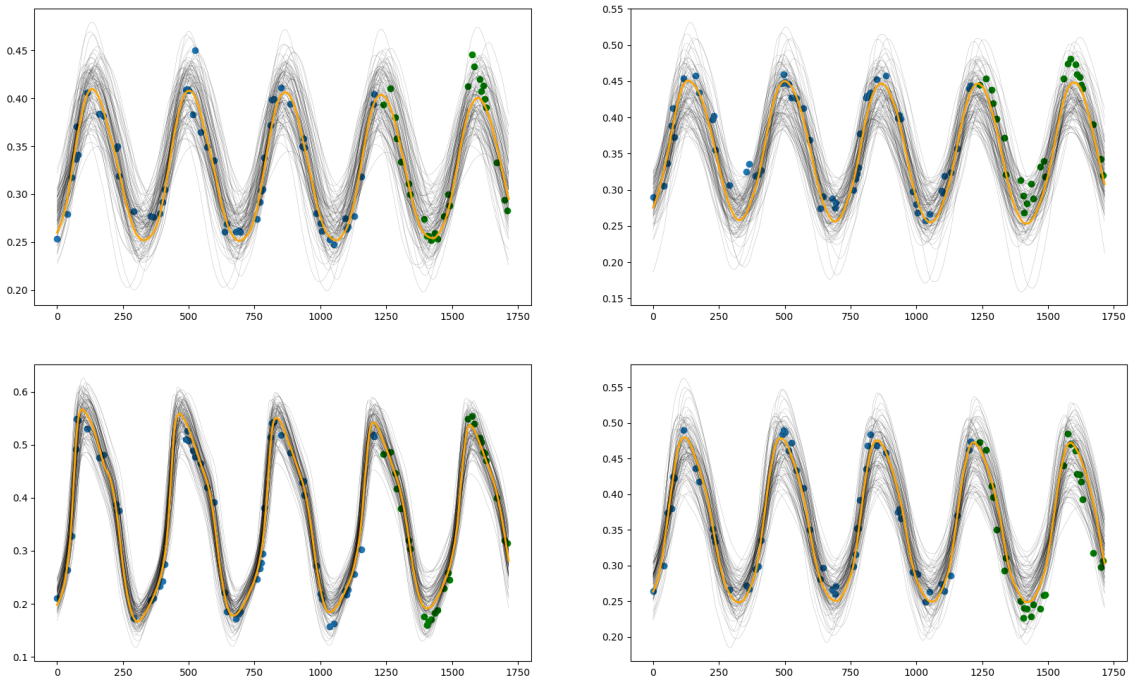

Figure 4.13 – Sample predictions for assimilation-forecasting with an AVIKAE model on the Fontainebleau training area. We represent the B7 spectral band, which is the most energetic one in the data. On each plot, the horizontal scale is in days, the blue dots are the assimilated datapoints, the green dots are the test datapoints, the orange plot is the central prediction and the black plots are samples from the distribution of predictions.

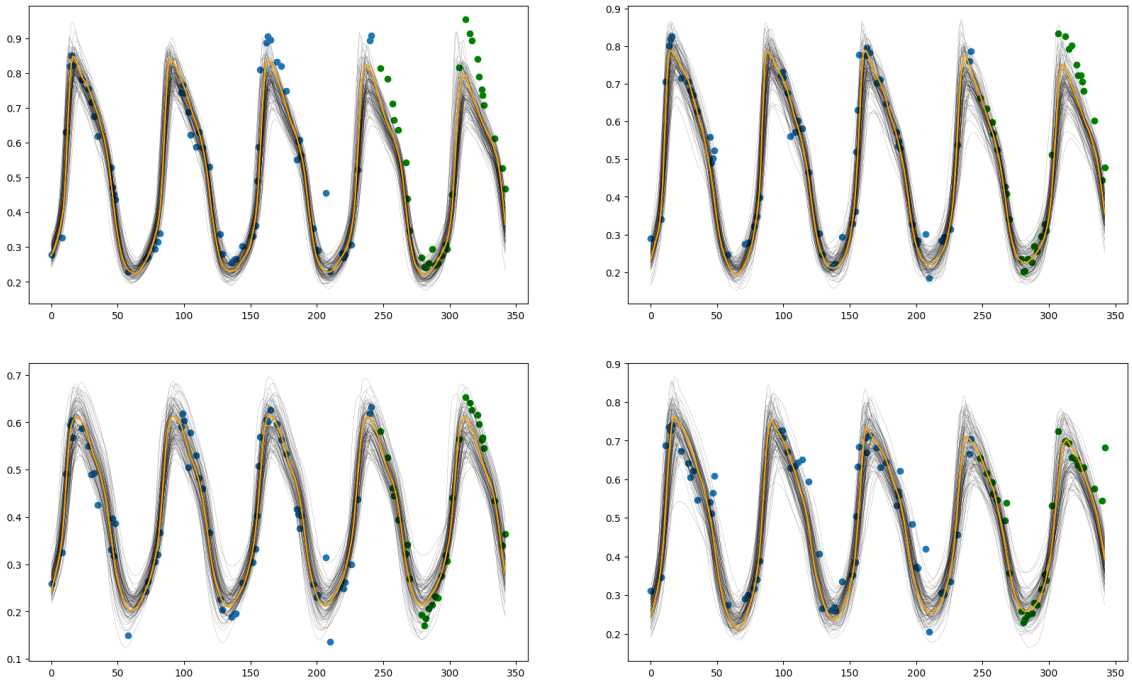

Figure 4.14 – Sample predictions for assimilation-forecasting with an AVIKAE model on the Orléans test area. We represent the B7 spectral band, which is the most energetic one in the data. On each plot, the horizontal scale is in days, the blue dots are the assimilated datapoints, the green dots are the test datapoints, the orange plot is the central prediction and the black plots are samples from the distribution of predictions.

indeed enables to obtain significantly more accurate predictions, which confirms the quantitative results of table 4.2.

4.4 Conclusion

In this chapter, we motivated the need for stochastic models in order to get a better notion of the uncertainty of the model predictions, either aleatoric or epistemic. In particular, we presented several existing frameworks of neural stochastic models such as GANs, VAEs and normalizing flows. We particularly focused on the conditional models which are of more interest for time series processing. Then, we introduced the stochastic models that have been designed throughout the thesis, which are twofold: the ensembles of Koopman autoencoders and the Augmented Variational Invertible Koopman AutoEncoder (AVIKAE).

Concerning the ensembles of Koopman autoencoders, we showed in section 4.2 that, although this technique indeed enables to obtain a quantification of the uncertainty of the predictions, the classical way of training the members independently from each other lead to overconfident predictions. Thus, we introduced new loss functions that enable to train the members jointly with a promotion of the inter-model variance, which results in more diversified predictions and therefore better CRPS.

As for the AVIKAE model, we introduced it in section 4.3 through successive refinements of the base KAE architecture. These refinements include the use of a variational autoencoder (i.e. one in which the encoder returns the mean and the variance of a latent Gaussian distribution instead of a single state), the use of an invertible encoder and the augmentation of the state of this invertible encoder. We showed how this model enables to obtain a complex probability distribution for the state of the trajectory at any time from the observation of an initial state. Then, we showed that AVIKAE compared favorably to the ensemble method of section 4.2 as well as to two other variants of variational KAEs in terms of the CRPS. Finally, we showed that a pre-trained AVIKAE can be used as a dynamical prior in the constrained assimilation method of chapter 3 in order to further improve the forecasting performance.

There are many interesting perspectives for the AVIKAE model. First, the model could be changed in order for its latent propagation to be stochastic. This could open the way to the computation of the joint likelihood of entire observed trajectories (instead of only the marginal distributions for each time index), for inference as well as for training the model.

Concerning the usage of the AVIKAE model as a data assimilation prior, the stochasticity of the model means that there are many new possibilities to explore compared to what was presented for KAE models in chapter 3. Our procedure for obtaining a probabilistic prediction in an assimilation-forecasting context, summarized in equations (4.66) and (4.67) yields qualitatively and quantitatively good results, yet it is intuitively sub-optimal since the variance of the

predictions is not explicitly fitted to the assimilated data. One relatively simple way to improve this result would be to jointly optimize on an initial latent mean and (diagonal) variance, with an assimilation cost consisting of a probabilistic metric such as the CRPS.

In addition, we have only computed probabilistic forecasts for individual pixels. Yet, when naively sampling a trajectory for a large area with an assumption of spatial independence between the pixels, we would certainly obtain visually incoherent results. Thus, the question of how to produce a physically realistic joint probabilistic forecast of an image is still unanswered so far.

Conclusion and perspectives

Summary of contributions

Throughout this manuscript, we have presented various methodological innovations for processing time series, primarily based on Koopman operator theory and on variational data assimilation. Although these methods are extensively applied to satellite image time series (SITS), they are not specifically designed for these particular data, and they could be used in other fields.

Our main key for understanding time series is the dynamical system perspective, which was introduced in section 1.1. This section notably described the correspondence between continuous and discrete formulations of dynamical systems, which is important since a time series might be seen as originating from a dynamical system which might have a continuous formulation. In section 1.2, we gave a description of linear dynamical systems, which intervene later to describe the Koopman-based methods of chapter 2. In section 1.4, we discussed the specificities of the SITS data. In particular, we introduced the SITS datasets that have been created as a part of the thesis work.

