# Metamodel and bayesian approaches for dynamic systems

Max Cohen

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAS003

Thèse de doctorat

# Métamodèles et approches bayésiennes pour les systèmes dynamiques

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Telecom SudParis

École doctorale n°574 Ecole doctorale de mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques appliquées

Thèse présentée et soutenue à Palaiseau, le 30 mars 2023, par

**MAX COHEN**

Composition du Jury :

| | |
|---|---|
| Wojciech Pieczynski<br>Professeur, Télécom SudParis, CITI | Président |
| Víctor Elvira<br>Professeur, School of Mathematics, University of Edinburgh | Rapporteur |
| Stéphane Lecœuche<br>Professeur, IMT Lille Douai | Rapporteur |
| Marie-Pierre Etienne<br>Enseignante-chercheuse, Agrocampus Ouest | Examinatrice |
| Sylvain Le Corff<br>Professeur, Télécom SudParis, CITI | Directeur de thèse |
| Maurice Charbit<br>Accenta | Co-encadrant de thèse |
| Gilles Nozières<br>Accenta | Invité |

# Remerciements

Je tiens à remercier tous mes collègues, amis et proches, qui m'ont soutenu et encouragé durant cette thèse. Merci aux membres du jury, et en particulier aux rapporteurs, pour leur attention et leur investissement dans ce projet. Merci à tous ceux que j'oublie, et qui y furent essentiels.

Je voudrais d'abord remercier Sylvain, le chef d'orchestre de cette grande aventure, qui a remué ciel et terre pour permettre de lancer ce projet de thèse, et qui m'a accordé sa confiance pour le mener à bien. Je lui suis reconnaissant pour sa disponibilité indéfectible, les heures passées à m'aider et me guider, bref à son soutien bienveillant et sans relâche tout au long de ces dernières années. Merci à Maurice de m'avoir si bien encadré et de m'avoir transmis tant de connaissances. Rien n'a été aussi agréable que nos discussions endiablées. Merci aussi à Charles et Guillaume, avec qui j'ai eu le plaisir de travailler lors d'une collaboration exceptionnelle, qui n'aurait pu se réaliser sans eux.

Je tiens ensuite à remercier Gilles, et toute l'équipe d'Accenta, qui m'ont accueilli, formé et encouragé. Ils m'ont permis de participer à ce projet visionnaire pour lequel j'ai été très heureux de m'investir. Merci beaucoup à Marius, qui lui aussi s'est battu pour permettre le lancement de cette thèse, et plus généralement à toute l'équipe du laboratoire Artémis avec qui j'ai pu échanger tout au long de ces dernières années. J'ai beaucoup apprécié nos débats sur l'état de l'art. Merci en particulier à Nicolas, pour nos discussions approfondies sur mes sujets de thèse, entre autres, et sans qui je risquais de boucler mon manuscrit en 2022. Je voudrais remercier Marc, qui m'a guidé dans le monde de la recherche, m'a encouragé à poursuivre mes ambitions dans le domaine, et accessoirement m'a introduit à Sylvain, mon directeur de thèse. C'est en grande partie grâce à son soutien que j'ai survécu à cette expérience, on fêtera ça à la yourte. Merci aussi à tous les amis qui m'ont permis de prendre du recul pendant mes recherches, et avec qui j'ai apprécié chaque minute, en particulier merci à Romain, Jonathan, Sebastien, Serhat, Louise, Pierre.

Un grand merci à Leslie et Lilou pour leur soutien quotidien. Merci de m'avoir épaulé, conseillé et encouragé pendant les moments difficiles, et d'avoir si bien su partager avec moi nos petites victoires. Merci enfin à toute ma famille, à mes parents sans qui je n'aurais pas pu arriver jusque-là, à mon frère, ma sœur et mes grands parents, avec qui j'ai passé de merveilleux moments.

# Table des matières

# Chapitre 1

# Introduction (version française)

## 1.1 Oze-Energies

Oze-Energies, nouvellement Accenta, est une entreprise française spécialisée dans l'optimisation de la consommation énergétique des bâtiments. Alors que la demande énergétique en chauffage, climatisation et ventilation n'a cessé d'augmenter depuis plusieurs décennies, réduire l'impact environnemental du parc immobilier, tout en maintenant une qualité de confort raisonnable, reste un problème complexe. Au travers de méthodes innovantes et durables, Oze-Energies vise à améliorer la qualité de l'air et le confort dans les bâtiments, tout en réduisant leur consommation, sans rénovation.

En 2009, le parc immobilier était responsable de 40% de la consommation énergétique française, et près d'un quart des émissions de gaz à effet de serre (Loi Grenelle [1], Figure 1.1). La stratégie la plus directe pour réduire cette consommation consiste à rénover l'isolation des vieux bâtiments, ce qui mène généralement à des travaux longs et coûteux. Malgré s'être fixé comme objectif de rénover 500 000 bâtiments par an avant 2020 [2] [3], les résultats ne sont à ce jour pas satisfaisants, comme souligné par l'Ademe (Agence de la transition écologique) [4]. Selon la SNBC (Stratégie Nationale Bas Carbone), le nombre moyen de rénovations annuelles effectuées d'ici 2030 avoisinera les 370 000 pour la période 2015-2030. Oze-Energies a choisi une approche orthogonale en proposant aux gestionnaires de suivre des feuilles de route, produites sur mesure, afin de réduire la consommation de leur bâtiments. Cela conduit à une réduction de 25% en moyenne de la facture énergétique, sans nécessiter de travaux de rénovation.

La consommation énergétique d'un bâtiment peut être réduite en grande partie en jouant sur les paramètres de chauffage, ventilation et air conditionné (abrégé HVAC pour Heating Ventilation and Air Conditioning). Oze-Energies travaille principalement sur l'occupation dans le secteur tertiaire où les comportements sont bien connus (journées classiques de travail, pas d'occupation le week-end, etc.), cependant on peut toujours noter des différences significatives d'un bâtiment à l'autre dues à une grande variété de facteurs. On peut citer par exemple l'isolation, l'occupation du bâtiment ou encore l'utilisation de différents services de chauffage et climatisation. Ces différences représentent une des difficultés techniques majeures de notre approche, que nous détaillons dans ce document. Afin de modéliser précisément les échanges de chaleur de chaque bâtiment, Oze-Energies suit leur consommation et mesure la qualité de leur air de leur locaux en y installant des capteurs environnementaux, connectés à travers LoRA (un réseau public et sécurisé). Ces données sont regroupées dans une base de données centralisée. Les experts thermiciens de Oze-Energies peuvent alors les combiner avec leur connaissance de la thermique des

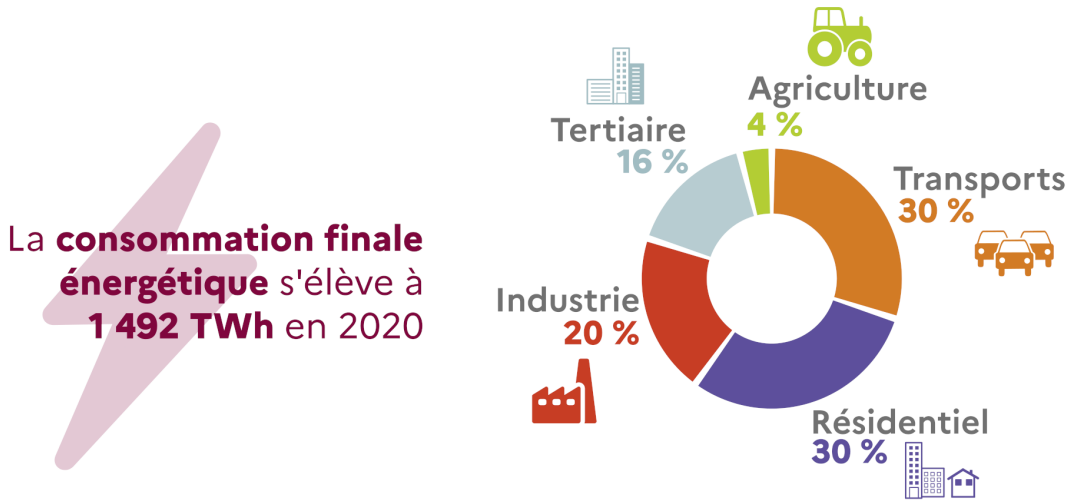
1. Loi Grenelle I, Article 3
2. Stratégie Nationale Bas-Carbone (SNBC)
3. Loi Grenelle I, Article 5
4. Hervé Lefebvre, directeur du département Climat de l'Ademe, 2019

bâtiments pour produire des feuilles de route qui détaillent un jeu de paramètres de HVAC permettant d'optimiser la consommation et d'assurer une bonne qualité de l'air. Avec une réduction de la facture énergétique de 25% en moyenne à confort constant, le coût de ce service est largement couvert par les économies réalisées : il s'agit du produit principal proposé par Oze-Energies, OPTIMZEN®.

Oze-Energies recherche des méthodes permettant de réduire le travail parfois répétitif de ses thermiciens : estimer les paramètres physiques des bâtiments, proposer de nouveaux scénarios de consommation, quantifier et comparer ces scénarios entre eux. Une solution que nous étudierons dans les chapitres suivants consiste à utiliser un simulateur numérique, configuré pour estimer automatiquement le meilleur jeu de paramètres pour un bâtiment durant une période donnée, en simulant et comparant un grand nombre de politiques. Ce logiciel expert, TRNSYS, permet de simuler les comportements complexes des bâtiments à partir d'une représentation schématisée, ainsi que de nombreux paramètres d'entrée comme l'occupation du bâtiment au cours de la période étudiée, les paramètres de HVAC ou encore les prévisions météorologiques. TRNSYS est d'abord calibré pour représenter au mieux un bâtiment précis, à travers des méthodes de Machine Learning, en utilisant les données historiques du bâtiment récoltées par les capteurs. Comme énoncé précédemment, cette tâche est d'autant plus complexe que la variété de sources de chaleur, de froid ou d'électricité augmente avec chaque nouveau client. Une fois calibré, nous pouvons utiliser TRNSYS pour optimiser la consommation, tout en maintenant un niveau de confort et une qualité de l'air raisonnables.

Oze-Energies a été fondée en 2014, couvre plus de 5 millions de mètres carrés, étalés sur 500 bâtiments, et fait aujourd'hui partie de l'entreprise Accenta.

FIGURE 1.1 – Consommation énergétique par principaux secteurs (France, 2020, www.statistiques.developpement-durable.gouv.fr)




## 1.2 Optimisation du confort et de la consommation

### 1.2.1 Capteurs, compteurs et données météorologiques

Pour comprendre le comportement d'un bâtiment, nous combinons différents types de données, souvent récupérées directement sur le site. La première étape de la production de feuilles de route consiste donc à installer des compteurs, qui transmettront aux bases de données de Oze-Energies à intervalles réguliers, en général toutes les 10 minutes. Ils peuvent être rangés en trois catégories :

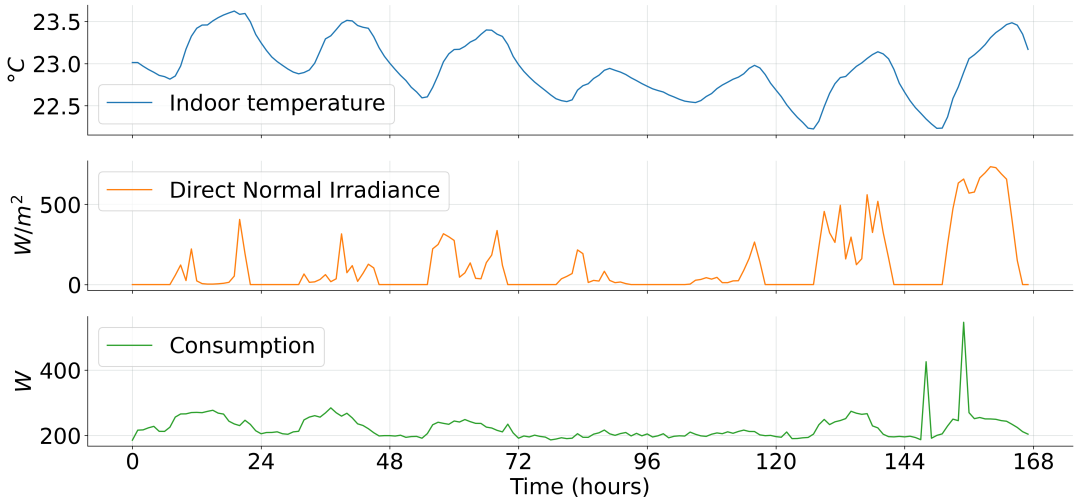
— Les **capteurs** sont installés par les experts thermiciens à divers emplacements du bâtiment, et récupèrent des données ambiantes, telles que le niveau de $CO_2$ , l'humidité relative ou encore la température intérieure. Parce que les données récoltées peuvent varier d'une zone du bâtiment à une autre, les capteurs présentent un problème d'agrégation complexe.

— Les **compteurs** enregistrent la consommation de chauffage, climatisation, ventilation et éventuellement d'électricité. En général, nous n'avons accès qu'à une variable pour chaque source de consommation, il n'y a donc pas de tâche d'agrégation nécessaire. La problématique principale liée aux compteurs réside dans la variété de fournisseurs d'énergie, de frigories et de calories : alors que certains nous transmettent une consommation horaire, d'autres ne nous donnent accès qu'à une donnée agrégée à la journée, ce qui implique un calcul de reconstruction non trivial. De plus, la gestion des erreurs d'enregistrement ou de transmission peut aussi varier, certains fournisseurs ignorant les données aberrantes, alors que d'autres les compensent sur les signaux suivants. Enfin, des fournisseurs différents peuvent répondre à un même besoin. Par exemple, il est possible de chauffer à partir de chauffages électriques, ou bien de se connecter à un réseau de chaud.

— Les **données météorologiques**, telles que la température extérieure, l'humidité ou les niveaux d'irradiation solaire, ont un impact important sur le comportement d'un bâtiment. Des données historiques correspondant à la zone étudiée sont disponibles pour les cinq dernières années. Pour réaliser les feuilles de route, on peut aussi utiliser les prévisions météorologiques, qui sont considérées précises jusqu'à une semaine dans le futur. Ces données ne sont pas récoltées directement par Oze-Energies, mais par Meteotest [5], une entreprise experte dans le domaine.

Un échantillon des données historiques est présenté dans la Figure 1.2.

FIGURE 1.2 – Nous avons tracé un échantillon d'une semaine de données historiques correspondant aux capteurs (température intérieure), compteurs (consommation) et données météorologiques (Direct Normal Irradiance).