In chapter 2, we have presented the Koopman operator theory and some of the numerous methods that are based on it. This study was focused on the Koopman autoencoder (KAE) framework, consisting in an autoencoder in which the dynamical evolution of the state is linear in the latent space, and therefore governed by a Koopman matrix $\mathbf{K}$. We provided a detailed analysis of the loss function from equation (2.14), first introduced by [32], which we consider to be a baseline for training a KAE. Then, in section 2.2.1, we introduced a new loss function term that promotes the (approximate) orthogonality of $\mathbf{K}$, and showed that the addition of this term to the loss function of equation (2.14) enables to improve the long-term prediction performance of a KAE through an experiment on our collected SITS. Other experiments in section 2.2.2.b and 3.2.2.b then further confirmed these results. In section 2.2.2, we leveraged our discussion of linear dynamical systems from section 1.2 in order to show that KAE models are able to produce continuously-evolving trajectories. We concretely showed how this enables to perform a temporal upsampling, even after training on scarcely sampled data, in section 2.2.2.a. Then, in sections 2.2.2.b and 2.2.2.c, we discussed and demonstrated the training of KAE models with irregularly-sampled data.

Apart from the upsampling to a higher temporal frequency, the experiments of chapter 2 were mainly performed in a supervised learning approach. Thus, in chapter 3, we showed how the

training of a KAE to a simple forecasting task, as presented in chapter 2, can actually be seen as a pretext task in a broader self-supervised learning context. In particular, we first introduced the concepts of variational data assimilation in section 3.1, before proposing several ways to use a KAE as a dynamical prior for variational data assimilation in section 3.2. These ways include an unconstrained variational cost where the optimized variable is the output time series, a constrained variational cost where the optimized variable is the initial latent state of the model, and a fine-tuning cost, where the parameters of the pre-trained model are optimized jointly with the latent initial state. We then performed experiments on our SITS datasets using these newly introduced methods, for three different tasks: denoising, forecasting and interpolation. We showed that the unconstrained approach was more suitable for denoising in contexts where the noise magnitude is relatively low. We also showed that the variational assimilation costs enable the inclusion of spatial regularisation terms, thus allowing for a joint inference on the pixels of an image, while previously introduced methods processed the pixels independently from each other.

Finally, in chapter 4, we have been considering stochastic aspects of the predictions performed by KAE models, notably for uncertainty quantification. We began by presenting stochastic models, with a particular focus on deep learning models, in section 4.1. Then, in order to quantify the uncertainty of the predictions, we discussed the training of ensembles of KAEs in section 4.2. We showed that the classical deep ensemble approach from [137], consisting in training the models independently from each other, leads to overconfident predictions in our case. Thus, we introduced variability-promoting loss terms, which involve all members of an ensemble of KAE models. We showed that these new loss functions enable to significantly improve the quality of the uncertainty estimates. Additionally, we introduced the AVIKAE architecture: an extension of the classical KAE which enables to predict complex probability distributions, with some density estimation capabilities. We showed that this model performed better stochastic predictions than other stochastic Koopman models, including the ensembles trained with the methods of section 4.2. We proceeded by discussing the possibilities offered by the AVIKAE model, and notably its usage as a dynamical prior like we did for a basic KAE in chapter 3.

Perspectives

Modelling other types of land occupation

Although we have demonstrated promising capacities to forecast, interpolate and denoise SITS, there remain important limitations to the extent of our results. The most important of these limitations is probably the fact that the data on which we worked were mainly composed of forest areas. Although these data already have some level of diversity, they are not nearly as diverse as what can be found over the surface of the Earth.

A good entry point to applying our methods on more diverse data and downstream tasks would be agriculture. Like the data that we consider in the current study, crop image time series largely evolve according to pseudo-periodic patterns. These data are of interest for, e.g, crop type classification and change detection: see [149] for a recent review. There exist several SITS datasets [150, 151] that are oriented towards this objective, with SITS generally spanning a duration of one year and associated to crop type labels. Unfortunately, those SITS are not formatted in ways that are ideal for training our models. In [150], not all clouds are filtered out from the SITS. Thus, it would be ineffective to solve a pretext forecasting task here since our model would basically learn to reconstruct the reflectance of the clouds. In [151], the SITS are heavily formatted in order to keep only one value per month, hence critically damaging the temporal structure of the data. Nonetheless, it could be interesting to test the few-shot or fine-tuning performance of a KAE pre-trained on SITS similar to ours and tested on these crop type datasets. In order to obtain optimal performance, one would certainly need to include crop data, and not only forest data, in the pre-training dataset.

Time series classification

One way to perform time series classification as a downstream task for a pre-trained KAE might be to, again, resort to constrained variational data assimilation. Concretely, we propose to 1) Solve the constrained assimilation cost of equation (3.19) for several time series, and store the corresponding initial latent states $\mathbf{z}_0^*$ along with the labels of the time series 2) Train a model to classify the labels using the initial latent states $\mathbf{z}_0^*$ as the input.

The underlying idea is that, if the constrained assimilation enables to perform a good reconstruction of the considered time series, then the latent vectors $\mathbf{z}_0^*$ that encode the reconstructed time series should contain all the necessary dynamical information that is required to classify this time series. Besides, since the latent dynamical model is all driven by the matrix $\mathbf{K}$, the classification according to $\mathbf{z}_0^*$ should not be overly complex. In fact, there might even be theoretical tools that we are not aware of yet that would enable to design a specifically adapted classifier in step 2) instead of a classical machine learning model.

Change detection

Concerning change detection, we believe that our constrained assimilation method may also be helpful. Concretely, given an observed time series, the goal would be to assess whether there is, at one point, an unusual and unpredictable change in the dynamics, and the time on which it happens. First, if the time series at hand is similar to what can be found in the pre-training dataset, then it should be well reconstructed as a whole. In contrast, if the dynamics do not correspond to what can be found in the pre-training dataset, then our method should not be able to accurately reconstruct it, even for a relatively small time window. The case that we find the most interesting is the one in which the time series first follows known dynamical patterns, and then starts following a different pattern at a precise point in time. In this case, the constrained

assimilation method should be able to accurately reconstruct the time series up to this point, and struggle afterwards. In order to use this assumption in practice, we propose two methodologies:

1) Reconstruct the whole time series through constrained assimilation, and plot the MSE of the reconstruction as a function of time. This function should have an increasing trend, with an "elbow" (i.e. a point where the trend accelerates) potentially marking a change in the dynamics.

2) Perform the constrained assimilation on several different time horizons, and plot the MSE of the reconstruction as a function of the horizon considered. Again, there should be an increasing trend, with an elbow marking a change.

We conjecture that the method 2) might be more robust than 1), at the cost of a higher computational burden.

Besides, the detection of anomalies, whether at the level of a point or of a time series, can be naturally performed with a stochastic model such as the AVIKAE. Indeed, an anomaly can simply be characterised as an event that is unlikely according to a trusted stochastic model. Concretely, one can perform direct likelihood computations using the inherent capacities of the AVIKAE model. Another possibility is to sample multiple model trajectories given the available data, and to leverage these samples to compute confidence intervals or empirical statistics (e.g. with a Gaussian approximation).

Better accounting for the spatial structure of the data

With the methods considered throughout the manuscript, we have been able to successfully account for the temporal and spectral dimensions of the SITS. However, the spatial structure of the data has only been marginally taken into account. The main method that we proposed to do so was the inclusion of a spatial prior S in the variational assimilation costs of equations (3.16), (3.19) and (3.20). However, in our experiments of chapter 3, we only implement this spatial prior as a very simple Tikhonov prior. Although we observe an improvement of the accuracy of our predictions thanks to this prior, the gain is relatively modest and we conjecture that more could be gained from the spatial information. Prior to designing this method, in [152], we proposed to train a convolutional neural network (CNN) to correct the errors made by a pre-trained KAE. This CNN can be trained on the training data or assimilated data, either when forecasting from a single observation or through assimilation-forecasting. The inconvenient of this approach is that it necessitates training a new model for each task and spatial area. Ideally, we would like to find a spatial prior that is as universal as the Tikhonov prior (in the sense that it is independent on the task and area) but with a better capacity to improve the performance of a KAE.

A natural extension of our work would be to directly learn a spatio-temporal prior model instead of a solely dynamical prior model. Thus, instead of decoupling the dynamical and spatial priors as we do in our assimilation frameworks of section 3.2.1, we would jointly take into account these sources of information. Promising strides in this direction have been made during the

internship of Clément Lacrouts, which was closely linked to the work of this thesis, and which we discuss in appendix C.

Dynamical spectral unmixing

Finally, a task that could be addressed through the self-supervised approach that we take in this thesis is to separate the different materials of the images in an unsupervised fashion. In its simplest form, this task amounts to clustering the pixels according to their spectral contents, with the underlying assumption that each pixel contains information associated to a given material, e.g. grass or trees, which can be characterized by a somewhat variable spectral signature. More generally, one can assume that the pixels are not always pure, which means that they often contain several materials, from which the spectral information is (often linearly) mixed. This leads to the spectral unmixing problem, which consists in estimating the proportions of different materials in each pixel of an image. While this task is generally carried on a single image, we are working with SITS here, and thus we can further assume that the content of each pixel does not change in time and that the signatures of the materials are in fact spatio-temporal. Interesting preliminary results have been obtained with different data-driven models during the internship of Tevchhorpoan Khieu, which was also linked to the thesis work. Notably, connections between dynamical spectral unmixing and data-driven Koopman models like the DMD have been drawn. The internship report will soon be available online.

Appendix A

A BRIEF REVIEW OF THE USE OF ORTHOGONALITY TO PREVENT VANISHING AND EXPLODING GRADIENTS

Vanishing and exploding gradients are major issues when training a neural network through backpropagation. They have first been evidenced by [153] and then further characterized for time series processing models by [154]. A more recent description of these issues can be found in [155]. They can be understood intuitively as originating from the multiplication of the gradients of a large number of trainable components, resulting in excessively low or high gradients. These problems have been extensively studied for recurrent neural networks (RNNs), which are particularly subject to vanishing and exploding gradients when they are trained over very large sequences of data. The celebrated long-short term memory (LSTM) architecture [156] was introduced in order to alleviate these issues, which it is able to do up to a certain extent. However, this architecture alone was not sufficient to completely deal with the vanishing and exploding gradients, which is why subsequent works tried to address them by introducing hard or soft constraints on the parameters of a RNN.

As an example, [157] proposed to initialize the recurrent weight matrix of a RNN to an identity matrix rather than through the otherwise standard Glorot initialisation [158]. The authors of [157] showed that this simple initialization enabled to significantly outperform a classical LSTM model on the sequential MNIST dataset, which consists in viewing an image as a sequence of 784 pixels that are passed through a sequence processing model. Later, [159] discussed the Copy problem, originally introduced in [156], which consists in reproducing the first tokens of a sequence of digits after processing a large number of subsequent elements. They analytically showed that this problem could be solved using a RNN with no hidden-to-hidden non-linearity and from which the weight matrix has well distributed eigenvalues on the unit circle. Consequently, they proposed to initialise a RNN model with a random orthogonal weight matrix, and showed that this significantly boosted the performance on the Copy task compared to a Glorot or identity initialisation.

Several subsequent works [67, 160, 130, 161, 162] have proposed to constrain the recurrent

matrix to remain inside of the orthogonal group (or the unitary group) throughout the training process, by leveraging different parameterizations of these groups. Remind that, by definition, the unitary group is a generalization of the orthogonal group of matrices to the set of complex matrices. Indeed, for a given integer size d, it corresponds to the space of matrices $\mathbf{A} \in \mathbb{C}^{d \times d}$ such that $\mathbf{A}\mathbf{A}^* = \mathbf{I}_d$, where $\mathbf{A}^*$ is the conjugate transpose of $\mathbf{A}$. The authors of [67] exhibit a bound on the propagated gradients in recurrent neural networks for which the recurrent matrix is orthogonal, which motivates their idea to restrain the optimization to the set of unitary matrices. To do so, a parameterization of this set as a product of simple unitary matrices is proposed, leveraging the property that the set of unitary matrices is stable by product. In this way, the number of parameters is linear in the size d of the latent space, while a more naive approach would have yielded a quadratic growth of the number of parameters. However, as demonstrated by [160], this representation has a limited capacity as it does not enable to obtain any desired unitary matrix. For this reason, the authors of [160] propose to directly perform gradient descent in the Stiefel manifold of unitary matrices, by leveraging a descent curve first suggested by [163], which results in the so-called full-capacity unitary RNN model. Subsequently, [161] showed that using a unitary matrix as a recurrent matrix in a RNN was equivalent to using a higher-dimensional orthogonal matrix, and consequently advocated for focusing on real orthogonal matrices in order to avoid the difficulties associated to manipulating matrices with complex coefficients. The authors proposed a parameterization of orthogonal matrices using Householder reflections [164]. Later, [162] proposed to parameterize orthogonal (or unitary) matrices through the exponential map of Lie group theory, with similar theoretical foundations as the ones used by [37] in a Koopman autoencoder context.

It should be noted that the matrix exponential function is the link between the set of skew-symmetric matrices and the set of special orthogonal matrices, but also the link between the continuous and the discrete formulations of linear dynamical systems. For this reason, a linear dynamical system that is governed, in a discrete formulation, by an orthogonal matrix, will be described by an antisymmetric matrix in a corresponding continuous formulation. This property motivated the use of ordinary differential equations for the formulation of recurrent neural networks. In the antisymmetricRNN model [165], the RNN cell is the discretization (through a Euler resolution scheme) of the ordinary differential equation

$$\mathbf{h}'(t) = tanh((\mathbf{W}_h - \mathbf{W}_h^\intercal)\mathbf{h}(t) + \mathbf{V}_h\mathbf{x}(t) + \mathbf{b}_h, \tag{A.1}$$

where $\mathbf{h}$ is the RNN's hidden state, $\mathbf{x}$ is the observed state and $\mathbf{W}_h, \mathbf{V}_h, \mathbf{b}_h$ are trainable parameters of the model. One can see that the matrix $\mathbf{W}_h - \mathbf{W}_h^\intercal$ is antisymmetric, hence making this approach similar in spirit to the orthogonality-based methods. The Jacobian associated to

this ODE is then

$$\mathbf{J}(t) = diag[\tanh'((\mathbf{W}_h - \mathbf{W}_h^\intercal)\mathbf{h}(t) + \mathbf{V}_h\mathbf{x}(t) + \mathbf{b})](\mathbf{W}_h - \mathbf{W}_h^T). \tag{A.2}$$

The eigenvalues of $\mathbf{J}(t)$ are thus all imaginary, i.e. their real parts are zero. This means that it conserves the information without diverging to infinity, as the associated discrete formulation of this matrix (which we use more often throughout the manuscript) then has its eigenvalues on the unit circle. Additionally, since the discretization of equation (A.1) tends to actually yield unstable dynamics in practice (although the exact ODE exhibits stable dynamics), the authors propose to add a regularisation diffusion term in the antisymmetric matrix, which consists in subtracting all diagonal coefficients by a small coefficient $\gamma > 0$. Equation (A.1) thus becomes

$$\mathbf{h}'(t) = tanh((\mathbf{W}_h - \mathbf{W}_h^\intercal - \gamma\mathbf{I})\mathbf{h}(t) + \mathbf{V}_h\mathbf{x}(t) + \mathbf{b}_h. \tag{A.3}$$

This ODE scheme is close to the parameterization chosen by [63] (which is a Koopman autoencoder model discussed in section 2.1.4) in equation (2.30). The main idea is indeed also to obtain stable dynamics through the model. According to [165], the inclusion of a coefficient of diffusion inside the weight matrices improves the classification performance on the sequential MNIST dataset over the constrained orthogonality proposed by [67]. Thus, [165] concludes that a softened orthogonality constraint, as in [68] (similar to our orthogonality loss term in [65, 66]) is more effective than a hard orthogonality constraint as in [67] (similar to the HNKO approach in [37]).

Appendix B

A brief review of self-supervised learning

Self-supervised learning [166] is a learning paradigm that greatly differs from the main approach of supervised learning, as it is closer in practice to unsupervised learning. It is generally used in a context where a large amount of data is available, but with very scarce or even no labels associated to this data. The main idea is to learn a representation or encoding of the data [1] by training a model on a pretext task that does not require any labels. One can then use this representation to solve the tasks that are actually of interest, called the downstream tasks.

As an example, let us suppose that we are trying to solve a task of image classification, and that we have at disposal a large number of images, for which only a very small subset has an associated categorical label. In such a context, simply training a neural network in a supervised way to classify the labelled images proves to be ineffective since the amount of available data is too low for the classifier to properly generalize to other data, which leads to the well-known overfitting problem. Thus, an unsupervised approach consists in leveraging a pretext task that does not require any labels, and enables to train an embedding neural network to recognize generally useful information in the data. In order for this approach to be successful, the pretext task has to be a nontrivial task that requires to have a notion of the semantics of the images. Therefore, a model that is able to successfully solve the pretext task is expected to hold enough information to solve the classification task, which is called a downstream task in this case.

Let us now mention some of the most popular pretext tasks that are used on image data. Many popular pretext tasks consist in applying some simple transformation on a sampled unlabelled image and then trying to predict the exact nature of this transformation. For example, one can rotate a source image by a random angle and then ask the embedding neural network to predict, from the rotated image, the angle of the rotation that has been performed, as proposed in [167]. One can also sample 2 patches of pixels in the image and try to predict their relative positions [168]. More generally, the authors of [169] propose to separate an image into many square patches of pixels, randomly shuffle these patches and ask a model to solve the corresponding jigsaw puzzle.

1. Some articles use the keyword "representation learning", which is conceptually very similar to self-supervised learning.

One family of approaches, referred to as contrastive self-supervised learning [170], consists in learning similar representations for images that are linked by a simple transformation. Typically, one can pick an unlabelled image from the dataset and perform a rotation, a color transfer, a horizontal or vertical flip, a crop or a combination of these simple transformations. This results in a new, transformed image, that qualitatively contains the same information as the source image. Thus, the latent representation of the transformed image produced by the trained encoder network is expected to be close to the latent representation of the source image. In contrast, the transformed image is expected to hold a different content than another randomly sampled image in the dataset or a simple transformation of this other image. Thus, contrastive learning consists in minimizing the distance between the encodings of images linked by simple transformations while maximizing the distance between the encodings of other pairs of images.

Besides from image datasets, self-supervised learning has recently shown remarkable success in natural language processing, where it significantly outperformed classical supervised approaches. Indeed, classical methods consisted in training a LSTM model [156] on a specific language task, such as question answering [171] or the translation from one specific language to another [172]. Later, [173] proposed to first pre-train a general transformer model [142] on the masked language modelling task, which consists in predicting randomly masked tokens in text datasets using the context of surrounding unmasked words. They showed that fine-tuning this pre-trained model on specific downstream tasks enabled to achieve better results than specific supervised models for these tasks. Then, by leveraging similar pre-training methods with larger models and datasets [174, 6], it was shown that the pre-trained models are actually able to solve a variety of tasks with a few-shot or zero-shot learning approach rather than a classical fine-tuning approach. These developments lead to the predominance of the so-called large language models [175].

Appendix C

Spatio-temporal models

Here, we discuss models that enable to process SITS by jointly taking into account the temporal, spectral and spatial dimensions of the data. The content of this appendix is based on the internship report of Clément Lacrouts [176], which was jointly supervised by Lucas Drumetz and Anthony Frion.