### Simulation de l'évolution des bâtiments avec TRNSYS

Les simulateurs physiques permettant de simuler l'évolution d'un bâtiment reposent généralement sur des équations de propagation thermique. Ils permettent de prédire, en particulier, l'énergie consommée et la température intérieure à partir d'une description schématique du bâtiment, des matériaux de construction ainsi que de leur di-

5. https ://meteotest.ch/

mensions, des données météorologiques et des usages et paramètres de HVAC. Due à la complexité de la tâche à résoudre, seule une poignée de simulateurs performants existent à ce jour.

Par exemple, EnergyPlus est utilisé dans Shabunko et al. (2018) afin de définir trois types génériques de bâtiments, et ainsi de comparer les performances énergétiques de 400 bâtiments résidentiels. Dans Zhao et al. (2016), les auteurs proposent un framework de prédiction basé sur Matlab et EnergyPlus, qui leur permet d'optimiser la consommation d'énergie tout en respectant des préférences de confort individuelles. Dans Magnier and Haghighat (2010), les auteurs mettent en avant les performances de TRNSYS comme simulateur physique, tout comme ses limites en termes de coût de calcul : selon les auteurs, un processus d'optimisation complet aurait pris jusqu'à 10 ans s'il n'avaient pas remplacé TRNSYS par un modèle substitut, beaucoup plus rapide. Les auteurs de Bre et al. (2016) ont étudié l'optimisation d'une maison familiale en combinant EnergyPlus et l'algorithme d'optimisation NSGA-II, et proposent une analyse de la sensibilité en utilisant la méthode Morris screening. Une analyse similaire est présentée dans Recht et al. (2014), accompagnée d'une analyse d'incertitude sur le simulateur COMFIE. Ses performances sont évaluées sur un bâtiment passif.

Parmi ces outils disponibles, Oze-Energies a choisi TRNSYS pour l'étendue de ses fonctionnalités, sa flexibilité et sa popularité. Cependant, TRNSYS ne peut être exécuté que sur le système d'exploitation Windows, et nécessite le lancement d'une fenêtre pour chaque simulation, ce qui induit un temps de calcul d'au moins quelques secondes.

**Paramètres de simulation.** Dans ce paragraphe, nous détaillons les paramètres de simulation nécessaires à l'exécution de TRNSYS. Bien que la majorité de ces paramètres soient communs à tous les bâtiments, les variables présentées ci dessous correspondent à deux bâtiments spécifiques gérés par Oze-Energies, que nous introduisons plus tard dans la Section 3.2. Nous traçons un échantillon des données disponibles dans la Figure 1.3, à titre de comparaison un échantillon d'une simulation réalisée par TRNSYS est affiché dans la Figure 1.4. Merci de se référencer à l'Appendice A.1.2 pour une liste et une description exhaustive des variables utilisées. Pour modéliser de futurs bâtiments, de nouvelles variables pourraient être nécessaires.

- Les paramètres physiques du bâtiment, tels que la capacité thermique ou la taille des isolants, sont stockés dans un vecteur $\lambda \in \mathbb{R}^{d_\lambda}$, où $d_\lambda$ est le nombre de paramètres disponibles. $\Lambda$ est l'ensemble des jeux de paramètres physiques possibles.
- Les usages sont décrits à travers une série temporelle, qui contient les paramètres de HVAC à chaque heure. Soit $T$ la durée d'un échantillon en heures, nous définissons les usages comme $(\psi_k)_{1\leq k\leq T} \in \mathbb{R}^{d_\psi}$. Ici, $d_\psi$ est le nombre de variables individuelles, telle que les horaires d'activation de la ventilation, les températures de chauffage. $\Psi$ est l'ensemble des paramètres de HVAC possibles.
- De la même manière, l'occupation du bâtiment est encodée dans une série temporelle $(\delta_k)_{1\leq k\leq T} \in \mathbb{R}$ qui dénote la fraction des occupants présents dans le bâtiment à chaque heure. $\Delta$ est l'ensemble des horaires d'occupation possibles.
- Les conditions météorologiques sont agrégées dans la série temporelle $(\varphi_k)_{1\leq k\leq T} \in \mathbb{R}^{d_\varphi}$.
- Enfin, les consommations sont notées $(\zeta_k)_{1\leq k\leq T} \in \mathbb{R}^{d_\zeta}$, et la température intérieure $(\tau_k)_{1\leq k\leq T} \in \mathbb{R}$. $\mathcal{Z}$ (resp. $\mathcal{T}$) est l'ensemble des consommations possibles (resp. températures intérieures).

Dans la suite de ce document, nous désignons par $\varphi$ (resp. $\psi, \delta, \zeta, \tau$) les séries temporelles complètes $(\varphi_k)_{k=1}^T$ (resp. $(\psi)_{k=1}^T, (\delta)_{k=1}^T, (\zeta)_{k=1}^T, (\tau)_{k=1}^T$). Les simulateurs physiques, tels que TRNSYS, peuvent être décrits comme des fonctions prenant comme paramètres d'entrée les propriétés du bâtiment, ses usages et son occupation pour la période considérée, ainsi que les données météorologiques associées, afin de prédire (entre autres) la température intérieure horaire et les diverses consommations.

$$f_{\texttt{building}} : (\lambda, \psi, \delta, \varphi) \mapsto (\hat{\tau}, \hat{\zeta}).$$

FIGURE 1.3 – Sur cette figure, nous traçons un échantillon des données disponibles d'occupation $\delta$ et météorologiques $\varphi$.

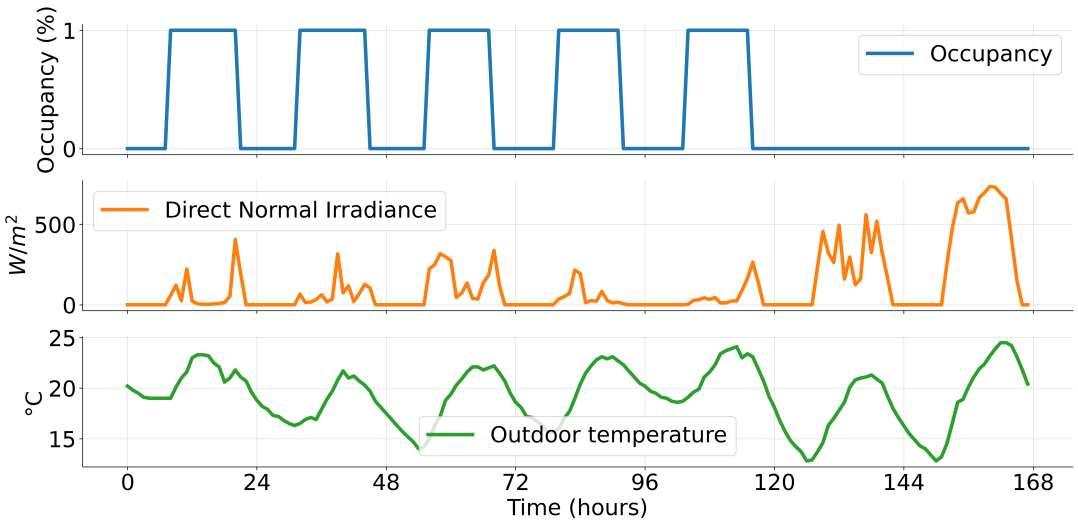
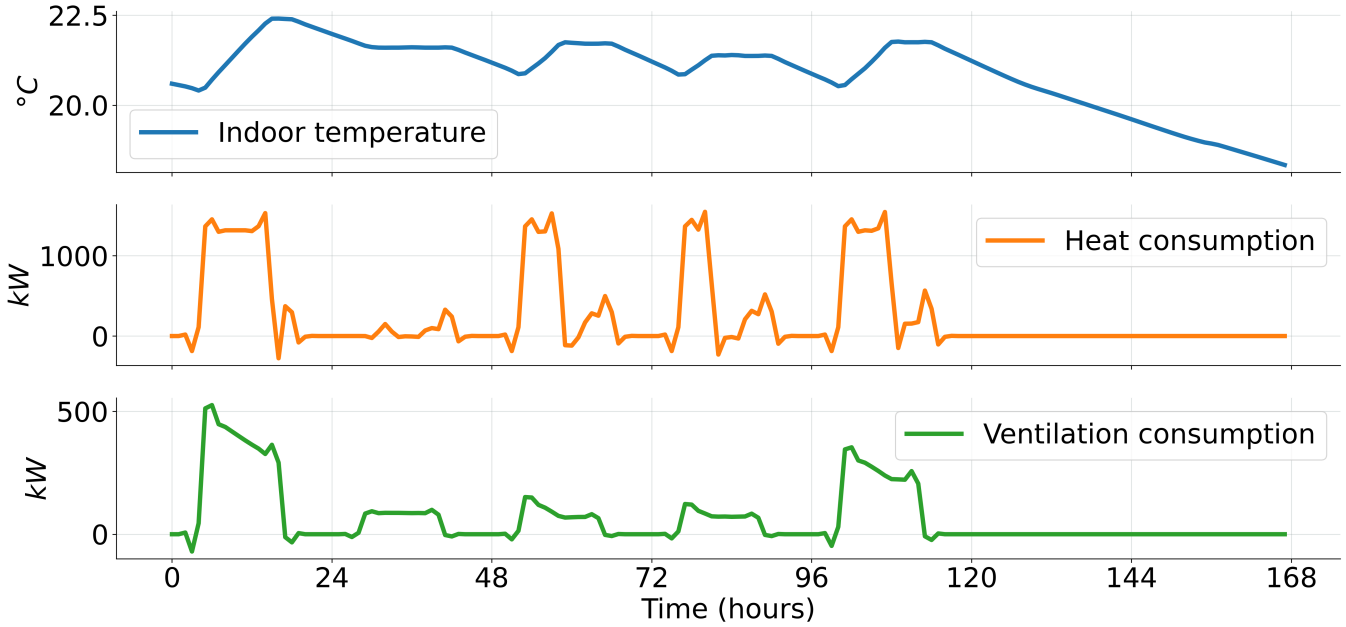



FIGURE 1.4 – Échantillons simulés à partir de TRNSYS, nous affichons un extrait des consommations $\zeta$ et de la température intérieure $\tau$.



**Intégration dans la solution R&D existante.** Lors du début des travaux présentés dans ce document, TRNSYS était déjà utilisé dans la solution développée par l'équipe R&D, et les tâches de calibration et d'optimisation, décrites plus bas, étaient déjà implémentées. Cependant, elles entraînaient de long temps de calcul, dûs aux nombreux appels nécessaire à TRNSYS, ainsi qu'à la nécessité de paramétrer chaque bâtiment individuellement. De plus, TRNSYS étant un logiciel expert, nous ne pouvons interpréter ou modifier sa représentation interne du bâtiment, en d'autre termes TRNSYS est, pour nous, une boîte noire. Nous ne pouvons donc pas utiliser les données réelles, récoltées par Oze-Energies, afin d'améliorer le modèle, ou de prendre en compte les sources potentielles d'incertitude.

### 1.2.2 Calibration et optimisation.

Oze-Energies vise à réduire la consommation énergétique des bâtiments, tout en maintenant leur niveau de confort. Le département de R&D propose des solutions algorithmiques afin d'assister les experts de la thermique dans la génération de scénarios associés à un bâtiment, ainsi que dans leur comparaison. Dans cette section, nous détaillons ce processus de calibration, puis d'optimisation.

**Calibration.** Afin de pouvoir générer des courbes de températures intérieures et les consommations associées du bâtiment, nous avons besoin des paramètres physiques $\lambda$, ainsi que des usages $(\psi_k)_{1\leq k\leq T}$ décrits dans la section précédente. Puisque ces paramètres sont en général inconnus, nous devons les estimer.

Nous avons tout d'abord étudié différentes approches pour estimer certains paramètres spécifiques. Par exemple, la capacité thermique d'un bâtiment peut être approchée en estimant la pente de l'évolution de la température intérieure ; de manière similaire, en analysant les courbes de consommation, on peut repérer les horaires de démarrage des HVAC, et donc en déduire les usages. Cependant, estimer chaque paramètre de cette manière est non seulement beaucoup trop coûteux, mais risque de surcroit de mener vers une modélisation biaisée du bâtiment. On notera que les experts thermiciens de Oze-Energies pourrait analyser directement le bâtiment sur place, mais encore une fois cela engendrerait des coûts trop importants.

Nous proposons plutôt d'estimer tous les paramètres inconnus en même temps, à travers une procédure de calibration automatique, qui consiste à minimiser une fonction de coût qui associe, avec chaque jeu de paramètres, la différence entre une simulation du modèle et les données historiques du bâtiment, voir par exemple Coakley et al. (2014); Le Corff et al. (2018) :

$$\mathcal{L}_{\texttt{calib}}(\lambda,\psi,\delta) = \mathcal{L}^{\texttt{calib}}_{\texttt{temperature}}(\tau,\hat{\tau}) + \beta\mathcal{L}^{\texttt{calib}}_{\texttt{energy}}(\zeta_k,\hat{\zeta}_k) \tag{1.1}$$

où $\hat{\tau},\hat{\zeta} = f_{\texttt{building}}(\lambda,\psi,\delta,\varphi)$ et $\beta$ est une paramètres d'échelle. Comme présenté dans Nagpal et al. (2019), cette méthodologie fournie des résultats précis pour de grande variété de bâtiments.

Nous déterminons un jeu de paramètres en utilisant des algorithmes génétiques, qui font partie des méthodes les plus efficaces pour la minimisation de fonctions non dérivables. Dans Aird et al. (2016), les auteurs démontrent l'utilité du Non-dominated Sorting Genetic Algorithm II (NSGA-II) afin de sélectionner un jeu de paramètres estimés minimisant à la fois un coefficient de variation sur l'erreur quadratique, ainsi que son biais. Ces critères peuvent aussi être combinés afin de se concentrer sur des méthodes d'optimisation à un seul objectif. Dans Le Corff et al. (2018), l'algorithme CMA-ES, introduit par Igel et al. (2007), est utilisé pour minimiser une combinaison d'erreurs liées à la prédiction de la consommation du chauffage et de la climatisation.

**Optimisation.** L'optimisation consiste à déterminer un jeu de paramètres qui mènera à une baisse de la consommation, tout en maintenant le niveau de confort. Pour cela, nous pouvons jouer sur les paramètres d'usage : par

exemple, en décalant les horaires de démarrage de la climatisation et du chauffage, ou en les réduisant pendant la nuit et le week-end. Nous considèrerons ici qu'une consommation plus faible correspond à une diminution de la consommation totale du bâtiment, sommée sur la fenêtre temporelle étudiée. On notera donc que nous ne prenons pas en compte les pics de consommation, ou l'évolution du prix du kilowattheure pendant la journée. Le confort est quant à lui défini comme la différence quadratique entre la température intérieure du bâtiment, et une température cible notée $T^\star$, pendant les périodes d'occupation. Nous définissons les critères suivants :

$$\mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(T^\star, \hat{\tau}) \quad \text{et} \quad \mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\hat{\zeta}) \tag{1.2}$$

où $\hat{\tau}, \hat{\zeta} = f_{\texttt{building}}(\lambda, \psi, \delta, \varphi)$.