As a reminder, we show in figure C.1 a schematic view of the approach adopted throughout the thesis, with our KAE models working at the level of a single pixel. Let us now present two simple ways to extend this approach in order to include spatial information in the data-driven models.

The first model, which we call a convolutional KAE, consists in replacing the pixelwise encoding of the KAE by a pre-processing of an entire image through a convolutional neural network (CNN). Thus, we have that the latent state of each pixel also contains information on some neighboring pixels. Then, the latent dynamics still consists in multiplying the latent state of each pixel by the same matrix $\mathbf{K}$. Afterwards, the decoder of the model again shares the information between several pixels through a CNN. The convolutional KAE is represented in figure C.2.

According to the viewpoint of the convolutional KAE, the pixelwise approach of figure C.1 can be seen as a special case where, in fact, only convolutional layers with a kernel of size 1×1 are used. In the more general convolutional KAE, one can use convolutional layers with bigger filters. We emphasize that the latent dynamical model still processes the pixels independently from each other: another possibility would be to consider the whole encoded image as the latent state, yet it would require a very large latent dimension and consequently a very large matrix $\mathbf{K}$.

Another possibility is to conserve the architectures of the encoder and decoder of the KAE, but to allow the matrix $\mathbf{K}$ to take into account the latent state of several neighboring pixels in order to predict the future state of one single pixel. This approach, which we call kernel KAE, is illustrated in figure C.3.

Outside of the KAE framework, there are a lot of possibilities for building a dynamical model that takes the spatial dimensions of the SITS into account in order to perform its predictions. One simple possibility would be to directly model the advancement by a certain amount of time as the action of a CNN. We took particular interest in an alternative choice, based on Fourier neural operators (FNOs).

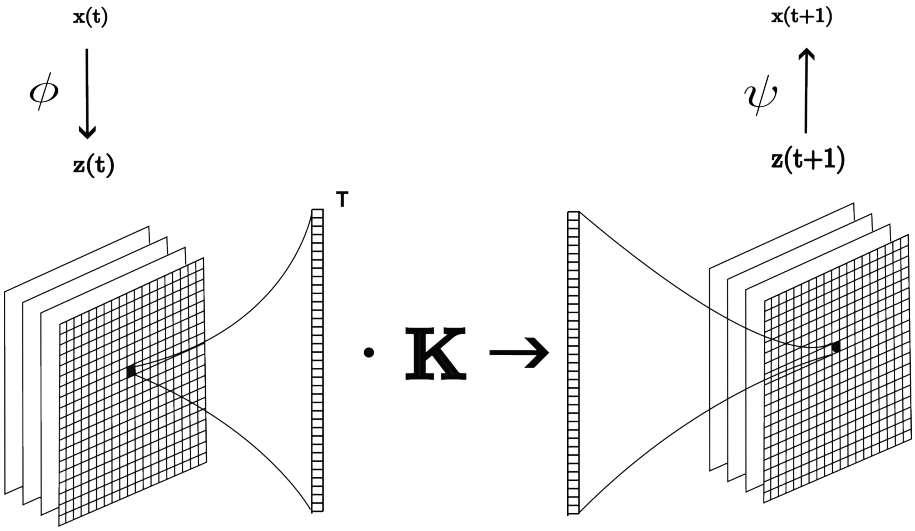

Figure C.1 – Graphical representation of our SITS forecasting approach from chapter 2. The encoder ϕ, the latent evolution matrix $\mathbf{K}$ and the decoder ψ all process the pixels of the SITS independently from each other.

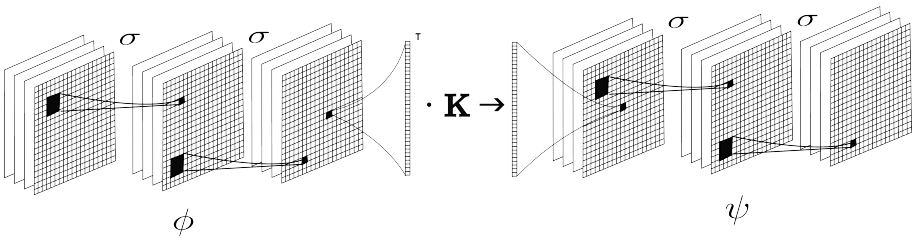

Figure C.2 – Graphical representation of the convolutional KAE for processing SITS. Here, both the encoder and the decoder share information between the pixels instead of processing each of them independently.

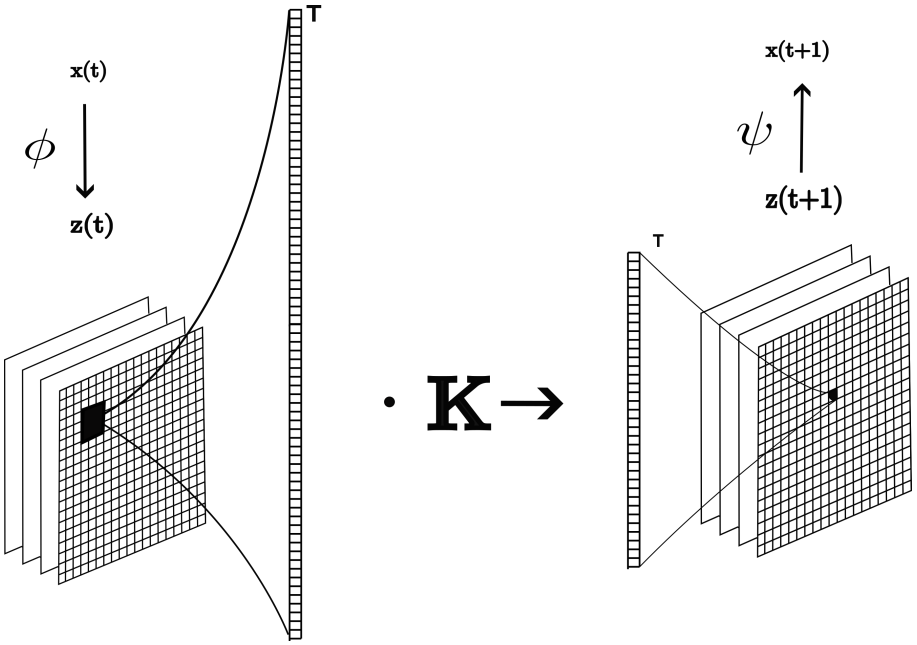

Figure C.3 – Graphical representation of the kernel KAE for processing SITS. Here, the encoder and the decoder have the same architecture as in the classical KAE approach, but the matrix $\mathbf{K}$ takes a kernel of several pixels as input for predicting the next state of a single pixel. Hence, $\mathbf{K}$ is not square in this case.

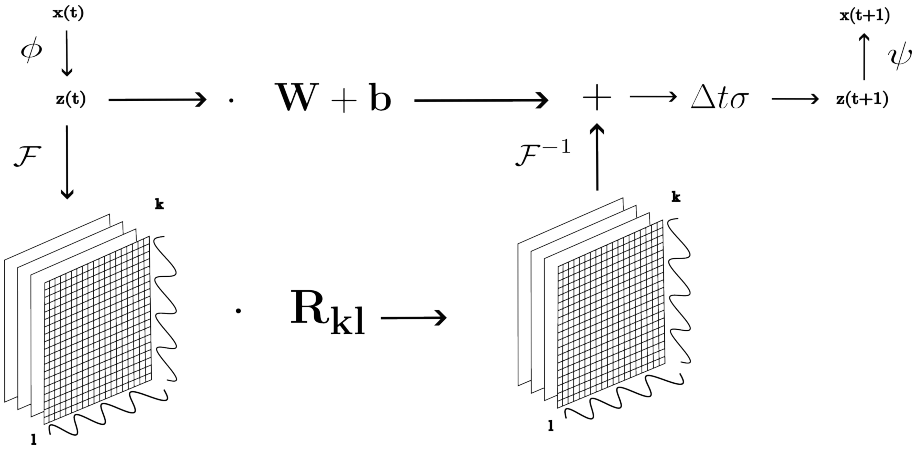

Figure C.4 – Graphical representation of a FNO approach for SITS. Here, we learn a FNO with one layer, with the learnt components ϕ, ψ, $\mathbf{W}$, $\mathbf{b}$, and $\mathbf{R}_{kl}$.

Model	RMSE	Number of parameters
KAE	**0.0207**	304 K
convolutional KAE	0.0270	304 K
kernel KAE	0.0236	312 K
FNO	0.0223	2.4M

Table C.1 – Forecasting root mean squared error of our KAE model from section 2.2, compared to several models that take into account the spatial dimension of the data, for a set of data from the Fontainebleau area.

The base principle of an FNO model is that the operator of time advancement is partially learnt in the Fourier domain. Indeed, in a layer of such a model, a Fourier transform is performed on the latent vector and a learnt matrix multiplication is applied in this domain before performing the inverse Fourier transform. In addition, since these models may tend to diverge in the long term, a new energy conservation loss term has been proposed during the internship. This term, inspired by the orthogonality loss term for KAE models (which we discussed in section 2.2.1), seeks to promote the conservation of the norm of the latent vectors throughout the predictions. However, the latent evolution for FNO models is not linear, hence one cannot simply promote the orthogonality of a latent evolution matrix like we do in KAE models. Instead, one possibility is to simply penalize the square of the difference between the squared L^2 norms of the encodings of consecutive states. With simplified notations, this stability loss term can be expressed as

$$L_{orth} = \sum_{i=1}^{N} \sum_{t=1}^{T} (||\mathbf{z}_t||_2^2 - ||\mathbf{z}_{t-1}||_2^2)^2. \tag{C.1}$$

In order to assess the possibilities offered by the differing architectures that were introduced, several experiments have been performed on SITS.