Contrairement à la tâche de calibration, nous devons ici résoudre un problème d'optimisation bi-objectif, car il n'existe pas de jeu de paramètres $\psi^\star$ permettant de minimiser les deux objectifs de consommation et de confort conjointement. En effet, si une telle solution existait, on pourrait toujours améliorer un peu plus l'un des deux critères en dégradant l'autre, par exemple on peut toujours réduire un peu plus la consommation de chauffage, mais cela mènera à une dégradation du confort. Nous cherchons donc un ensemble de paramétrages équivalents, permettant un compromis optimal entre les deux objectifs, que l'on formalise en front de Pareto.

Un jeu de paramètres $\psi^\star \in \Psi$ en domine (Pareto) un second $\psi \in \Psi$ s'il vérifie les deux conditions suivantes :

1. $\mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(\psi^\star) \leq \mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(\psi)$ et $\mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\psi^\star) \leq \mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\psi)$,
2. $\mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(\psi^\star) > \mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(\psi)$ ou $\mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\psi^\star) > \mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\psi)$.

Une solution $\psi^\star \in \Psi$ est alors optimale s'il n'existe aucune autre solution qui la domine. Le front de Pareto et l'ensemble des solutions optimales.

Sur l'illustration de la Figure 3.8, chaque point de l'espace représente la consommation totale et le niveau de confort correspondant à un jeu de paramètres. Le front de Pareto, représenté par les points tracés en bleu, divise l'espace en deux parties : au dessus du front se trouvent les compromis sous optimaux entre consommation et confort ; la zone en dessous n'est pas atteignable dans le contexte de nos simulations.

### 1.2.3 Qualité de l'air

Nous nous intéressons maintenant à l'impact de l'air intérieur sur les conditions d'hygiène et de confort, à travers l'analyse de la qualité de l'air. Parmi les nombreux facteurs qui y jouent un rôle, détaillés dans Zhang et al. (2021), nous n'étudions que les grandeurs mesurées par Oze-Energies : la température intérieure, la concertation de $CO_2$ et l'humidité relative.

Il a été montré que la qualité de l'air a un impact fort sur le confort intérieur, et peut mener à de mauvaises conditions d'hygiène. Par exemple, de fortes concentrations de $CO_2$ , dues en général à une surpopulation dans une pièce close en l'absence de ventilation, peut être la source de maladies telles que le Sick Building Syndrome détaillé dans Hou et al. (2021). Un air trop ou pas assez humide mène au développement de bactéries, virus et champignons, voir le graphique de Sterling en Figure 2.4.

La modélisation de la qualité de l'air est un sujet central tout au long de cette thèse. Dans le Chapitre 3, nous optimisons le confort lié à la température intérieure, afin qu'elle ne dépasse pas des limites prédéfinies. Dans les Chapitre 4 et Chapitre 5, nous modélisation l'évolution de l'humidité relative.

## 1.3 Les motivations derrière la modélisation statistique

### 1.3.1 Métamodélisation de TRNSYS à travers des méthodes d'apprentissage profond

Nous avons pour objectif de répliquer les simulations du logiciel TRNSYS à travers un modèle statistique qui serait à la fois plus rapide et plus flexible. En effet, TRNSYS est particulièrement lent à opérer. Si les quelques secondes nécessaires à son lancement ne posent en général pas de problème aux experts, elles représentent un temps colossal lorsque nous effectuons nos tâches d'optimisation itératives (la calibration et l'optimisation). En revanche, les modèles statistiques modernes sont capable de profiter des dernières avancées techniques dans le domaine du hardware, en particulier les cartes graphiques (GPU pour Graphical Processing Unit) qui leur permettent de calculer des dizaines de simulations en parallèle. De plus, alors que TRNSYS est une boîte noire d'un point de vue statistique, le métamodèle que nous proposons permet, à terme, de modéliser l'incertitude liée aux prédictions, comme détaillé dans les Chapitre 4 et Chapitre 5.

Une approche naïve pour modéliser le comportement d'un bâtiment consisterait à assembler une base à partir des données historiques disponibles, puis entraîner un modèle statistique tel qu'un réseau de neurones sur cette base. Cependant, les données disponibles se sont révélées trop bruitées et clairsemées pour permettre un entraînement correct, vis à vis de la tâche à réaliser. Nous proposons la méthodologie suivante :

1. **Entraînement d'un métamodèle** : en générant divers scénarios de bâtiments (propriétés physiques, usages, conditions météorologiques) à partir de TRNSYS, pour construire une base de données synthétique sur laquelle nous entraînons les différents modèles présentés dans ce manuscrit.
2. **Calibration et optimisation** : le métamodèle remplace TRNSYS lors des tâches de calibration et d'optimisation pour permettre d'en réduire largement le temps de calcul.
3. **Amélioration du métamodèle** : nous pouvons estimer l'incertitude du modèle à partir des données historiques disponibles.

**Réseaux de neurones profonds.** Les méthodes d'apprentissage profond visent à exploiter des quantités de données de plus en plus volumineuses. Elles consistent à combiner des fonctions paramétriques non linéaires différentiables, nommées couches, pour résoudre une tâche de classification ou de régression à l'aide d'une fonction de coût. Les paramètres des couches sont estimés de manière itérative par descente de gradient.

Les réseaux de neurones profonds ont remplacé la plupart des architectures traditionnelles dans beaucoup de champs d'application tels que la vision par ordinateur, le traitement naturel du langage ou encore la prévision de séries temporelles. Parce qu'ils sont au centre d'un champ de recherche qui évolue rapidement, et pour les applications diverses qu'ils permettent, beaucoup des modèles présentés dans ce manuscrit sont basés sur des architectures de réseaux de neurones.

### 1.3.2 Estimation de l'incertitude

Les réseaux de neurones profonds sont souvent prisés pour leur capacité à estimer des millions de paramètres à partir de gigantesques bases de données, et ainsi de résoudre des tâches complexes. C'est pourquoi nous nous en sommes inspiré pour développer l'architecture du métamodèle. En effet, le comportement de TRNSYS est très complexe, fortement non linéaire et traite des vecteurs d'entrée et de sortie de grande dimension. Cependant, les réseaux de neurones produisent aussi des prévisions incorrectes, qu'il n'est pas aisé de discerner. Puisque nous ne pouvons pas simplement nous reposer sur les capacités du métamodèle à être précis pour gérer correctement la consommation et le confort dans un bâtiment, notre travail nous mène à modéliser l'incertitude des modèles statistiques que nous développons.

Nous proposons de quantifier l'incertitude associée à une prédiction en modélisant la distribution des observations. Si cela ne permettra sans doute pas d'améliorer les performances, il sera tout de même possible d'interpréter la distribution prédite afin de quantifier l'incertitude. Par exemple, si cette distribution semble Gaussienne avec une petite variance, on pourra considérer la prédiction du modèle crédible. Si la distribution semble bimodale, la prédiction associée ne permet probablement pas d'apprécier correctement les variables estimées.

Après avoir développé le métamodèle dans le Chapitre 3, nous explorons deux méthodes de quantification de l'incertitude dans le Chapitre 4, avant de détailler et d'étendre la seconde dans le Chapitre 5.

### 1.3.3 Changement de régime pour les modèles à états cachés discrets

Dans cette thèse, nous modélisons l'évolution de variables liées à un bâtiment à travers des modèles à état cachés, qui reposent sur une représentation interne des données pour modéliser les observations. Si ces états cachés sont souvent modélisés comme des variables continues, nous présentons aussi des modèles à états cachés discrets, et cela pour deux raisons principales. Tout d'abord ils peuvent permettre de simplifier la procédure de l'entraînement, car on peut alors s'affranchir de certaines approximations complexes nécessaires dans le cas continu. De plus, ils conviennent particulièrement bien à la modélisation des problèmes de changement de régimes, comme nous le détaillons maintenant.

Représenter l'évolution d'un bâtiment à travers un nombre fini d'états cachés peut grandement en simplifier la modélisation, sans nécessairement sacrifier les performances. Par exemple, on peut imaginer un régime qui résumerait le comportement du bâtiment chaque matin (la température extérieure augmente, l'humidité décroît, les occupants qui arrivent participent au réchauffement du bâtiment), puis des comportements différents pour chaque saison de l'année, ou encore simplement pour différencier la semaine du week-end.

Nous démontrons l'intérêt de ces modèles à états latents discrets en les comparant à leur contrepartie continue, sur une tâche de prévision de l'humidité relative, dans le Chapitre 4. Dans le Chapitre 5, nous étendons notre cadre de travail à des distributions discrètes plus complexes.

## 1.4 Contributions

Dans cette thèse, nous présentons les contributions suivantes. Dans le Chapitre 3, nous proposons d'entraîner un métamodèle basé sur un réseau de neurones récurrent (RNN). Nous le comparons à plusieurs approches alternatives, qui illustrent que les modèles de traitement des séquences conduisent à une amélioration significative des performances par rapport aux méthodes de l'état de l'art. Notre métamodèle est alors calibré aux données historiques réelles de deux bâtiments, afin d'illustrer la flexibilité de notre approche. La dernière étape de notre méthodologie de bout en bout consiste à optimiser la consommation énergétique, tout en maintenant le niveau de confort. Cette méthodologie nous permet de réduire la consommation des deux bâtiments précédemment mentionnés de 5% et 10%. Les résultats présentés sont adaptés de la contribution suivante : *End-to-end deep meta modelling to calibrate and optimize energy consumption and comfort*, Cohen, M. Le Corff, S., Charbit, M., Champagne, A., Nozière, G, Preda, M., Energy and Buildings, Volume 250, November 2021.

Dans le Chapitre 4, nous présentons deux approches pour modéliser l'incertitude de modèles statistiques, tels que le métamodèle, appliqué à la prévision de l'humidité relative. Nous proposons d'abord de découpler l'apprentissage de la représentation latente des données et celui de l'incertitude, dans une procédure d'entraînement à deux étapes. Les paramètres inconnus sont estimés en minimisant une fonction de coût déterministe, puis la dernière couche du modèle est entraînée à nouveau en utilisant des méthodes de Monte Carlo Séquentielles. Les résultats présentés sont adaptés de la contribution suivante : *Last layer state space model for representation learning and uncertainty quantification*, Cohen, M., Charbit, M. and Le Corff, S., 2023 IEEE Statistical Signal Processing Workshop.

Nous présentons ensuite une seconde approche basée sur un modèle à états cachés discrets. Nous montrons ainsi que l'utilisation de régimes discrets permet une meilleure interprétabilité du modèle, sans perdre en précision. De plus, l'estimation des paramètres, réalisée par inférence variationnelle, ne nécessite pas d'approximation complexe. Les résultats présentés sont adaptés de la contribution suivante : *Variational Discrete Latent Representation for Time Series Modelling*, Cohen, M., Charbit, M. and Le Corff, S., 2023 IEEE Statistical Signal Processing Workshop.

Dans le Chapitre 5, nous modélisons des distributions discrètes plus complexes de l'espace latent. Par rapport aux modèles a priori de la littérature, dont l'architecture et la complexité entraînent l'utilisation d'approximations diverses durant l'entraînement, nous proposons un cadre de travail théorique pour définir et entraîner des modèles à états latents discrets, en utilisant des ponts de diffusion. Nous montrons que notre architecture produit des performances similaires à l'état de l'art, sur des tâches de vision par ordinateur telles que la synthèse d'images, ou la complétion d'images, et ouvre la voix à de nouvelles perspectives. Les résultats présentés sont adaptés de la contribution suivante : *Diffusion Bridges Vector Quantized Variational Autoencoders*, Cohen, M., Quispe, Q., Le Corff, S., Ollion, C., Moulines, E., Proceedings of the 39th International Conference on Machine Learning (ICML), Volume 162.

# Bibliographie


Aird, G., Coakley, D., and Kerrigan, R. (2016). Application of an optimisation approach for the calibration 1 of high-fidelity building energy models to support model-2 predictive control ( mpc ) of hvac systems.

Bre, F., Silva, A. S., Ghisi, E., and Fachinotti, V. D. (2016). Residential building design optimisation using sensitivity analysis and genetic algorithm. *Energy and Buildings*, 133 :853–866.

Coakley, D., Raftery, P., and Keane, M. (2014). A review of methods to match building energy simulation models to measured data. *Renewable & Sustainable Energy Reviews*, 37 :123–141.

Hou, J., Sun, Y., Dai, X., Liu, J., Shen, X., Tan, H., Yin, H., Huang, K., Gao, Y., Lai, D., Hong, W., Zhai, X., Norbäck, D., and Chen, Q. (2021). Associations of indoor carbon dioxide concentrations, air temperature, and humidity with perceived air quality and sick building syndrome symptoms in chinese homes. *Indoor Air*, 31(4) :1018–1028.

Igel, C., Hansen, N., and Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 11 :1–28.

Le Corff, S., Champagne, A., Charbit, M., Noziere, G., and Moulines, E. (2018). Optimizing thermal comfort and energy consumption in a large building without renovation works. *2018 IEEE Data Science Workshop (DSW)*, pages 41–45.

Magnier, L. and Haghighat, F. (2010). Multiobjective optimization of building design using trnsys simulations, genetic algorithm, and artificial neural network. *Building and Environment*, 45 :739–746.

Nagpal, S., Mueller, C., Aijazi, A., and Reinhart, C. (2019). A methodology for auto-calibrating urban building energy models using surrogate modeling techniques. *Journal of Building Performance Simulation*, 12 :1 – 16.

Recht, T., Munaretto, F., Schalbart, P., and Peuportier, B. (2014). Analyse de la fiabilité de COMFIE par comparaison à des mesures. Application à un bâtiment passif. In *IBPSA 2014*, Arras, France.

Shabunko, V., Lim, C., and Mathew, S. (2018). EnergyPlus models for the benchmarking of residential buildings in Brunei Darussalam. *Energy and Buildings*, 169 :507–516.

Zhang, H., Srinivasan, R. S., and Ganesan, V. (2021). Low cost, multi-pollutant sensing system using raspberry pi for indoor air quality monitoring. *Sustainability*.

Zhao, J., Lam, K. P., Ydstie, B. E., and Loftness, V. (2016). Occupant-oriented mixed-mode EnergyPlus predictive control simulation. *Energy and Buildings*, 117 :362–371.

# Chapitre 2

# Introduction

## 2.1 Oze-Energies

Oze-Energies is a French company specialized in optimizing building's energy consumption. Global energy demand for heating, ventilation and air-conditioning in commercial or public buildings has been increasing rapidly for the past few decades. This rising demand is at the root of the complex problem of simultaneously maintaining a satisfactory thermal comfort in buildings while reducing the environmental impact. Through innovative and durable methods, Oze-Energies aims at improving air quality and comfort, while simultaneously reducing energy consumption, without requiring any site work.