First, an experiment has been performed in a setup similar to the one of section 2.2.1, where the goal is to perform forecasting of SITS that are regularly sampled in time as a result of a pre-processing interpolation step. The performance of the different models, measured with the root mean squared error (RMSE), are presented in table C.1.

The conclusion of this experiment is that, when the data are not particularly noisy or difficult to process, processing the pixels independently from each other is largely sufficient in order to perform an accurate forecasting. In fact, the purely temporal model performs better here than the three spatio-temporal models described above.

However, it appears that the spatio-temporal models are significantly more robust than the temporal KAE model. In order to assess this additional robustness, we use the exact same dataset and add a random noise to the initial condition to the long-term forecasting. The signal-to-noise ratio is 30 dB, which represents a relatively low noise. The root mean-squared errors obtained

Model	RMSE
KAE	0.03667
convolutional KAE	0.03296
kernel KAE	0.02423
FNO	**0.02385**

Table C.2 – Forecasting root mean squared errors in the same setup as for table C.1, except that an artificial noise has been added to the input of the models.

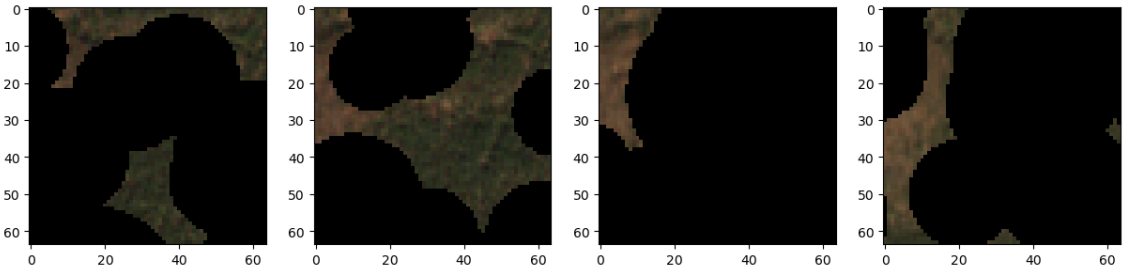

Figure C.5 – Samples of the synthetic incomplete SITS with a spatio-temporal masking pattern.

when forecasting from this noisy initial condition with the 4 tested models are reported in table C.2.

Here, the best performing models are the FNO and the kernel KAE model, while the base KAE performs worst. Our interpretation for this result is that taking into account the spatial dimensions of the data largely enables to mitigate the noise. The easiest spatial pre-processing that one can think of is an averaging of the neighbors of each pixel, with a similar spirit to the moving average classically applied on the temporal dimension of time series to remove the high frequency patterns.

In order to further assess the potential gains of leveraging a spatio-temporal model rather than a purely temporal model, we introduce a synthetic spatio-temporal masking procedure, consisting in masking some parts of the regular version of the Fontainebleau time series presented in section 1.4.2. The reason for creating a synthetic mask rather than directly using the irregular version of the dataset is that this version only presents a temporal masking pattern. In other words, the images are either fully retained or not used at all, which is a very simple masking pattern.

In the masking procedure used as part of the internship, some images are left unchanged, while the remaining ones are randomly masked with imaginary circular clouds, from which the locations are randomly and uniformly sampled, with parameters adjusted to have a desired rate of missing pixels on average. Some sample masks obtained with this procedure are shown on figure C.5.

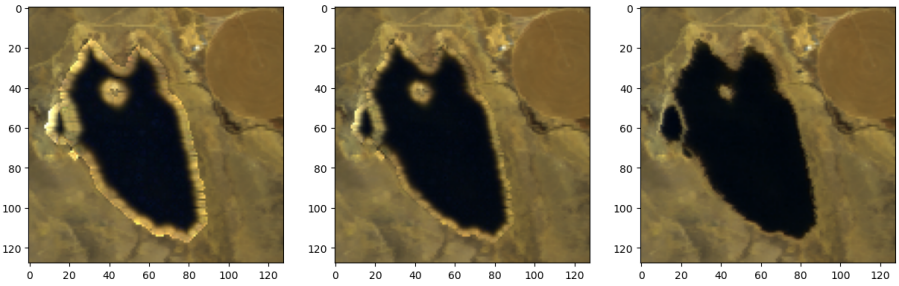

Figure C.6 – Samples of the synthetic dataset of an eroding and dilating lake.

Using this procedure, we test different forms of data-driven dynamical (or spatio-temporal) priors for performing spatio-temporal interpolation based on an (unconstrained) data assimilation framework as described in equation (3.7). We assimilate using a mask with 90% of cloudy days, and an overall proportion of 70% masked pixels. The assimilated time series are then compared to the groundtruth in order to compute mean squared errors. The results of the different models are summarized in table C.3. We report the mean squared error as well as the standard deviation of this error according to different masks.

Prior	**MSE** ($\times 10^{-3}$)	**standard deviation** ($\times 10^{-3}$)
KAE	0.6994	0.2381
convolutional KAE	0.5410	0.2365
kernel KAE	0.4564	0.1433
FNO	**0.3771**	0.1905

Table C.3 – Interpolation errors when assimilating data with a spatio-temporal masking pattern, using different pre-trained models as priors for the assimilation.

Again, one can observe that the spatio-temporal models perform better than the pixelwise KAE model here, since they are able to share the information between neighboring pixels. In particular, the FNO, which is able to share information all across the image thanks to its Fourier layer, yields the best interpolation performance. Thus, spatio-temporal models are better able to leverage spatio-temporal masking patterns, while the KAE model is more suited for SITS with only a temporal masking pattern, as was the case in sections 2.2.2.b and 3.2.2.c.

Finally, we introduce another synthetic dataset, in which the spatial information has to be leveraged in order to differentiate pixelwise initial conditions that look identical but lead to differing dynamical patterns. This dataset is based on a remote sensing image of a small lake called Mud Lake near lake Tahoe, at the border between California and Nevada. Using mathematical morphology operations [177], we simulate a periodic change in the level of the lake, which translates in successive erosions and dilations of its borders. The dataset is illustrated

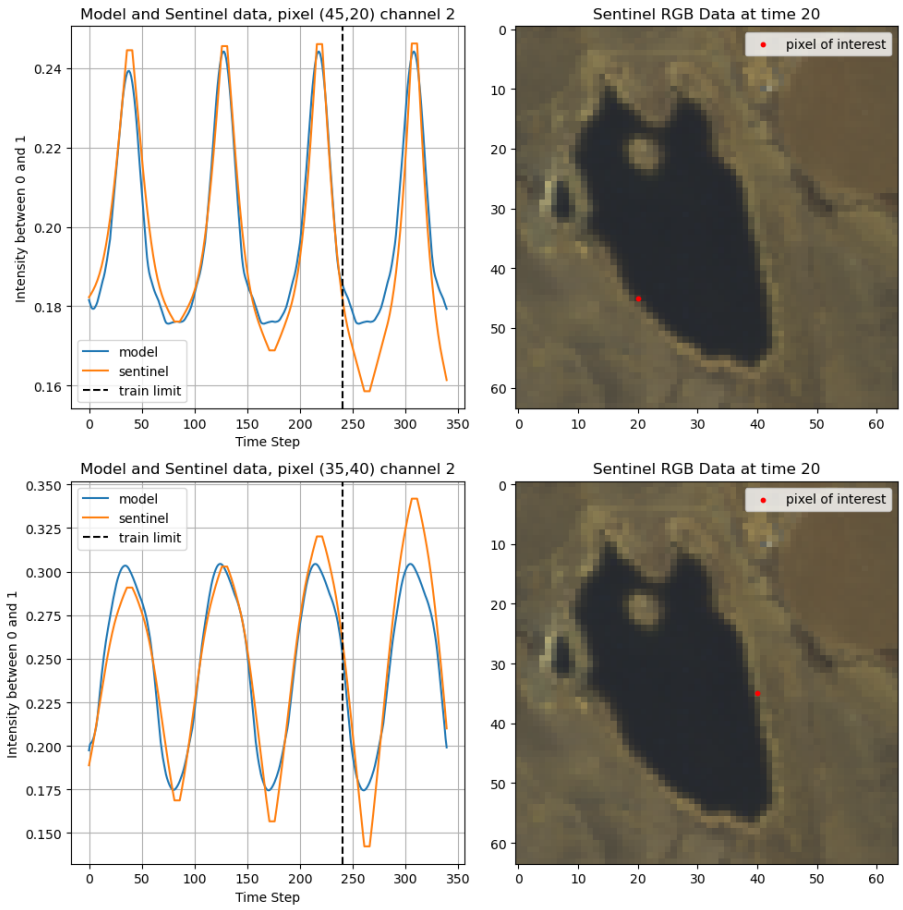

Figure C.7 – Predictions performed by an FNO model on the synthetic lake dataset, for particular pixels located on areas that alternate between ground and water. The FNO manages to infer the dynamical pattern by leveraging the spatial dimension, although only one image is used as the input to the model.

in figure C.6. Its fundamental difference with the datasets of the Fontainebleau and Orléans areas that were described in section 1.4.2 is that, for some pixels, it is impossible to predict the long-term dynamics when only knowing an initial reflectance vector with no spatial information. Indeed, some pixels close to the shores are alternately composed of ground or water, and this alternation can only be predicted when leveraging either many time steps of data or the spatial structure of a single image. Thus, we have indeed observed that the spatio-temporal FNO method indeed performs much better when forecasting this dataset than the pixelwise KAE model, contrarily to the results observed for a dataset with more spatial stationarity in table C.1. Sample predictions of an FNO model on pixels of interest are shown on figure C.7.

Appendix D

Bibliography

[1] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han, "Object detection in optical remote sensing images: A survey and a new benchmark," *ISPRS journal of photogrammetry and remote sensing*, vol. 159, pp. 296–307, 2020.