In 2009, the building stock accounted for over 40% of the total French energy consumption, as well as almost a quarter of greenhouse emissions (Loi Grenelle [1], Figure 1.1). The most straight forward strategy for reducing building's consumption consists in improving their isolation, which usually involves costly renovation works. Despite aiming to renovate 500,000 buildings every year before 2020 [2] [3], the actions carried out in France to this date still fall short in terms of results, as stated by the Ademe (Agency for the environnement and energy) [4]. According to the National Low-Carbon Strategy (SNBC), the average number of yearly renovations is expected to be around 370,000 for the period 2015-2030. In contrast, the premise of Oze-Energies is the ability to produce tailored road maps for managing buildings, without requiring any renovation work.

Building's energy consumption can be largely reduced by tuning the HVAC (Heating Ventilation and Air Conditioning) settings. Oze-Energies targets tertiary buildings, whose occupation behaviors are usually well understood (workday hours, no occupation during the weekend), however their behaviors still differ from one to another. This is due to a wide variety of factors, such as isolation, occupancy, or heating and cooling providers : it is one of the major technical challenges of this approach that we will detail in the following chapters. In order to precisely model the heat exchanges of each unique building, Oze-Energies monitors their consumption and air quality by integrating environmental sensors, connected through LoRA (a secured and public network) and reporting to a centralized dataset. By combining this data with their precise understanding of building's behavior, experts in energy efficiency known as Energy Managers are able to produce road maps for HVAC settings. They aim at reaching the best compromise between indoor comfort and energy consumption, while improving air quality ($CO_2$ levels, humidity, etc.). These road maps average in a 25% reduction in consumption, in just a few weeks. They are delivered for a recurring subscription, usually largely covered by the energy savings generated : this is Oze-Energies' main product, OPTIMZEN®.

1. Loi Grenelle I, Article 3
2. Stratégie Nationale Bas-Carbone (SNBC)
3. Loi Grenelle I, Article 5
4. Hervé Lefebvre, head of the Climat departiement of the Ademe, 2019

Oze-Energies has been exploring methods to reduce the tedious work of Energy Managers : estimating building's physical attributes, proposing new road map scenarios, quantitatively comparing them. In particular, a numerical simulator can be configured to estimate the best management settings for a building, by predicting and comparing the impact of various policies. This expert software, TRNSYS, is able to simulate the complex behavior of buildings based on their schematics, as well as numerous related inputs such as building occupation, HVAC settings or ambient weather. TRNSYS is first calibrated to match the real building by a machine learning procedure, using data collected from sensors. As previously mentioned, due to the number of heat, cold or electricity providers, this calibration step becomes more and more complex as Oze-Energies acquires new clients. Once calibrated, TRNSYS can optimize the HVAC settings to reduce consumption while maintaining or even improving comfort in the building.

Oze-Energies was created in 2014, covers over 5 millions $m^2$ over 500 buildings, and is now part of Accenta.

## 2.2 Optimizing comfort and consumption

### 2.2.1 Sensors, counters and weather data

Knowledge of a building's behavior is assembled by combining various kinds of data, usually gathered directly from the site. Therefore, the first step in managing a new building consists in setting up counters responsible for sending data back to Oze-Energies' servers every few minutes. They can be divided in three main categories.

— **Sensors** are installed by Oze-Energies Managers in various areas of the building, and collect ambient variables, such as $CO_2$ levels, humidity, or indoor temperature. They typically report data every 10 minutes. Because they can greatly vary from one area to another, sensor data involves a challenging aggregation problem.

— **Counters** report the building's consumption, such as heating, cooling, or electricity. We usually have access to a single time series for each variable, so no further aggregation task is required ; however, the main challenge of handling counters data resides in the variety of energy providers and building usage. While some providers report consumption every hour, other only give access to a single daily aggregated value, requiring a complex reconstruction task to get an hour to hour time series, usually based on usual consumption patterns. In addition, counter error handling can differ, as some providers ignore outliers, while others compensate on subsequent values. Finally, usage can differ significantly from one building to another : for instance, heating can be achieved through a local heater, consuming electricity, or by connecting the building to a heat provider.

— **Weather data**, such as outdoor temperature and humidity, or irradiance levels, play a significant role on buildings' behaviors. Historic data are available from the past few years. Weather forecast are also available, and considered to be accurate up to a week forward. These sensors are not setup by Oze-Energies directly, but rather by a weather forecasts company : Meteotest [5].

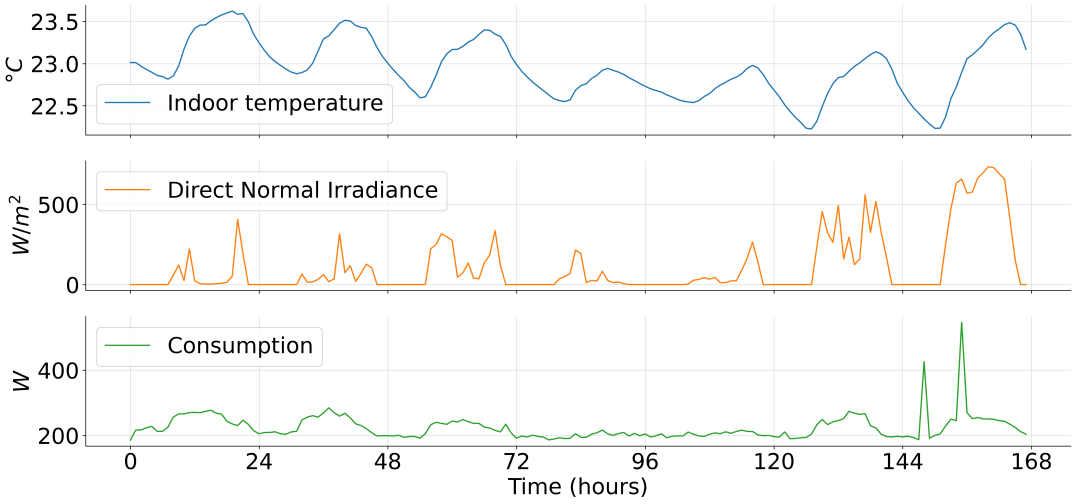
A sample of the available historic data is presented in Figure 2.1.

#### Building behavior simulation with TRNSYS

Physical simulators based on thermal propagation equations are traditionally used to describe buildings. These transient systems simulators can predict the energy use based on a description of the building's layout, construction materials and dimensions, usage and HVAC schedules along with weather data. Because of the sheer complexity arising from modelling such diverse systems, only a few simulators are available today.

5. https ://meteotest.ch/

FIGURE 2.1 – We plotted week long samples of sensors (indoor temperature), counters (consumption) and weather data (Direct Normal Irradiance).




EnergyPlus was used for instance in Shabunko et al. (2018) to build three types of typical designs and to benchmark the energy performance of 400 residential buildings. In Zhao et al. (2016), the authors proposed a predictive control framework based on Matlab and EnergyPlus in order to optimize energy consumptions while meeting the individual thermal comfort preference. In Magnier and Haghighat (2010), the authors highlighted the performance of TRNSYS as a physical simulator, as well as its limits in terms of computational speed : the authors claimed that a full optimization process would take as much as ten years, had they not replaced TRNSYS with a faster surrogate model during optimization. The authors of Bre et al. (2016) studied the optimization of a single-family house using a combination of Energy-Plus and the NSGA-II optimization algorithm, and discussed sensitivity analysis using the Morris screening method. Likewise, the authors of Recht et al. (2014) performed sensitivity and uncertainty analysis on another building simulator known as COMFIE, and displayed its modelling performance on a passive building.

Among them, Oze-Energies chose TRNSYS for its completeness, flexibility and popularity. However, TRNSYS only runs under the Windows operating system, and requires launching a window for every single simulation, incurring a runtime of at least a few seconds.

**Simulation parameters.** In this section, we describe the simulation parameters used by TRNSYS. Although the majority of these parameters are relevant for any type of buildings, the specific variables presented bellow were designed for two buildings handled by Oze-Energies, introduced later in Section 3.2. We plotted a sample of the available time series in Figure 2.2, while an example of TRNSYS simulations can be found in Figure 2.3. Please check Appendix A.1.2 for an exhaustive list and description of all used variables. In order to model more complex buildings in the future, introducing new parameters may be required.

— Physical properties of the building, such as heat capacity, size of the isolation, etc. are stacked together and denoted by a vector $\lambda \in \mathbb{R}^{d_\lambda}$, where $d_\lambda$ is the number of parameters. $\Lambda$ is the set of all possible set of physical properties.

— Usage is encoded as a time series describing the state of the HVAC at each hour. Let $T$ be the length of a sample, usage is defined as $(\psi_k)_{1 \leq k \leq T} \in \mathbb{R}^{d_\psi}$. Here, $d_\psi$ is the number of individual variable, such as ventilation schedule, heating temperature, etc. $\Psi$ is the set of all possible HVAC settings.

— Similarly, occupancy is encoded as a time series $(\delta_k)_{1 \leq k \leq T} \in \mathbb{R}$ denoting the fraction of occupants present in the building at each hour. $\Delta$ is the set of all possible occupancy schedules.

— Weather conditions are gathered as a time series $(\varphi_k)_{1\leq k\leq T} \in \mathbb{R}^{d_\varphi}$.

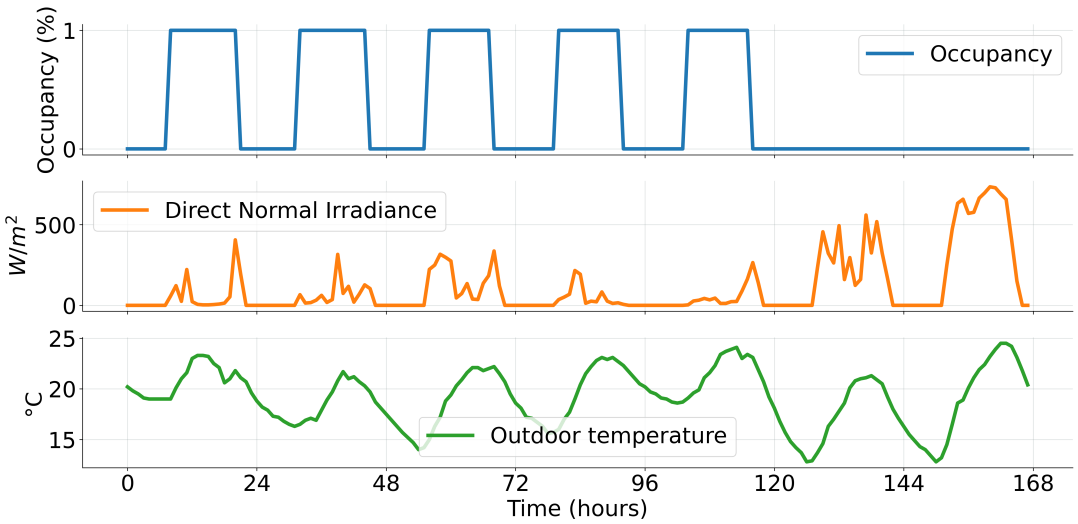
— Finally, consumptions are denoted as $(\zeta_k)_{1\leq k\leq T} \in \mathbb{R}^{d_\zeta}$, and indoor temperature is denoted $(\tau_k)_{1\leq k\leq T} \in \mathbb{R}$. $\mathcal{Z}$ (resp. $\mathcal{T}$) is the set of all possible consumptions (resp. indoor temperatures).

In the rest of this document, we use the short hand notation $\varphi$ (resp. $\psi, \delta, \zeta, \tau$) to denote the entire time series $(\varphi_k)_{k=1}^T$ (resp. $(\psi)_{k=1}^T, (\delta)_{k=1}^T, (\zeta)_{k=1}^T, (\tau)_{k=1}^T$).

Building simulators, such as TRNSYS, are functions taking as input the physical properties of the building, its usage and occupancy for the given period, as well as the associated weather data, and predict (among others) the hourly indoor temperature and consumptions.

$$f_{\texttt{building}} : (\lambda, \psi, \delta, \varphi) \mapsto (\hat{\tau}, \hat{\zeta}).$$

FIGURE 2.2 – On this figure, we plotted samples from the available occupancy $\delta$ as well as weather data $\varphi$.



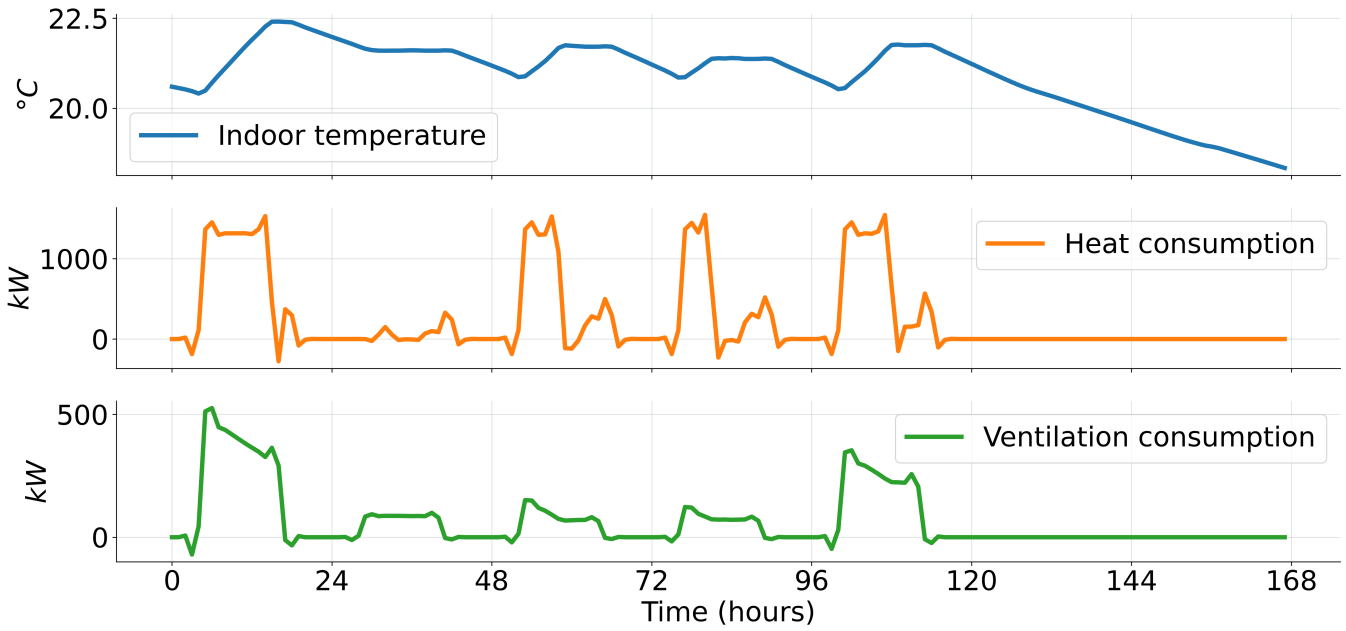

**Integration in the R&D pipeline.** At the beginning of this work, TRNSYS was already integrated in the R&D pipeline in order to run automated simulations, as well as as calibration and optimization tasks that are described further below. However, this implementation suffered long computation times, linked to the numerous calls to the TRNSYS simulation function, as well as the need to manually configure each new building. Furthermore, because TRNSYS is an expert software, we have no way to interpret or modify its internal representation of the problem, in other words TRNSYS is a black box. Because of this, we cannot benefit from the real data gathered in order to learn model noise or other uncertainty sources.