[2] I. Kotaridis and M. Lazaridou, "Remote sensing image segmentation advances: A meta-analysis," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 173, pp. 309–322, 2021.

[3] A. Asokan and J. Anitha, "Change detection techniques for remote sensing applications: A survey," *Earth Science Informatics*, vol. 12, pp. 143–160, 2019.

[4] R. Fablet, P. H. Viet, and R. Lguensat, "Data-driven models for the spatio-temporal interpolation of satellite-derived SST fields," *IEEE Transactions on Computational Imaging*, vol. 3, no. 4, pp. 647–657, 2017.

[5] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[7] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[8] M. A. Wulder, D. P. Roy, V. C. Radeloff, T. R. Loveland, M. C. Anderson, D. M. Johnson, S. Healey, Z. Zhu, T. A. Scambos, N. Pahlevan *et al.*, "Fifty years of Landsat science and impacts," *Remote Sensing of Environment*, vol. 280, p. 113195, 2022.

[9] C. Xu, X. Du, X. Fan, G. Giuliani, Z. Hu, W. Wang, J. Liu, T. Wang, Z. Yan, J. Zhu *et al.*, "Cloud-based storage and computing for remote sensing big data: a technical review," *International Journal of Digital Earth*, vol. 15, no. 1, pp. 1417–1445, 2022.

[10] E. Chuvieco, I. Aguado, J. Salas, M. García, M. Yebra, and P. Oliva, "Satellite remote sensing contributions to wildland fire science and management," *Current Forestry Reports*, vol. 6, pp. 81–96, 2020.

[11] T. T. Nguyen, T. D. Hoang, M. T. Pham, T. T. Vu, T. H. Nguyen, Q.-T. Huynh, and J. Jo, "Monitoring agriculture areas with satellite images and deep learning," *Applied Soft Computing*, vol. 95, p. 106565, 2020.

[12] W. Turner, C. Rondinini, N. Pettorelli, B. Mora, A. K. Leidner, Z. Szantoi, G. Buchanan, S. Dech, J. Dwyer, M. Herold *et al.*, "Free and open-access satellite data are key to biodiversity conservation," *Biological Conservation*, vol. 182, pp. 173–176, 2015.

[13] F. Liu, D. Chen, Z. Guan, X. Zhou, J. Zhu, Q. Ye, L. Fu, and J. Zhou, "Remoteclip: A vision language foundation model for remote sensing," *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

[14] D. Hong, B. Zhang, X. Li, Y. Li, C. Li, J. Yao, N. Yokoya, H. Li, P. Ghamisi, X. Jia *et al.*, "SpectralGPT: Spectral remote sensing foundation model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[15] I. Dumeur, S. Valero, and J. Inglada, "Paving the way toward foundation models for irregular and unaligned satellite image time series," *arXiv preprint arXiv:2407.08448*, 2024.

[16] O. Maler, "A unified approach for studying discrete and continuous dynamical systems," in *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)*, vol. 2. IEEE, 1998, pp. 2083–2088.

[17] D. Whitley, "Discrete dynamical systems in dimensions one and two," *Bulletin of the London Mathematical Society*, vol. 15, no. 3, pp. 177–217, 1983.

[18] G. Butler and P. Waltman, "Persistence in dynamical systems," *Journal of Differential Equations*, vol. 63, no. 2, pp. 255–263, 1986.

[19] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of atmospheric sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[20] L. Petzold, "Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations," *SIAM journal on scientific and statistical computing*, vol. 4, no. 1, pp. 136–148, 1983.

[21] N. J. Higham, "Functions of matrices," 2014.

[22] H. E. Haber, "Notes on the matrix exponential and logarithm," *Santa Cruz Institute for Particle Physics, University of California: Santa Cruz, CA, USA*, 2018.

[23] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

[24] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *Journal of Computational Dynamics*, vol. 1, no. 2, pp. 391–421, 2014.

[25] M. Wu, C. Zhang, J. Liu, L. Zhou, and X. Li, "Towards accurate high resolution satellite image semantic segmentation," *Ieee Access*, vol. 7, pp. 55 609–55 619, 2019.

[26] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, "Vehicle detection in satellite images by hybrid deep convolutional neural networks," *IEEE Geoscience and remote sensing letters*, vol. 11, no. 10, pp. 1797–1801, 2014.

[27] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[28] L. Loncan, L. B. De Almeida, J. M. Bioucas-Dias, X. Briottet, J. Chanussot, N. Dobigeon, S. Fabre, W. Liao, G. A. Licciardi, M. Simoes *et al.*, "Hyperspectral pansharpening: A review," *IEEE Geoscience and remote sensing magazine*, vol. 3, no. 3, pp. 27–46, 2015.

[29] L. Baetens, C. Desjardins, and O. Hagolle, "Validation of Copernicus Sentinel-2 cloud masks obtained from MAJA, Sen2Cor, and FMask processors using reference cloud masks generated with a supervised active learning procedure," *Remote Sensing*, vol. 11, no. 4, p. 433, 2019.

[30] G. P. Cressman, "An operational objective analysis system," *Monthly Weather Review*, vol. 87, no. 10, pp. 367–374, 1959.

[31] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for Koopman operators of nonlinear dynamical systems," in *American Control Conference (ACC)*. IEEE, 2019, pp. 4832–4839.

[32] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, p. 4950, 2018.

[33] J. Morton, A. Jameson, M. J. Kochenderfer, and F. Witherden, "Deep dynamical modeling and control of unsteady fluid flows," *NeurIPS*, vol. 31, 2018.

[34] O. Azencot, N. B. Erichson, V. Lin, and M. Mahoney, "Forecasting sequential data using consistent Koopman autoencoders," in *ICML*. PMLR, 2020, pp. 475–485.

[35] F. Fan, B. Yi, D. Rye, G. Shi, and I. R. Manchester, "Learning stable Koopman embeddings," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 2742–2747.

[36] P. Bevanda, M. Beier, S. Kerz, A. Lederer, S. Sosnowski, and S. Hirche, "Diffeomorphically learning stable Koopman operators," *IEEE Control Systems Letters*, vol. 6, pp. 3427–3432, 2022.

[37] J. Zhang, Q. Zhu, and W. Lin, "Learning Hamiltonian neural Koopman operator and simultaneously sustaining and discovering conservation laws," *Physical Review Research*, vol. 6, no. 1, p. L012031, 2024.

[38] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, "Modern Koopman theory for dynamical systems," *arXiv preprint arXiv:2102.12086*, 2021.

[39] B. O. Koopman, "Hamiltonian systems and transformation in Hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[40] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, pp. 1307–1346, 2015.

[41] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis, "A kernel-based approach to data-driven Koopman spectral analysis," *arXiv preprint arXiv:1411.2260*, 2014.

[42] Y. Kawahara, "Dynamic mode decomposition with reproducing kernels for Koopman spectral analysis," *NeurIPS*, vol. 29, 2016.

[43] V. Kostic, P. Novelli, A. Maurer, C. Ciliberto, L. Rosasco, and M. Pontil, "Learning dynamical systems via Koopman operator regression in reproducing kernel Hilbert spaces," *NeurIPS*, vol. 35, pp. 4017–4031, 2022.

[44] B. Dufée, B. Hug, E. Memin, and G. Tissot, "Ensemble forecasts in reproducing kernel Hilbert space family: dynamical systems in Wonderland," *arXiv preprint arXiv:2207.14653*, 2022.

[45] P. Bevanda, M. Beier, A. Lederer, S. Sosnowski, E. Hüllermeier, and S. Hirche, "Koopman kernel regression," *arXiv:2305.16215*, 2023.

[46] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis, "Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 10, p. 103111, 2017.

[47] A. Surana and A. Banaszuk, "Linear observer synthesis for nonlinear systems using Koopman operator framework," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 716–723, 2016.

[48] M. Fathi, C. Gehring, J. Pilault, D. Kanaa, P.-L. Bacon, and R. Goroshin, "Course correcting Koopman representations," *arXiv preprint arXiv:2310.15386*, 2023.

[49] Y. Meng, J. Huang, and Y. Qiu, "Koopman operator learning using invertible neural networks," *Journal of Computational Physics*, vol. 501, p. 112795, 2024.

[50] Y. Jin, L. Hou, S. Zhong, H. Yi, and Y. Chen, "Invertible Koopman network and its application in data-driven modeling for dynamic systems," *Mechanical Systems and Signal Processing*, vol. 200, p. 110604, 2023.

[51] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[52] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using Real NVP," *arXiv preprint arXiv:1605.08803*, 2016.

[53] T. Teshima, I. Ishikawa, K. Tojo, K. Oono, M. Ikeda, and M. Sugiyama, "Coupling-based invertible neural networks are universal diffeomorphism approximators," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3362–3373, 2020.

[54] C. Wehmeyer and F. Noé, "Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics," *The Journal of chemical physics*, vol. 148, no. 24, 2018.

[55] S. E. Otto and C. W. Rowley, "Linearly recurrent autoencoder networks for learning dynamics," *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 1, pp. 558–593, 2019.

[56] Y. Li, H. He, J. Wu, D. Katabi, and A. Torralba, "Learning compositional Koopman operators for model-based control," in *ICLR*, 2019.

[57] N. B. Erichson, M. Muehlebach, and M. W. Mahoney, "Physics-informed autoencoders for Lyapunov-stable fluid flow prediction," *arXiv preprint arXiv:1905.10866*, 2019.

[58] A. M. Lyapunov, "The general problem of the stability of motion," *International journal of control*, vol. 55, no. 3, pp. 531–534, 1992.

[59] V. Simoncini, "Computational methods for linear matrix equations," *siam REVIEW*, vol. 58, no. 3, pp. 377–441, 2016.