### 2.2.2 Calibration and optimization.

The aim of Oze-Energies is to reduce energy consumption while improving, or maintaining comfort. While Energy Managers use their own knowledge and experience to reach these objectives, the R&D department implements algorithmic solutions to provide coherent scenarios for the building, and later assist Energy Managers in their decision. In this section, we describe the two-step process for producing these scenarios.

**Calibration.** Sampling indoor temperatures and consumptions associated with a given time period requires estimates of the unknown physical parameters $\lambda$ as well as usage $(\psi_k)_{1\leq k\leq T}$ described in the previous section.

FIGURE 2.3 – Sample simulated with TRNSYS, we plotted consumptions $\zeta$ and indoor temperature $\tau$.




We first consider that various methods can be applied for estimating specific parameters. For instance, the heat capacity of the building can be approximated by analyzing the slope of the indoor temperature curve ; start and stop hours of the heaters can be extrapolated from the time of increase of the consumption every morning and evening. However, estimating each parameter on its own is not only exceedingly time consuming, but also not guaranteed to result in an good estimation of the building behavior and these estimations are likely to be biased. Additionally, building experts could analyse and estimate those parameters on site, but such costly measure campaigns would have to be reiterated for every new building.

Instead, these unknown parameters may be estimated using an automatic calibration procedure, by minimizing a cost function which associates, with each set of parameters, the discrepancy between the simulation and the true consumptions and temperatures, see Coakley et al. (2014); Le Corff et al. (2018) :

$$\mathcal{L}_{\texttt{calib}}(\lambda, \psi, \delta) = \mathcal{L}^{\texttt{calib}}_{\texttt{temperature}}(\tau, \hat{\tau}) + \beta \mathcal{L}^{\texttt{calib}}_{\texttt{energy}}(\zeta_k, \hat{\zeta}_k) \tag{2.1}$$

where $\hat{\tau}, \hat{\zeta} = f_{\texttt{building}}(\lambda, \psi, \delta, \varphi)$ and $\beta$ is a scaling parameter. As shown in Nagpal et al. (2019), calibration yields sufficiently accurate results for a variety of different buildings.

The calibration task revolves around a non differentiable optimization problem, which is often tackled using genetic optimization methods. In Aird et al. (2016), the authors demonstrate the use of the Non-dominated Sorting Genetic Algorithm II (NSGA-II) to select a set of estimated parameters that jointly minimize the coefficient of variation of the root mean square error, and the normalized mean bias error. All criteria can instead be combined in a single calibration error, in order to turn to single objective differentiation free algorithms that offer a single best candidate, avoiding the need for further selection processes. In Le Corff et al. (2018), the CMA-ES algorithm introduced in Igel et al. (2007) was used to minimize a combination of heating and cooling errors.

**Optimization.** The optimization task aims at reaching scenarios corresponding to lower consumption and higher comfort, by changing the usage of the building, in other words by improving the handling of the HVAC : changing the start and stop time of heaters, lowering cooling during the night, etc. We consider that lowering consumption simply

corresponds to reaching a lower total consumption over the considered time frame, although it could be argued that high peak in consumption should be further penalized. We define a comfort metric as the squared distance between indoor temperature and a target temperate $T^\star$ during occupancy hours. From the simulation parameters defined in the previous section, we define the two following criteria :

$$\mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(T^\star, \hat{\tau}) \quad \text{and} \quad \mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\hat{\zeta}) \tag{2.2}$$

where $\hat{\tau}, \hat{\zeta} = f_{\texttt{building}}(\lambda, \psi, \delta, \varphi)$.

Unlike the calibration task, we are facing a bi-objective problem, as we cannot find a set of usages $\psi^\star$ that optimizes both objectives simultaneously. For any such solution, we can always further improve one of the objectives by degrading the other, as an example reducing the heater will save energy but lower comfort, while providing the exact targeted indoor temperature throughout the day will undoubtedly increase consumption. Instead, we search for a collection of equivalent compromises between the two objectives, formalized as a Pareto front.

A feasible solution $\psi^\star \in \Psi$ is said to Pareto dominate another solution $\psi \in \Psi$ if both :

1. $\mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(\psi^\star) \leq \mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(\psi)$ and $\mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\psi^\star) \leq \mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\psi)$,
2. $\mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(\psi^\star) > \mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(\psi)$ or $\mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\psi^\star) > \mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\psi)$.

A solution $\psi^\star \in \Psi$ is then called Pareto optimal if there does not exist another solution that dominates it. The set of Pareto optimal outcomes, is called the Pareto front.

In the illustration shown in Figure 3.8, each point in space represents the total consumption and comfort level corresponding to a single set of usage. The Pareto front, formed by the set of drawn points, divide the space in two parts : the upper zone corresponds to sub optimal compromises between consumption and comfort ; the lower zone is not physically attainable (in the context of our simulations).

### 2.2.3 Air quality

The indoor air quality measures the impact of indoor air conditions on health and well being. A wide array of chemicals and conditions are involved in its analysis, as detailed in Zhang et al. (2021). In this work, we will only study the variables monitored by Oze-Energies, for which we can retrieve historic data : indoor temperature, $CO_2$ concentrations and relative humidity.

Air quality has been shown to strongly impact indoor well being ; when poorly managed, indoor air can lead to unhealthy conditions. For instance, high $CO_2$ concentrations, usually caused by having too many people enclosed in a limited space with no ventilation, can cause a disease known as Sick Building Syndrome detailed in Hou et al. (2021). Low and high relative humidity lead to the development of bacteria, viruses and fungi, as summarized in the Sterling Chart in Figure 2.4.

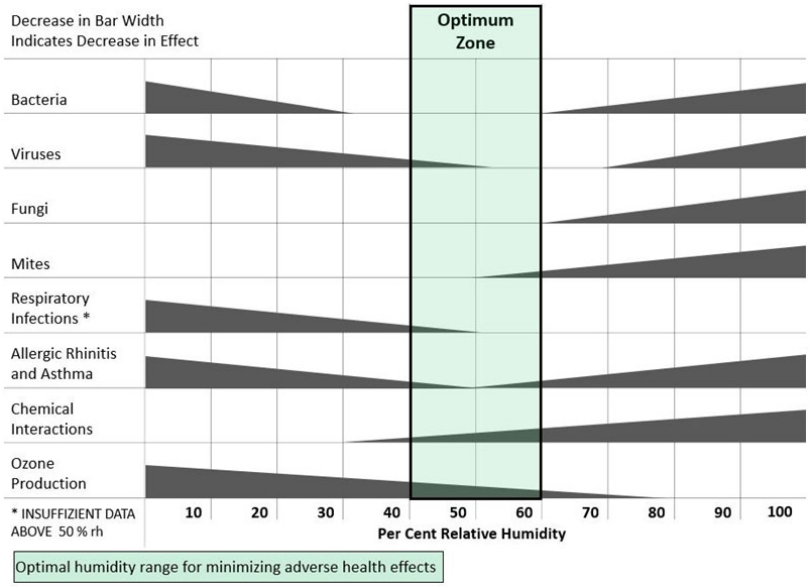
Air quality modelling is at the center of our focus all along this thesis. In Chapter 3, we optimize indoor temperature to remain within predefined ranges. In Chapter 4 and Chapter 5, we model the evolution of relative humidity.

## 2.3 Motivation for statistical modelling

### 2.3.1 Metamodelling TRNSYS through deep learning methods

We aim at developing statistical models able to provide similar simulations as TRNSYS, while being both faster and more flexible. One of the most cumbersome limitation of TRNSYS resides in its speed, or rather lack of. If a few seconds for an Energy Manager to generate a scenario is a reasonable delay, computation times become unmanageable for the calibration and optimization tasks presented in the previous section, as they require thousands of

FIGURE 2.4 – Sterling Chart introduced in Sterling et al. (1985).




calls to the simulation function. Modern statistical models are able to take advantage of the most recent hardware developments, such as the Graphical Processing Units (GPU), to compute dozens of simulations in parallel, in ever shorter times. Additionally, whereas TRNSYS is a black box from a statistical point of view, the metamodel we aim to design would be able to model uncertainty, as we detail in this document.

A naive data driven approach for modelling building's behavior could have consisted in creating a dataset from the available historic data, then training a statistical model such as a deep neural network. However, the data was too noisy and limited in the number of samples available for a neural network to learn such a complex task. This is why we propose the following methodology :

1. **Train a metamodel** : using TRNSYS, we sample various input scenarios (building properties, usage, weather conditions, etc.) and compute the associated simulation (indoor temperature and consumption). We train the models presented in this thesis on this synthetic dataset, in order to learn a surrogate function of the TRNSYS simulator.
2. **Perform calibration and optimization** : we can plug this metamodel to solve optimization and calibration tasks with limited computation time.
3. **Improve the metamodel** : finally, we can leverage and develop statistical methods for estimating uncertainty on the model, using the available historic data.

**Deep neural networks.** Deep learning methods aim at leveraging ever increasing amounts of data available. They consist in combining well known layers, differentiable nonlinear parametric functions, to approximate a given task associated with a loss function. The parameters are then updated iteratively, by gradient descent.

Neural networks have steadily replaced most traditional architectures in fields such as Computer Vision, Natural Language Processing, or time series forecasting. Because deep learning is a quickly evolving field of research, with diverse applications, many of the models presented in this thesis are derived from neural networks.

**Parameters estimation through automated gradient descent.** Neural networks have recently outperformed most other statistical models by inferring up to millions of parameters through an automated gradient descent. As all

layers composing a neural network are differentiable, it is possible to compute the gradient of the loss function with respect to each parameter, as a function of the architecture of the network. In other words, it is possible to define a training algorithm, based on gradient descent, able to estimate the parameters of a deep learning model, regardless of the number or type of layers it is composed of. By combining this flexibility of the training procedure, with modern computing units able to parallelize computations (GPU), neural networks have been able to take advantage of arbitrary amounts of training samples.

### 2.3.2 Uncertainty estimation

Deep learning models are often utilized for their ability to infer millions of parameters from huge amounts of data, leading to accurate predictions during inference. This is why we chose such architecture for the metamodel, as the behavior of TRNSYS is complex (strongly non linear, high dimensional inputs and outputs). Yet neural networks can provide erroneous predictions, especially when explanatory variables are unknown or unavailable, and are usually over confident in doing so. As we cannot simply rely on a neural network being correct to provide adequate air quality, our work with Oze-Energies will lead us to investigate uncertainty modelling methods.

On way to quantify the uncertainty associated with a prediction is to model the distribution of the observations, instead of simply predicting its most likely value. Although this could hardly improve performance, one could interpret the modelled distribution in order to quantify the uncertainty of the model. For instance, if its behavior is similar to a Gaussian with small variance, the prediction could be considered accurate. On the other hand, if the distribution seems bimodal or uniform, the associated prediction is likely not enough to understand the underlying state of the estimated variable.

After developing the metamodel methodology in Chapter 3, we explore two uncertainty estimation methods in Chapter 4 and dive into the details of the last one in Chapter 5.

### 2.3.3 Regime switching with discrete latent models

In this thesis, we model the evolution of building related variables mainly through hidden latent models, which rely on an internal representation of the data to model observations. While these latent variables are often continuous, we will be utilizing discrete latent models too, for two main reasons. Not only can they lead to simpler training procedure, as they often require less approximations in order to estimate their parameters, but they are also particularly adequate for modelling inherently discrete behaviors, such as regime switching.

Representing the evolution of a building through a finite set of states can greatly simplify our modelling task, without necessarily loosing in performance. For instance, we could imagine a regime summarizing building's behavior each morning (outdoor temperature rising, outdoor humidity lowering, indoor heating through occupants and electronic devices increasing), different behaviors for different seasons of the year or simply to easily differentiate week days from week ends.

We demonstrate the interest behind discrete latent models by first comparing them to their continuous counterpart on a relative humidity forecasting task in Chapter 4. In Chapter 5, we extend our framework to more complex discrete latent distributions.

## 2.4 Notation

In the following paragraphs, we detail a set of recurring symbols and acronyms. This list is not exhaustive.

### 2.4.1 Symbols

**Time series.**

— $y$ : observations
— $u$ : commands
— $x$ : latent states
— $a_{n:m}$ : the sequence $(a_n, \cdots, a_m)$

**Deep learning.**

— $\theta$ : unknown parameters
— $w, b$ : weights and biases of the models
— $f, g, h$ : non linear parametric functions

**Statistical modelling.**

— $\mathcal{L}$ : the Evidence Lower BOund, or ELBO
— $\mathbb{E}_q[X]$ : the expectation of the random variable $X$ with probability density $q$
— $\Psi_{\mu,\Sigma}$ : the Gaussian probability function with mean vector $\mu$ and covariance matrix $\Sigma$
— $\epsilon, \eta$ : centered Gaussian noise

### 2.4.2 Acronyms

— **HMM** Hidden Markov Model
— **FFN** Feed Forward Network
— **RNN** Recurrent Neural Network
— **LSTM** Long Short Term Memory
— **GRU** Gated Recurrent Unit
— **VI** Variational Inference
— **ELBO** Evidence Lower BOund
— **VAE** Variational Auto Encoder
— **VQ-VAE** Vector Quantized Variational Auto Encoder
— **SMC** Sequential Monte Carlo
— **MCD** Monte Carlo Dropout
— **MSE** Mean Squared Error
— **RMSE** Root Mean Squared Error
— **MAE** Mean Absolute Error

## 2.5 Contributions

In this thesis, we present the following contributions. In Chapter 3, we propose to train a metamodel based on Recurrent Neural Networks (RNN). We compare several approaches which illustrate that sequence to sequence models can yield a significant increase in performance with respect to the alternatives previously considered in our framework. Our metamodel, which depends on a few physical parameters, is then calibrated using real data to provide accurate predictions for two buildings, to illustrate the flexibility of this approach. The final step of our end-to-end methodology consists in optimizing energy consumption, while maintaining a given level of comfort. Following this methodology, we were able to train and calibrate our metamodel and to reduce the hourly consumption of two

buildings by 5% and 10%. The results presented are adapted from the following contribution : *End-to-end deep meta modelling to calibrate and optimize energy consumption and comfort*, Cohen, M. Le Corff, S., Charbit, M., Champagne, A., Nozière, G, Preda, M., Energy and Buildings, Volume 250, November 2021.

In Chapter 4, we present two approaches to model the uncertainty of statistical models, such as the metamodel, applied to relative humidity forecasting. We first propose to decouple representation learning from uncertainty modelling, in a two step training procedure. The unknown parameters are estimated by minimizing a deterministic cost function, then the last layer of the architecture is finetuned using Sequential Monte Carlo (SMC) methods. The results presented are adapted from the following contribution : *Last layer state space model for representation learning and uncertainty quantification*, Cohen, M., Charbit, M. and Le Corff, S., 2023 IEEE Statistical Signal Processing Workshop. In a second approach, we develop a model with a discrete latent representation of the data. We show that discrete regimes allow better interpretability of the model. Additionally, parameter estimation does not require the complex approximations that come with continuous latent vectors, and is achieved through Variational Inference. The results presented are adapted from the following contribution : *Variational Discrete Latent Representation for Time Series Modelling*, Cohen, M., Charbit, M. and Le Corff, S., 2023 IEEE Statistical Signal Processing Workshop.

In Chapter 5, we explore more complex modelling of discrete latent spaces. In contrast with most prior models in the literature, whose architecture and complexity entail to various implementation tricks during the training procedure, we propose a theoretically grounded framework for discrete latent models, using diffusion bridges. We show that our architecture is consistent with state of the art performance on computer vision tasks, such as image synthesis and inpainting, and offer new perspectives. The results presented in this chapter are adapted from the following contribution : *Diffusion Bridges Vector Quantized Variational Autoencoders*, Cohen, M., Quispe, Q., Le Corff, S., Ollion, C., Moulines, E., Proceedings of the 39th International Conference on Machine Learning (ICML), Volume 162.

# Bibliographie


Aird, G., Coakley, D., and Kerrigan, R. (2016). Application of an optimisation approach for the calibration 1 of high-fidelity building energy models to support model-2 predictive control ( mpc ) of hvac systems.

Bre, F., Silva, A. S., Ghisi, E., and Fachinotti, V. D. (2016). Residential building design optimisation using sensitivity analysis and genetic algorithm. *Energy and Buildings*, 133 :853–866.

Coakley, D., Raftery, P., and Keane, M. (2014). A review of methods to match building energy simulation models to measured data. *Renewable & Sustainable Energy Reviews*, 37 :123–141.

Hou, J., Sun, Y., Dai, X., Liu, J., Shen, X., Tan, H., Yin, H., Huang, K., Gao, Y., Lai, D., Hong, W., Zhai, X., Norbäck, D., and Chen, Q. (2021). Associations of indoor carbon dioxide concentrations, air temperature, and humidity with perceived air quality and sick building syndrome symptoms in chinese homes. *Indoor Air*, 31(4) :1018–1028.

Igel, C., Hansen, N., and Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 11 :1–28.

Le Corff, S., Champagne, A., Charbit, M., Noziere, G., and Moulines, E. (2018). Optimizing thermal comfort and energy consumption in a large building without renovation works. *2018 IEEE Data Science Workshop (DSW)*, pages 41–45.

Magnier, L. and Haghighat, F. (2010). Multiobjective optimization of building design using trnsys simulations, genetic algorithm, and artificial neural network. *Building and Environment*, 45 :739–746.

Nagpal, S., Mueller, C., Aijazi, A., and Reinhart, C. (2019). A methodology for auto-calibrating urban building energy models using surrogate modeling techniques. *Journal of Building Performance Simulation*, 12 :1 – 16.

Recht, T., Munaretto, F., Schalbart, P., and Peuportier, B. (2014). Analyse de la fiabilité de COMFIE par comparaison à des mesures. Application à un bâtiment passif. In *IBPSA 2014*, Arras, France.

Shabunko, V., Lim, C., and Mathew, S. (2018). EnergyPlus models for the benchmarking of residential buildings in Brunei Darussalam. *Energy and Buildings*, 169 :507–516.

Sterling, E., Arundel, A., and Sterling, T. (1985). Criteria or human exposure to humidity in occupied buildings. *ASHRAE Transactions*, 91 :611–622.

Zhang, H., Srinivasan, R. S., and Ganesan, V. (2021). Low cost, multi-pollutant sensing system using raspberry pi for indoor air quality monitoring. *Sustainability*.

Zhao, J., Lam, K. P., Ydstie, B. E., and Loftness, V. (2016). Occupant-oriented mixed-mode EnergyPlus predictive control simulation. *Energy and Buildings*, 117 :362–371.

# Chapitre 3

# Time series modelling for Air Quality and Energy optimization

This first chapter focuses on the building-oriented data gathered by Oze-Energies, and the introduction of statistical models and inference procedures for time-series forecasting.

We propose in Section 3.1 a meta modelling protocol of the transient simulator TRNSYS. We find that there is an extensive literature on building meta modelling, however only the most simple statistical models are usually presented in the literature dedicated to energy use and efficiency in buildings. Therefore, we compare the performance of these models with the traditional deep learning architectures for sequential data presented in the introduction, and introduce a new metamodel based on recurrent deep neural networks. An appealing feature of the proposed model is that its training procedure allows to model multiple buildings at once.

In Section 3.2, we present the advantages of substituting TRNSYS for the metamodel during the calibration and optimization tasks, and demonstrate that the computation time reduction allows for the calibration of more complex buildings, that would have been prohibitive with TRNSYS. We also describe in detail the optimization task, as well as the protocol for choosing an optimal set of usages for the buildings, leading to a reduction of up to 10% in electric consumption. Our end-to-end methodology is summarized in Figure 3.10.

Finally, in Section 3.3, we introduce the Air Quality analysis, and provide a first approach to modelling indoor relative humidity. The high uncertainty on the data and the input variables lead us to explore state space models in the next chapter.

The results presented in this chapter are adapted from the following contributions : *End-to-end deep meta modelling to calibrate and optimize energy consumption and comfort*, Cohen, M. Le Corff, S., Charbit, M., Champagne, A., Nozière, G, Preda, M., Energy and Buildings, Volume 250, November 2021.

## 3.1 Energy meta modelling

In this section, we introduce a new metamodel to predict building behaviors after a comprehensive study of several approaches from traditional RNN to a model based on a Transformer architecture Vaswani et al. (2017). The performance of this metamodel are compared both in terms of accuracy and computational efficiency with TRNSYS.

### 3.1.1 Related works

The building optimization literature has seen an increasing number of surrogate approaches, as recent sophisticated statistical models provide appealing solutions to be used in this context. In Bre et al. (2020); Reynolds et al. (2018), statistical models were trained on a dataset sampled from EnergyPlus, allowing significant computational savings during optimization. In Bre et al. (2020), the authors proposed to combine NSGA-II with an artificial neural network metamodel, here a Feed Forward Network (FFN), in order to optimize the consumption of a $83\,\mathrm{m}^2$ house. Optimization was also conducted with the original building simulator, EnergyPlus, in order to compare both results and ensure that the FFN could be used as a substitute during optimization. Similarly, Reynolds et al. (2018) proposed a FFN based meta modelling approach to reduce up to 25% the energy consumption in a small office building. EnergyPlus was used to sample a dataset for various zones of the building. The metamodel was tested using a 4-week long EnergyPlus simulation with variable set point temperatures and using an alternative weather file. An example of recurrent neural architecture as a surrogate model can be found in Ohta et al. (2020), where the authors focused on an air-conditioning optimization problem using time series.

If these articles justify the use of metamodels, the question of which type of model to choose remains. In an in-depth review, Roman et al. (2020) compares standard statistical models, such as polynomial regression, multivariate adaptive regression splines, Gaussian processes or Decision Trees, in the context of building performance simulation. Artificial Neural Networks models stand out as a particularly relevant alternative, but are often presented in their most simple, time independent form, such as the FFN used in Bre et al. (2020). Although they may yield accurate predictions in some frameworks, these neural networks handle every time step independently, and are thus not adapted to time series problems. They are usually substituted for their sequential counterparts, such as recurrent or convolutional based approaches, as demonstrated by the authors of Sendra-Arranz and Gutiérrez (2020). In their paper, they explored various architectures of Long Short Term Memory models, in order to predict HVAC consumption in buildings. Therefore, designing metamodels for building calibration and optimization is likely to benefit from such recurrent and attention-based models.

Recurrent Neural Network (RNN) were first introduced as a more suited architecture for dealing with time varying input patterns Mozer (1989). By replacing buffer based approaches with an updated context state, RNN are able to solve time series problems with short time dependencies, but are lackluster in problems requiring long term memory due to vanishing and exploding gradient Bengio et al. (1994). The Long Short Term Memory (LSTM) model proposed in Hochreiter and Schmidhuber (1997) aims at bridging that gap by enforcing error flow throughout time in the network. The LSTM architecture was modified in Cho et al. (2014) in order to simplify its implementation and improve computation times, resulting in a novel model called Gated Recurrent Unit (GRU). In parallel to these advances on recurrent architectures, Convolutional Neural Networks (CNN), rendered popular by Krizhevsky et al. (2012) for image classification, have been adapted to time series problem. The approaches proposed in Józefowicz et al. (2016); Kim et al. (2016) outperformed traditional Natural Language Processing (NLP) models by replacing the embedding layer with a character-level convolutional layer.

Recurrent and convolutional approaches coincide in that temporally close time steps data are matched together. In 2017, Vaswani et al. (2017) proposed an attention based approach to solve NLP tasks that consider the entire input sequence in parallel. The Transformer model is based on a self-attention mechanism, that computes an attention value for every element of a sequence with respect to all others to model their dependency. This attention mechanism allows to understand at each time step which input elements are crucial to predict the new state. This makes these networks more interpretable than their most widely-used recurrent counterparts such as LSTM or GRU networks and motivates a keen interest for such approaches to predict complex time series.

### 3.1.2 Notation

In the following sections, we benchmark multiple models on the following regression task : for all $t \in \{1 \cdots T\}$, let $y_t = (\zeta_t, \tau_t)$ be the vector of observations at time $t$, such as inside temperatures in $\tau_t$, heating, cooling and ventilation consumptions in $\zeta_t$. We compare predictions of $y_t$ from a set of inputs $u_t = (\lambda, \varphi_t, \delta_t, \psi_t)$, describing the building properties in $\lambda$, usage in $\psi_t$, occupation in $\delta_t$ as well as weather data in $\varphi_t$. This input vector contains $d_u = 34$ variables at each time step : 17 variables from $\lambda$, 7 from $\varphi_t$, 1 from $\delta_t$ and 9 from $\psi_t$. Note that some of these properties, such as the ones contained in $\lambda$, do not vary with time. In an attempt to keep the definition of the problem as well as the notation as simple as possible, we choose to include them in the input vector $u_t$ indexed by the time step $t$, even if they remain constant for the entirety of the time series. A detailed list of all variables used in the following experiments is available, see Appendix A.1.2.

### 3.1.3 Proposed benchmarks

In most recent works, a great deal of research activities focused on FFN as surrogate models, see Bre et al. (2020); Magnier and Haghighat (2010); Reynolds et al. (2018). Although they may lead to interesting performance during the training phase, these fully connected architectures are not well suited for time series prediction, in particular for long time spans. We ceased this opportunity to explore other approaches that have proven to be more relevant for solving time series tasks in the past few years. Therefore, we decided to evaluate the go-to architectures for time series.

— A LSTM model presented in details in Section 3.1.4.
— A bidirectional GRU (BiGRU). A definition of this model can be found in Li et al. (2021). We chose the same hyper parameters (number of layers, latent space dimensions, etc.) as for the LSTM model.
— A hybrid model mixing both convolutional and GRU layers (ConvGru). This model was inspired by Zhang et al. (2018) and consists of three one dimension convolution layers, followed by three GRU layers.
— A Feed Forward Network (FFN) with two hidden layers, as for the one used in Bre et al. (2020). At each time step $k$, the model computes an estimation of the observation $y_k$ based on $u_k$ only. Note that because this process is done independently for each time step, we are able to parallelize the sequence of predictions, leading to a reduced computation time. However, this also means that the model cannot infer time dependencies from the data.
— A Transformer model modified for inference based on long time series is also considered. The original Transformer, as well as this new model, are presented in in the following chapter.

These models have been implemented using the deep learning framework PyTorch [1], and can be found on our Github repository. [2]

#### The Transformer

The Transformer is a neural network architecture developed originally to solve Natural Language Processing (NLP) tasks. It relies on attention mechanisms to address the lack of long term memory of LSTM models, highlighted in Zhou et al. (2021) for instance. We first present the self attention layer, then detail the full architecture of the Transformer. In the following chapter, we discuss our modified implementation adapted for time series.

**Self attention.** The self attention layer aims at combining elements of the input time series $(u_1, \cdots, u_T)$ that relate to each other. This relation is learned by the network during training, and represented during inference by the Score

1. https ://pytorch.org
2. https ://github.com/maxjcohen/transformer

matrix. Let $X = (u_1, \cdots, u_T)^\top$ the input matrix with shape $(T, d_u)$ created by stacking all time steps of the input time series vertically. We start by computing three matrices $Q \in \mathbb{R}^{T \times d_q}$, $K \in \mathbb{R}^{T \times d_q}$ and $V \in \mathbb{R}^{T \times d_v}$ containing query and keys vectors with dimension $d_q$, as well as values vectors with dimension $d_v$ associated with each time step. Both $d_q$ and $d_v$ are hyper parameters, set in the original paper to $d_q = d_v = 64$. Then the score matrix, with shape $(T, T)$, represents the attention of each time step towards every other.

$$\begin{aligned} Q &= XW_q && \texttt{Queries} \\ K &= XW_k && \texttt{Keys} \\ V &= XW_v && \texttt{Values} \\ \texttt{Scores} &= d_q^{-1/2} QK^\top , \end{aligned}$$

where $\{W_q, W_k, W_v\}$ are parameters of the Transformer model, learned during the training procedure. Such transformations, referred to as self-attention, compares inputs with one another as the matrix $QK^\top$ contains all dot products between entries (columns) of $Q$ and $K$. The attention is given by

$$Attention = \texttt{softmax}\left(d_q^{-1/2} QK^\top\right) V ,$$

where the softmax function is defined, for any $z \in \mathbb{R}^d$, by

$$\texttt{softmax} : z \mapsto \left(\mathrm{e}^{z_1}, \ldots, \mathrm{e}^{z_d}\right) / \sum_{j=1}^{d} \mathrm{e}^{z_j} .$$

By computing the product between $Q$ and $K^\top$, we are evaluating the association between every element of $X$, regardless of their position in the time series. This is the main contribution of the Transformer, as it allows the model to address all time steps at the same time, replacing the previous memory mechanism. The score matrix was designed to be computed in parallel, using modern hardware, allowing for very fast computation times. This makes for a computationally interesting alternative to RNNs, where the latent state must be computed recursively. However, with this parallelisation comes a quadratic complexity in the time length. While this is usually not a problem for NLP tasks, as the length of a sentence is usually small (see Proust (1921) for an inconvenient truth), it is a major limitation in our case.