[60] R. A. Nelson and M. Olsson, "The pendulum-rich physics from a simple system," *Am. J. Phys*, vol. 54, no. 2, pp. 112–121, 1986.

[61] Y. Chen and M. J. Wainwright, "Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees," *arXiv preprint arXiv:1509.03025*, 2015.

[62] T. Bröcker and T. Tom Dieck, *Representations of compact Lie groups*. Springer Science & Business Media, 2013, vol. 98.

[63] S. Pan and K. Duraisamy, "Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability," *SIAM Journal on Applied Dynamical Systems*, vol. 19, no. 1, pp. 480–509, 2020.

[64] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for non-linear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[65] A. Frion, L. Drumetz, M. Dalla Mura, G. Tochon, and A. Aïssa-El-Bey, "Leveraging neural Koopman operators to learn continuous representations of dynamical systems from scarce data," in *ICASSP*. IEEE, 2023, pp. 1–5.

[66] A. Frion, L. Drumetz, M. Dalla Mura, G. Tochon, and A. Aïssa-El-Bey, "Neural koopman prior for data assimilation," *IEEE Transactions on Signal Processing*, 2024.

[67] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *International conference on machine learning*. PMLR, 2016, pp. 1120–1128.

[68] E. Vorontsov, C. Trabelsi, S. Kadoury, and C. Pal, "On orthogonality and learning recurrent networks with long term dependencies," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3570–3578.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[70] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence: proceedings of a symposium held at the University of Warwick 1979/80*. Springer, 2006, pp. 366–381.

[71] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[72] S. Le Clainche and J. M. Vega, "Higher order dynamic mode decomposition," *SIAM Journal on Applied Dynamical Systems*, vol. 16, no. 2, pp. 882–925, 2017.

[73] B. R. Noack, K. Afanasiev, M. Morzyński, G. Tadmor, and F. Thiele, "A hierarchy of low-dimensional models for the transient and post-transient cylinder wake," *Journal of Fluid Mechanics*, vol. 497, pp. 335–363, 2003.

[74] L. Wingate, J. Ogée, E. Cremonese, G. Filippa, T. Mizunuma, M. Migliavacca, C. Moisy, M. Wilkinson, C. Moureaux, G. Wohlfahrt *et al.*, "Interpreting canopy development and physiology using a European phenology camera network at flux sites," *Biogeosciences*, vol. 12, no. 20, pp. 5995–6015, 2015.

[75] Y. Takeuchi, *Global dynamical properties of Lotka-Volterra systems*. World Scientific, 1996.

[76] C. R. Vogel, *Computational methods for inverse problems*. SIAM, 2002.

[77] I. M. Navon, "Data assimilation for numerical weather prediction: a review," *Data assimilation for atmospheric, oceanic and hydrologic applications*, pp. 21–65, 2009.

[78] X. Jin, L. Kumar, Z. Li, H. Feng, X. Xu, G. Yang, and J. Wang, "A review of data assimilation of remote sensing and crop models," *European Journal of Agronomy*, vol. 92, pp. 141–152, 2018.

[79] K. H. Cho, Y. Pachepsky, M. Ligaray, Y. Kwon, and K. H. Kim, "Data assimilation in surface water quality modeling: A review," *Water Research*, vol. 186, p. 116307, 2020.

[80] Q. Xu, B. Li, R. E. McRoberts, Z. Li, and Z. Hou, "Harnessing data assimilation and spatial autocorrelation for forest inventory," *Remote Sensing of Environment*, vol. 288, p. 113488, 2023.

[81] D. Stammer, M. Balmaseda, P. Heimbach, A. Köhl, and A. Weaver, "Ocean data assimilation in support of climate applications: Status and perspectives," *Annual review of marine science*, vol. 8, no. 1, pp. 491–518, 2016.

[82] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[83] G. Evensen, "Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics," *Journal of Geophysical Research: Oceans*, vol. 99, no. C5, pp. 10 143–10 162, 1994.

[84] ——, "The ensemble Kalman filter: Theoretical formulation and practical implementation," *Ocean dynamics*, vol. 53, pp. 343–367, 2003.

[85] G. D. Granzow, "A tutorial on adjoint methods and their use for data assimilation in glaciology," *Journal of Glaciology*, vol. 60, no. 221, pp. 440–446, 2014.

[86] R. N. Bannister, "A review of operational methods of variational and ensemble-variational data assimilation," *Quarterly Journal of the Royal Meteorological Society*, vol. 143, no. 703, pp. 607–633, 2017.

[87] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[88] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: a system for Large-Scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[89] J. Revels, M. Lubin, and T. Papamarkou, "Forward-mode automatic differentiation in Julia," *arXiv preprint arXiv:1607.07892*, 2016.

[90] R. A. Willoughby, "Solutions of Ill-Posed Problems (A. N. Tikhonov and V. Y. Arsenin)," *SIAM Review*, vol. 21, no. 2, pp. 266–267, 1979.

[91] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[92] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[93] T. Burgess and R. Webster, "Optimal interpolation and isarithmic mapping of soil properties: I the semi-variogram and punctual kriging," *Journal of soil science*, vol. 31, no. 2, pp. 315–331, 1980.

[94] C. K. Wikle and L. M. Berliner, "A Bayesian tutorial for data assimilation," *Physica D: Nonlinear Phenomena*, vol. 230, no. 1-2, pp. 1–16, 2007.

[95] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[96] M. Gelbrecht, A. White, S. Bathiany, and N. Boers, "Differentiable programming for Earth system modeling," *Geoscientific Model Development*, vol. 16, no. 11, pp. 3123–3135, 2023.

[97] R. Fablet, L. Drumetz, and F. Rousseau, "Joint learning of variational representations and solvers for inverse problems with partially-observed data," *arXiv:2006.03653*, 2020.

[98] M. Nonnenmacher and D. S. Greenberg, "Deep emulators for differentiation, forecasting, and parametrization in Earth science simulators," *Journal of Advances in Modeling Earth Systems*, vol. 13, no. 7, p. e2021MS002554, 2021.

[99] A. Farchi, P. Laloyaux, M. Bonavita, and M. Bocquet, "Using machine learning to correct model error in data assimilation and forecast applications," *Quarterly Journal of the Royal Meteorological Society*, vol. 147, no. 739, pp. 3067–3084, 2021.

[100] S. Cheng *et al.*, "Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review," *arXiv:2303.10462*, 2023.

[101] C. Fefferman, S. Mitter, and H. Narayanan, "Testing the manifold hypothesis," *Journal of the American Mathematical Society*, vol. 29, no. 4, pp. 983–1049, 2016.

[102] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[103] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[104] A. Frion, L. Drumetz, G. Tochon, M. D. Mura, and A. A. E. Bey, "Koopman ensembles for probabilistic time series forecasting," in *2024 32nd European Signal Processing Conference (EUSIPCO)*, 2024.

[105] K. Haynes, R. Lagerquist, M. McGraw, K. Musgrave, and I. Ebert-Uphoff, "Creating and evaluating uncertainty estimates with neural networks for environmental-science applications," *Artificial Intelligence for the Earth Systems*, vol. 2, no. 2, p. 220061, 2023.

[106] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[107] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society: series B (methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[108] D. Freedman and P. Diaconis, "On the histogram as a density estimator: L2 theory," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 57, no. 4, pp. 453–476, 1981.

[109] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," *Biostatistics & Epidemiology*, vol. 1, no. 1, pp. 161–187, 2017.

[110] P. Zhao and L. Lai, "Analysis of KNN density estimation," *IEEE Transactions on Information Theory*, vol. 68, no. 12, pp. 7971–7995, 2022.

[111] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 448–455.

[112] A. Damianou and N. D. Lawrence, "Deep Gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2013, pp. 207–215.

[113] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[114] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[115] F. Draxler, S. Wahl, C. Schnörr, and U. Köthe, "On the universality of coupling-based normalizing flows," *arXiv preprint arXiv:2402.06578*, 2024.

[116] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[117] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[118] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[119] "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[120] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[121] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[122] B. Dai, S. Fidler, R. Urtasun, and D. Lin, "Towards diverse and natural image descriptions via a conditional GAN," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2970–2979.

[123] E. A. Barnes and R. J. Barnes, "Controlled abstention neural networks for identifying skillful predictions for regression problems," *Journal of Advances in Modeling Earth Systems*, vol. 13, no. 12, p. e2021MS002575, 2021.

[124] B. Van Schaeybroeck and S. Vannitsem, "Ensemble post-processing using member-by-member approaches: theoretical aspects," *Quarterly Journal of the Royal Meteorological Society*, vol. 141, no. 688, pp. 807–818, 2015.

[125] S. Veldkamp, K. Whan, S. Dirksen, and M. Schmeits, "Statistical postprocessing of wind speed forecasts using convolutional neural networks," *Monthly Weather Review*, vol. 149, no. 4, pp. 1141–1152, 2021.

[126] B. Schulz and S. Lerch, "Machine learning methods for postprocessing ensemble forecasts of wind gusts: A systematic comparison," *Monthly Weather Review*, vol. 150, no. 1, pp. 235–257, 2022.

[127] J. B. Bremnes, "Ensemble postprocessing using quantile function regression based on neural networks and Bernstein polynomials," *Monthly Weather Review*, vol. 148, no. 1, pp. 403–414, 2020.

[128] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[129] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, "Hands-on Bayesian neural networks—a tutorial for deep learning users," *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 29–48, 2022.

[130] J. E. Matheson and R. L. Winkler, "Scoring rules for continuous probability distributions," *Management science*, vol. 22, no. 10, pp. 1087–1096, 1976.