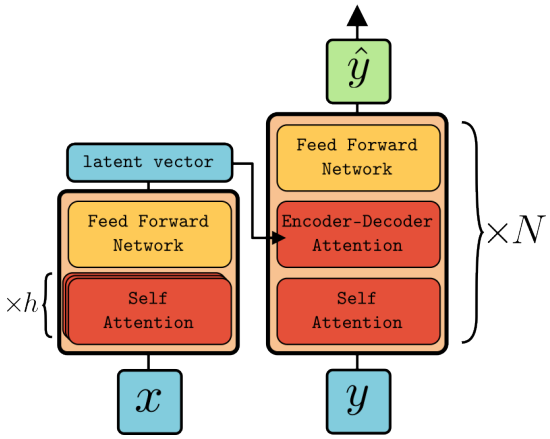
**Architecture of the Transformer.** The Transformer builds on an encoder-decoder structure, which was introduced in Cho et al. (2014), in order to push the model toward a meaningful, abstract representation of the data. By dividing the model in two almost symmetrical sections - an encoder and a decoder - with a bottleneck referred to as the latent vector, the model is constrained to perform a compression of the available information.

In the Transformer, the encoder blocks are composed of a self attention layer, as defined in the previous paragraph, followed by a two-layer Feed Forward Network (FFN) with a Rectified Linear Activation Unit (ReLU, see Bai (2022) for a review) : $\texttt{FFN}(x) = max(0, xW_1 + b_1)W_2 + b_2$, where $\{W_1, W_2, b_1, b_2\}$ are unknown parameters learnt during training. The decoder blocks follow the same architecture, with an additional "Encoder Decoder Attention" layer, where the attention score matrix is computed with respect to the latent vector. In our implementation, we chose to stack $N = 4$ encoder and decoder blocks, which is slightly lower that the value proposed in the original paper of $N = 6$, in order to reduce the size of the model and the computation cost. Similarly, we combine $h = 4$ attention layers instead of the original $h = 8$. See Figure 3.1 for a diagram of the full Transformer architecture.

During inference, the Transformer generates the latent vector in an autoregressive manner, by recursively computing predictions with the decoder section. In order to speed up evaluations of the model during the training phase,

the ground truth is directly fed to the decoder, while future observations are masked. All parameters are then estimated using gradient descent.

FIGURE 3.1 – The original Transformer contains $N$ encoder and decoder layers, each one combining $h$ self attention blocks.




**The Transformer adapted to Time Series**

Because of its quadratic complexity in the time dimension, the Transformer is too computationally expensive for our use case, as time series can span up to 672 hours long. However, the self attention mechanism being appealing to take into account long-term dependencies, we propose to adapt the Transformer architecture for long time series. In the proposed benchmark, the input sequence of the attention layers is decomposed using a rolling window, and each split is treated independently and in parallel. Let $p$ be the size of this attention window, and $\kappa$ a stride, we define a set of input matrices as follows : for all $0 \leq i \leq \lfloor (T-p-1)/\kappa \rfloor$,

$$x_i = (u_{i\kappa+1}, \cdots, u_{i\kappa+p+1})^\top .$$

Following the same equations as for the original Transformer, presented in Section 3.1.3, we compute the attention for each subsection of the input time series, and concatenate the result.

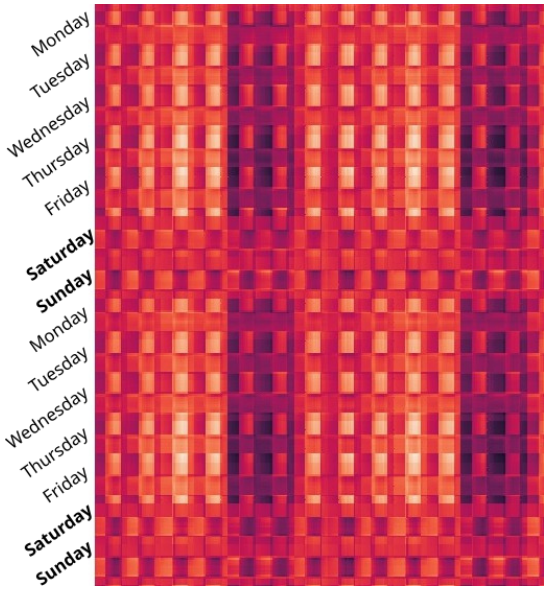
In addition to greatly reducing the computation time for long sequences, we introduced a local attention mechanism which improved the model performance. After training our model on the metamodel dataset, see Section 3.1.5, we were able to produce attention maps presented in Figure 3.2.

### 3.1.4 Our metamodel

Our metamodel is built following a classical recurrent mechanism. We assume that the observations can be modeled using a sequence of hidden states $(h_t)_{t\geq 0}$. At each time step $t$, the observations are defined as random transformations of the input $u_t$ and the hidden state $h_{t-1}$ depending on the past values $(u_1, \ldots, u_{t-1})$. We use as a backbone a many to many RNN architecture, and denote by $h_t^\ell$ and $x_t^\ell$ the hidden state and input of layer $1 \leq \ell \leq L$ at time step $t$, with the additional convention $x_t^0 \equiv u_t$. The hidden state is traditionally initialized as the zero vector, $h_0^\ell \equiv 0$ for all $1 \leq \ell \leq L$.

In the original and most simple definition of a RNN, the hidden state is computed recursively as $h_t^\ell = \tanh(W_{ih}^\ell x_t^\ell + W_{hh}^\ell h_{t-1}^\ell + b_h^\ell)$, where $W_{ih}$, $W_{hh}$ and $b_h$ are the weight matrices and bias learned during training, and initialized with random values. Our metamodel is based on a LSTM architecture and replaces the update of the hidden state by the

FIGURE 3.2 – Attention map for building behavior prediction, for two consecutive weeks. Each pixel corresponds to a scalar value in the attention map produced by the Transformer model during inference, with darker colors indicating lower values and brighter colors higher values. We can identify day and night cycles, as well as week and weekend.




following state equations :

$$
\begin{aligned}
\Gamma_i^\ell &= \sigma(W_{xi}^\ell x_t^\ell + W_{hi}^\ell h_{t-1}^\ell + b_i^\ell)\,, \\
\Gamma_f^\ell &= \sigma(W_{xf}^\ell x_t^\ell + W_{hf}^\ell h_{t-1}^\ell + b_f^\ell)\,, \\
\Gamma_o^\ell &= \sigma(W_{xo}^\ell x_t^\ell + W_{ho}^\ell h_{t-1}^\ell + b_o^\ell)\,, \\
\tilde{c}_t &= \tanh(W_{xc}^\ell x_t^\ell + W_{hc}^\ell h_{t-1}^\ell + b_c^\ell)\,, \\
c_t^\ell &= \Gamma_f^\ell * c_{t-1}^\ell + \Gamma_i^\ell * \tilde{c}_t\,. \\
h_t^\ell &= \Gamma_o^\ell * \tanh c_t^\ell\,.
\end{aligned}
$$

where $x_t^\ell \equiv h_t^{\ell-1}$ is the hidden state computed in the previous layer. An additional fully connected layer is added on top of the RNN architecture, following results presented in Sendra-Arranz and Gutiérrez (2020), to mobtain the observation model :

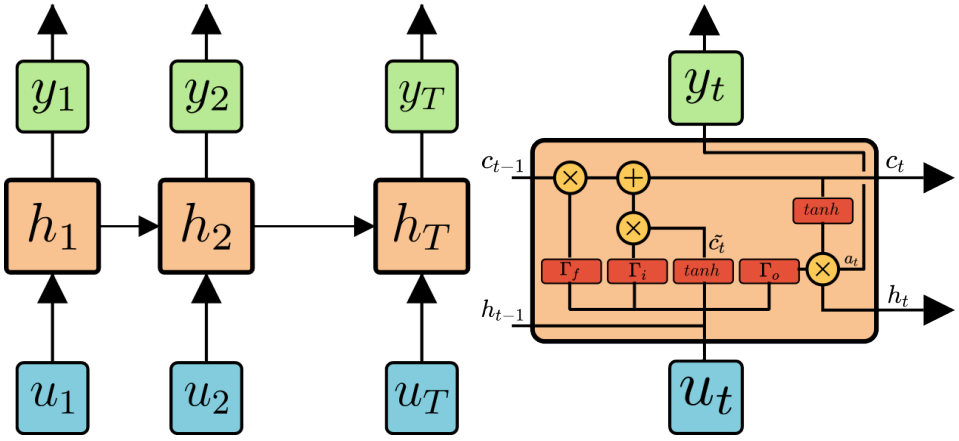
$$y_t = \sigma(W_y h_t^L + b_y) + \varepsilon_t\,,$$

where $\sigma$ is the sigmoid activation function $\sigma : x \mapsto (1+e^{-x})^{-1}$ and where $(\varepsilon_t)_{t=1}^T$ are independent centered Gaussian variables with covariance matrix $\Sigma_y$. The architecture is represented in Figure 3.3. The unknown parameters to be estimated during the training phase of the metamodel are

$$\theta = \left\{ \left(W_{xi}^\ell, W_{hi}^\ell, W_{xf}^\ell, W_{hf}^\ell, W_{xo}^\ell, W_{ho}^\ell, W_{xc}^\ell, W_{hc}^\ell, W_y, b_i^\ell, b_f^\ell, b_o^\ell, b_c^\ell, b_y\right)_{1\leq\ell\leq L} \right\}\,.$$

The model is trained to produce accurate predictions by tuning its parameters $\theta$, usually referred to as weights, through an iterative back propagation algorithm, where predictions are compared to the ground truth $y_t$. Since we choose the Mean Square Error as a cost function, the covariance matrix $\Sigma_y$ is not estimated during training.

FIGURE 3.3 – Our metamodel architecture (left), and a detailed LSTM cell (right). The LSTM cell improves on the classic RNN by introducing a cell state $c_t$ supposed to carry long term memory, without additional alterations, throughout the sequence. The three input gate $\Gamma_i$, forward gate $\Gamma_f$ and output gate $\Gamma_o$ determine whether information in both hidden state $h_k$ and cell state $c_k$ should be carried away or discarded.




### 3.1.5 Dataset sampling

The training dataset is sampled by exploring the input space of the simulator. We chose TRNSYS as it was the simulator used by Oze-Energies, but any simulator can be used to train the metamodel. We define ranges for each input variable in $\lambda$ and $(\psi_k)_{k\geq 0}$ with the help of energy managers, such as highest and lowest scheduled temperature, or the earliest and latest hours of arrival of occupants, see the appendices for a complete list of these ranges. Because our dataset aims at capturing multiple buildings, these ranges are not centered around a specific set of variables, but rather cover all possible values across our cluster of buildings. In addition, real weather data $(\varphi_k)_{k\geq 0}$ acquired between June and August 2020 around the Parisian area where used to obtain a dataset consistent with the real buildings.

In our numerical experiments, we chose a uniform sampling method over the ranges for each building and weather variable. This allows us to easily split the dataset uniformly into training and testing sets, which is crucial to validate the metamodel.

A total of 15,000 training examples were sampled, an example being 672 hours i.e. 28 days, which we will refer to as to as a month. During the training phase, the parameters of each metamodel described in Section 3.1.3, and called $\theta$ in the detailed case of the RNN approach, are estimated based on this dataset. The metamodels compared in this section are defined with a latent dimension of $d_{emb} = 64$ and a total of $L = 4$ layers. Hyper parameters, such as learning rate, dropout, number of epochs or batch size, were chosen by cross validation.

### 3.1.6 Training

During training, for each example, we compute the Mean Squared Error (MSE) loss, and combine consumption and temperature errors :

$$\mathrm{MSE_T} = \frac{\sum_{t=1}^{T}(\tau_t - \widehat{\tau}_t)^2}{\sum_{t=1}^{T}(\tau_t - \overline{\tau})^2} \quad \text{and} \quad \mathrm{MSE_Q} = \frac{\sum_{t=1}^{M}(\zeta_t - \widehat{\zeta}_t)^2}{\sum_{t=1}^{M}(\zeta_t - \overline{\zeta})^2}$$
$$\mathrm{loss} = \beta \mathrm{MSE_T} + (1-\beta)\mathrm{MSE_Q}\,,$$

where $\bar{\tau} = T^{-1}\sum_{t=1}^{T}\tau_t$, $\bar{\zeta} = T^{-1}\sum_{t=1}^{T}\zeta_t$ and $(\hat{\tau}_{1:T}, \hat{\zeta}_{1:T}) = f_{\texttt{building}}(\lambda, \psi_{1:T}, \delta_{1:T}, \varphi_{1:T})$ and $f_{\texttt{building}}$ are all the architecture proposed in the previous sections. In the experiments below, as the inside temperatures and all consump-

TABLE 3.1 – Metrics (means and standard deviations) of the metamodels on the validation splits. The best mean values are displayed in bold (the lowest losses and mean squared errors). $\mathrm{Time}$ is the computation time to run a single simulation through the network, and was estimated by averaging 100 inferences. Our selected architecture is detailed in Section 3.1.4, and achieves the best performance on all metrics, while still being on par with most models in computation time.

| | BiGRU | Transformer | Ours | ConvGru | FFN |
|---|---|---|---|---|---|
| $\mathrm{Loss}$ $(\times 10^{-4})$ | $2.05\ (1.61)$ | $2.72\ (2.88)$ | **1.65 (1.47)** | $2.26\ (2.04)$ | $90.5\ (41.1)$ |
| $\mathrm{MSE_T}$ $(\times 10^{-5})$ | $1.00\ (1.53)$ | $1.49\ (2.75)$ | **0.820 (1.44)** | $1.20\ (2.05)$ | $39.5\ (39.7)$ |
| $\mathrm{MSE_Q}$ $(\times 10^{-4})$ | $3.84\ (2.05)$ | $4.12\ (3.01)$ | **2.75 (1.72)** | $3.53\ (1.77)$ | $172\ (60.2)$ |
| $\mathrm{MSE_T^{occ}}$ $(\times 10^{-5})$ | $4.60\ (7.16)$ | $6.56\ (12.2)$ | **3.79 (6.81)** | $5.89\ (9.91)$ | $176\ (189)$ |
| $\mathrm{MSE_Q^{occ}}$ $(\times 10^{-4})$ | $1.45\ (1.00)$ | $2.07\ (1.80)$ | **1.22 (0.878)** | $1.85\ (1.23)$ | $113\ (50.8)$ |
| $\Delta_{Q^{\mathrm{Tot}}}$ $(\times 10^{-3})$ | $4.03\ (11.7)$ | $20.1(12.4)$ | **2.46 (12.0)** | $14.9\ (16.2)$ | $182\ (68.8)$ |
| $\mathrm{Time}$ $(\times 10^{-2}s)$ | 6.46 | 4.52 | 6.51 | 6.77 | **0.341** |

tions are normalized, we chose the non informative value $\beta = 0.5$. We chose the Adam optimizer Kingma and Ba (2015) and all simulations were computed on a single 1080TI GPU card. Visuals sample of the metamodel predictions are available, see Appendix A.1.1.