[131] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[132] L. Delle Monache, F. A. Eckel, D. L. Rife, B. Nagarajan, and K. Searight, "Probabilistic weather prediction with an analog ensemble," *Monthly Weather Review*, vol. 141, no. 10, pp. 3498–3516, 2013.

[133] O. Talagrand, "Evaluation of probabilistic prediction systems," in *Workshop Proceedings" Workshop on Predictability", 20-22 October 1997, ECMWF, Reading, UK*, 1999.

[134] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[135] ——, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[136] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra, "Why m heads are better than one: Training a diverse ensemble of deep networks," *arXiv preprint arXiv:1511.06314*, 2015.

[137] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[138] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*. Springer, 2000, pp. 1–15.

[139] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *International conference on machine learning*. PMLR, 2015, pp. 1861–1869.

[140] S. Jain, G. Liu, J. Mueller, and D. Gifford, "Maximizing overall diversity for improved uncertainty estimates in deep ensembles," in *Proceedings of the AAAI conference*, vol. 34, no. 04, 2020, pp. 4264–4271.

[141] A. M. Mood, *Introduction to the Theory of Statistics*. McGraw-hill, 1950.

[142] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[143] N. Geneva and N. Zabaras, "Transformers for modeling physical systems," *Neural Networks*, vol. 146, pp. 272–289, 2022.

[144] I. Naiman, N. B. Erichson, P. Ren, M. W. Mahoney, and O. Azencot, "Generative modeling of regular and irregular time series data via Koopman VAEs," *arXiv preprint arXiv:2310.02619*, 2023.

[145] K. Cho, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[146] M. Han, J. Euler-Rolle, and R. K. Katzschmann, "Desko: Stability-assured robust control with a deep stochastic Koopman operator," in *International Conference on Learning Representations*, 2021.

[147] W. Rudin *et al.*, *Principles of mathematical analysis*. McGraw-hill New York, 1964, vol. 3.

[148] L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, and X. Alameda-Pineda, "Dynamical variational autoencoders: A comprehensive review," *arXiv preprint arXiv:2008.12595*, 2020.

[149] M. Alami Machichi, l. E. mansouri, Y. Imani, O. Bourja, O. Lahlou, Y. Zennayi, F. Bourzeix, I. Hanadé Houmma, and R. Hadria, "Crop mapping using supervised machine learning and deep learning: a systematic literature review," *International Journal of Remote Sensing*, vol. 44, no. 8, pp. 2717–2753, 2023.

[150] M. Rußwurm, C. Pelletier, M. Zollner, S. Lefèvre, and M. Körner, "Breizhcrops: A time series dataset for crop type mapping," *arXiv preprint arXiv:1905.11893*, 2019.

[151] G. Weikmann, C. Paris, and L. Bruzzone, "Timesen2crop: A million labeled samples dataset of Sentinel 2 image time series for crop-type classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 4699–4708, 2021.

[152] A. Frion, L. Drumetz, G. Tochon, M. Dalla Mura, and A. Aïssa-El-Bey, "Learning Sentinel-2 reflectance dynamics for data-driven assimilation and forecasting," in *2023 31st European Signal Processing Conference (EUSIPCO)*. IEEE, 2023, pp. 1390–1394.

[153] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen netzen," *Diploma, Technische Universität München*, vol. 91, no. 1, p. 31, 1991.

[154] Y. Bengio, P. Frasconi, and P. Simard, "The problem of learning long-term dependencies in recurrent networks," in *IEEE international conference on neural networks*. IEEE, 1993, pp. 1183–1188.

[155] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. Pmlr, 2013, pp. 1310–1318.

[156] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[157] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units," *arXiv preprint arXiv:1504.00941*, 2015.

[158] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[159] M. Henaff, A. Szlam, and Y. LeCun, "Recurrent orthogonal networks and long-memory tasks," in *International Conference on Machine Learning*. PMLR, 2016, pp. 2034–2042.

[160] S. Wisdom, T. Powers, J. Hershey, J. Le Roux, and L. Atlas, "Full-capacity unitary recurrent neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[161] Z. Mhammedi, A. Hellicar, A. Rahman, and J. Bailey, "Efficient orthogonal parametrisation of recurrent neural networks using householder reflections," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2401–2409.

[162] M. Lezcano-Casado and D. Martınez-Rubio, "Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3794–3803.

[163] H. D. Tagare, "Notes on optimization on Stiefel manifolds," *Yale University, New Haven*, 2011.

[164] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," *Journal of the ACM (JACM)*, vol. 5, no. 4, pp. 339–342, 1958.

[165] B. Chang, M. Chen, E. Haber, and E. H. Chi, "AntisymmetricRNN: A dynamical system view on recurrent neural networks," *arXiv preprint arXiv:1902.09689*, 2019.

[166] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE transactions on knowledge and data engineering*, vol. 35, no. 1, pp. 857–876, 2021.

[167] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *arXiv preprint arXiv:1803.07728*, 2018.

[168] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1422–1430.

[169] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European conference on computer vision*. Springer, 2016, pp. 69–84.

[170] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[171] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[172] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand *et al.*, "Findings of the 2014 workshop on statistical machine translation," in *Proceedings of the ninth workshop on statistical machine translation*, 2014, pp. 12–58.

[173] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[174] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[175] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[176] C. Lacrouts, A. Frion, and L. Drumetz, "Apprentissage d'un modèle dynamique pour l'interpolationspatio-temporelle d'images satellitaires multi-spectrales," IMT Atlantique, Tech. Rep., Oct. 2024. https://imt-atlantique.hal.science/hal-04710544

[177] P. Maragos, "Algebraic and PDE approaches for lattice scale-spaces with global constraints," *International Journal of Computer Vision*, vol. 52, pp. 121–137, 2003.

Appendix E

List of publications associated to the thesis work

[1] A. Frion, L. Drumetz, M. Dalla Mura, G. Tochon, and A. Aïssa-El-Bey, "Leveraging neural Koopman operators to learn continuous representations of dynamical systems from scarce data," in *ICASSP*. IEEE, 2023, pp. 1–5.

[2] A. Frion, L. Drumetz, G. Tochon, M. Dalla Mura, and A. Aïssa-El-Bey, "Learning Sentinel-2 reflectance dynamics for data-driven assimilation and forecasting," in *2023 31st European Signal Processing Conference (EUSIPCO)*. IEEE, 2023, pp. 1390–1394.

[3] A. Frion, L. Drumetz, M. Dalla Mura, G. Tochon, and A. Aïssa-El-Bey, "Assimilation de données variationnelle de séries temporelles d'images Sentinel-2 avec un modèle dynamique auto-supervisé," in *GRETSI 2023-XXIXème Colloque Francophone de Traitement du Signal et des Images*, 2023.

[4] ——, "Neural Koopman prior for data assimilation," *IEEE Transactions on Signal Processing*, 2024.

[5] A. Frion, L. Drumetz, G. Tochon, M. D. Mura, and A. A. E. Bey, "Koopman ensembles for probabilistic time series forecasting," in *2024 32nd European Signal Processing Conference (EUSIPCO)*, 2024.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Apprentissage de dynamiques dans les séries temporelles d'images satellites multispectrales

Mot clés : Télédétection, Systèmes dynamiques, Opérateur de Koopman, Assimilation de données, Quantification d'incertitudes

Résumé : Via les missions d'observation relevant, par exemple, du programme Copernicus de l'Union européenne, de très grandes quantités d'images satellites multispectrales de la surface de la Terre sont aujourd'hui disponibles. Ces données permettent, entre autres, de surveiller l'état d'environnements tels que les forêts, glaciers et océans ou de faciliter l'organisation des secours lors de catastrophes naturelles telles que les inondations et incendies. Ainsi, les méthodes de traitement des données issues de satellites, souvent basées sur des réseaux de neurones artificiels, sont d'un grand intérêt pour ces applications. Cependant, entraîner ces modèles avec un paradigme d'apprentissage supervisé requiert l'annotation de grandes quantités de données pour chacune des applications considérées, et la prise en compte de séries temporelles d'images (plutôt que d'images seules) reste difficile.

Dans cette thèse, nous proposons d'adopter une approche auto-supervisée, dans le but d'apprendre un modèle dynamique d'évolution à partir de données non annotées, disponibles publiquement et massivement. Nous proposons différentes méthodes pour utiliser un tel modèle dynamique pour résoudre différentes tâches pratiques, notamment en l'intégrant dans un coût variationnel d'assimilation de données. Bien que nous nous concentrions sur des séries temporelles d'images de forêts prises par les satellites Sentinel-2, nous avons pour ambition de développer des méthodes applicables à divers types de séries temporelles, y compris en dehors du champ de l'imagerie satellitaire.

Title: Learning the dynamics of multispectral satellite image time series

Keywords: Remote sensing, Dynamical systems, Koopman operator, Data assimilation, Uncertainty quantification

Abstract: Thanks to the Earth observation missions such as the Copernicus programme of the European Union, vast amounts of satellite images from the surface of the Earth are now available. These data enable, among other applications, to monitor the state of environments such as forests, glaciers and oceans or to facilitate the coordination of rescues in case of natural disasters such as floods and forest fires. Thus, the methods for processing satellite images, often based on artificial neural networks, are of great interest for these applications. However, training these models with a supervised learning approach requires annotating large quantities of data for each of the applications considered, and processing time series of images (rather than isolated images) is still a difficulty.

In this thesis, we propose to adopt a self-supervised approach, aiming at learning a dynamical model of evolution from unlabelled data, which are publicly and massively available. We introduce several methods for using such a dynamical model to solve various practical tasks, notably by integrating it in a variational data assimilation cost. Although we focus on time series of forest images taken by the Sentinel-2 satellites, we ambition to develop methods that are applicable to various types of time series, including outside the field of satellite imagery.