### 3.1.7 Validation

Validation is essential to identify any potential overfit of the model on the training dataset. In this study, we implement a traditional cross-validation, whereby the dataset is split into $k$ folds, and the model is trained on the $k-1$ first folds and evaluated on the last. We average this validation score by iteratively changing the validation fold, as detailed by the authors of Seyedzadeh et al. (2020), with $k = 5$. This method ensures that our model is always evaluated on unseen data, which demonstrates its generalization capability and avoids any potential bias of the validation split. Table 3.1 displays the mean values and standard deviations of the loss function of this cross validation at the end of the training procedure. The table also displays the mean squared error $\mathrm{MSE_T}$ (resp. $\mathrm{MSE_Q}$) on the temperatures (resp. consumptions) only, as well as these same metrics computed only during occupation time : $\mathrm{MSE_T^{occ}}$ and $\mathrm{MSE_Q^{occ}}$. For a global consumption evaluation, we compute the absolute relative error on the cumulative consumptions $\Delta_{Q^{\mathrm{Tot}}}$.

## 3.2 Energy Optimization in real buildings

The experiments conducted in this thesis to analyze the performance of the trained metamodel focused on the optimization of two buildings in the Parisian region. Each one is represented by a single thermal zone.

— Stanley is a $18,512\,m^2$ building. It is delimited by four vertical walls of dimension $2,314\,m^2$, $1,917\,m^2$, $2,123\,m^2$ and $1,725\,m^2$, as well as a roof and ground of dimension $2,304\,m^2$. The main insulator is a $10\,cm$ layer of polystyrene. It was built in 1983.

— Livingstone is a $13,594\,m^2$ building, including 4 vertical walls with respective areas $1,678\,m^2$, $1,274\,m^2$, $1,281\,m^2$ and $1,252\,m^2$, a horizontal roof and a horizontal ground of dimension $4,653\,m^2$ and $4,286\,m^2$. The main insulator is a $8\,cm$ layer of polyurethane. It was built in 2006.

Based on a commonly used rule, it is assumed that $2/3$ of the full area is occupied by people. Assuming that each occupant requires $12\,m^2$, this allows to set the initial values for the number of occupants and the number of PCs (set to 1.2 times this value) in the building during occupancy hours. These values are assumed to be known

and fixed and used to sample the training dataset.

### 3.2.1 Calibration

During the training phase, metamodel parameters are estimated by minimizing the loss function on the simulated dataset which corresponds to various configurations associated with choices of $\lambda$ and $(\psi_k, \varphi_k)_{1\leq k\leq T}$. Because this dataset is sampled from a simulation model, we trained the metamodel ignoring real building related noise and measurement errors. Additionally, both TRNSYS and our metamodel require an estimate of the parameters $\lambda$ that cannot be properly identified for each building, especially without any site work. By comparing the metamodel predictions to real historic data during the calibration phase, we search for a set of building related parameters that best match reality. During this step, the weights $\theta$ of the metamodel are frozen, meaning that we no longer update each weight matrix of the neural network.

We can compute, for each given set of input parameters $\lambda$ and $(\psi_k, \varphi_k)_{1\leq k\leq T}$, the difference between estimated and real historical data. Goodness of fit of the model is measured with by combining two calibration criteria toward temperature and consumption, as presented in 2.1. Following the performance evaluation criteria in Ajib (2018), we define the normalized Mean Square Error for both :

$$\mathcal{L}^{\texttt{calib}}_{\texttt{temperature}}(\tau, \hat{\tau}) = \frac{\sum_{t=1}^{T}(\tau_t - \hat{\tau}_t)^2}{\sum_{t=1}^{T}(\tau_t - \bar{\tau})^2}\,,$$
$$\mathcal{L}^{\texttt{calib}}_{\texttt{energy}}(\zeta, \hat{\zeta}) = \frac{\sum_{t=1}^{T}(\zeta_t - \hat{\zeta}_t)^2}{\sum_{t=1}^{T}(\zeta_t - \bar{\zeta})^2}\,,$$

where

$$\bar{\tau} = T^{-1}\sum_{t=1}^{T}\tau_t\ ,\bar{\zeta} = T^{-1}\sum_{t=1}^{T}\zeta_t\ ,\text{and } (\hat{\tau}_{1:T}, \hat{\zeta}_{1:T}) = f_{\texttt{building}}(\lambda, \psi_{1:T}, \delta_{1:T}, \varphi_{1:T}).$$

Because this is a non differentiable problem, the cost function cannot be minimized using a stochastic gradient descent algorithm as in the training step ; instead we use the CMA-ES algorithm Hansen et al. (2003), an evolutionary algorithm designed to solve constrained non-convex optimization problems. In our experiments, the variables we adjust for fitting are constrained by the same ranges defined in the data sampling section. The algorithm is implemented by the author of the paper in the pycma library. [3]

Following traditional methodology in building calibration, we measure the performance of the calibrated model with the Mean Bias Error ($\mathrm{MBE}$) and Coefficient of variation of the Root Mean Square Error ($Cv(\mathrm{RMSE})$) criteria. For any sequence of temperature $(\tau_t)_{1\leq t\leq T}$ associated with predictions $(\hat{\tau}_t)_{1\leq t\leq T}$, these quantities are defined as follows :

$$\mathrm{MBE}_T(\%) = 100\frac{\sum_{t=1}^{T}(\tau_t - \hat{\tau}_t)}{\sum_{t=1}^{T}\tau_t}\,,\quad Cv(\mathrm{RMSE})_T(\%) = \frac{100}{\bar{\tau}}\left(\frac{\sum_{t=1}^{T}(\tau_t - \widehat{\tau}_t)^2}{T}\right)^{1/2}. \tag{3.1}$$

The criteria $\mathrm{MBE}_Q$ and $Cv(\mathrm{RMSE})_Q$ are computed similarly for consumptions instead of temperature.

In a detailed review of calibration methods, the authors of Fabrizio and Monetti (2015) have gathered the international recommended ranges regarding these criteria, when validating a calibrated model. Regardless of the simulation program, the $Cv(\mathrm{RMSE})$ should fall within $\pm 20\%$, and the $\mathrm{MBE} \pm 5\%$ when considering hourly calibrations. As shown in Table 3.2, our results for both consumptions and indoor temperatures calibration are well within these guidelines.

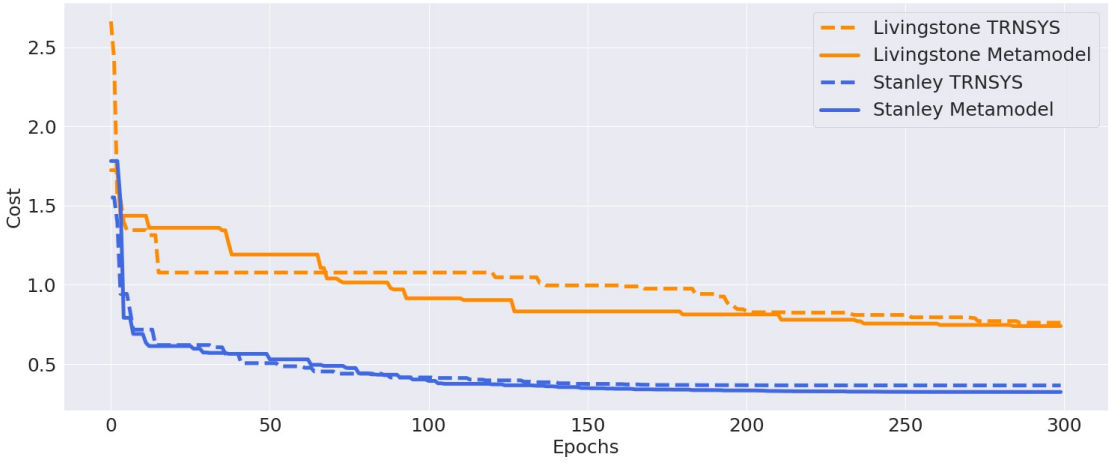
3. https ://github.com/CMA-ES/pycma

Calibration was run for both the metamodel and TRNSYS for a maximum of 3 hours. We chose the non informative value $\beta = 0.5$ for Equation 2.1. As shown in Table 3.2, we can achieve satisfactory results for Stanley in this timespan, as both model converge to close values for both the $Cv(\mathrm{RMSE})$ and $\mathrm{MBE}$. Figure 3.5 displays both models calibration results, compared to real data. On the other hand, TRNSYS calibration of Livingstone is sensibly below the results obtained with the metamodel, as calibration did not converge in the available time, see Figure 3.4. The calibration of the metamodel reached convergence but with a tremendous number of epochs, that would have required to run TRNSYS for about 10 hours in order to get similar performance. As a comparison, we calibrated the metamodel for the same number of epochs as TRNSYS, and obtained similar results. This experiment comforts the idea that TRNSYS and the metamodel behave similarly after the calibration step, but the much shorter computation time of the metamodel allows us to better calibrate complex buildings, such as Livingstone. See Figure 3.6 for a visualization of the TRNSYS and metamodel calibration after one hour.

TABLE 3.2 – Calibration metrics for Stanley and Livingstone buildings, see 3.1. Convergence is reached for Stanley after $300$ iterations, which is not enough for Livingstone, as displayed in Figure 3.4. This table demonstrates that the metamodel and TRNSYS perform similarly when calibrated for the same number of iterations, although the metamodel is much faster. Additionally, only the metamodel is able to reach convergence for Livingstone in a reasonable time frame.

| | $\mathrm{MBE}_Q$ | $Cv(\mathrm{RMSE})_Q$ | $\mathrm{MBE}_T$ | $Cv(\mathrm{RMSE})_T$ | Iterations | Computational time |
|---|---|---|---|---|---|---|
| **Stanley** | | | | | | |
| Metamodel | -0.627 | **11.0** | 0.134 | **1.20** | 300 | 2mn |
| TRNSYS | -0.409 | 12.1 | -0.264 | 1.24 | 300 | 3h |
| **Livingstone** | | | | | | |
| Metamodel | -0.690 | **14.2** | -0.0551 | **1.29** | 10000 | 1h |
| Metamodel | -0.574 | **14.2** | -0.413 | 1.95 | 300 | 2mn |
| TRNSYS | -1.08 | 15.8 | 0.156 | 1.96 | 300 | 3h |

FIGURE 3.4 – Calibration cost evolution for the metamodel and TRNSYS (see 2.1), on Livingstone and Stanley. Both models were calibrated for 300 epochs, which is enough to reach convergence for Stanley, but not Livingstone.




**Validation.** The metamodel will assist the decision process for building management by simulating thermal behavior of future weeks. Because the calibration process requires real data, the metamodel is calibrated on several past

FIGURE 3.5 – Consumption and temperature simulations after calibration, for both the metamodel and TRNSYS, for Stanley.

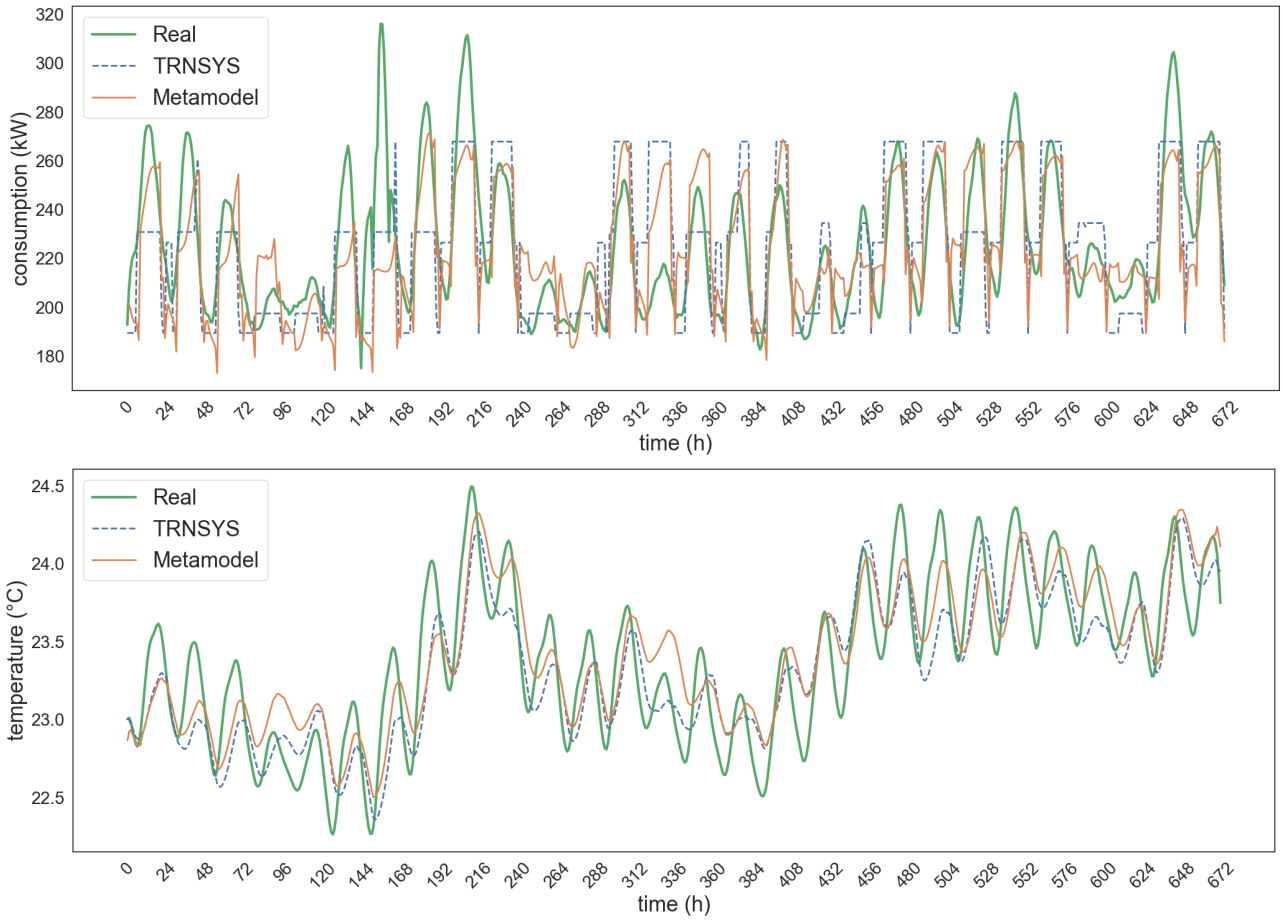
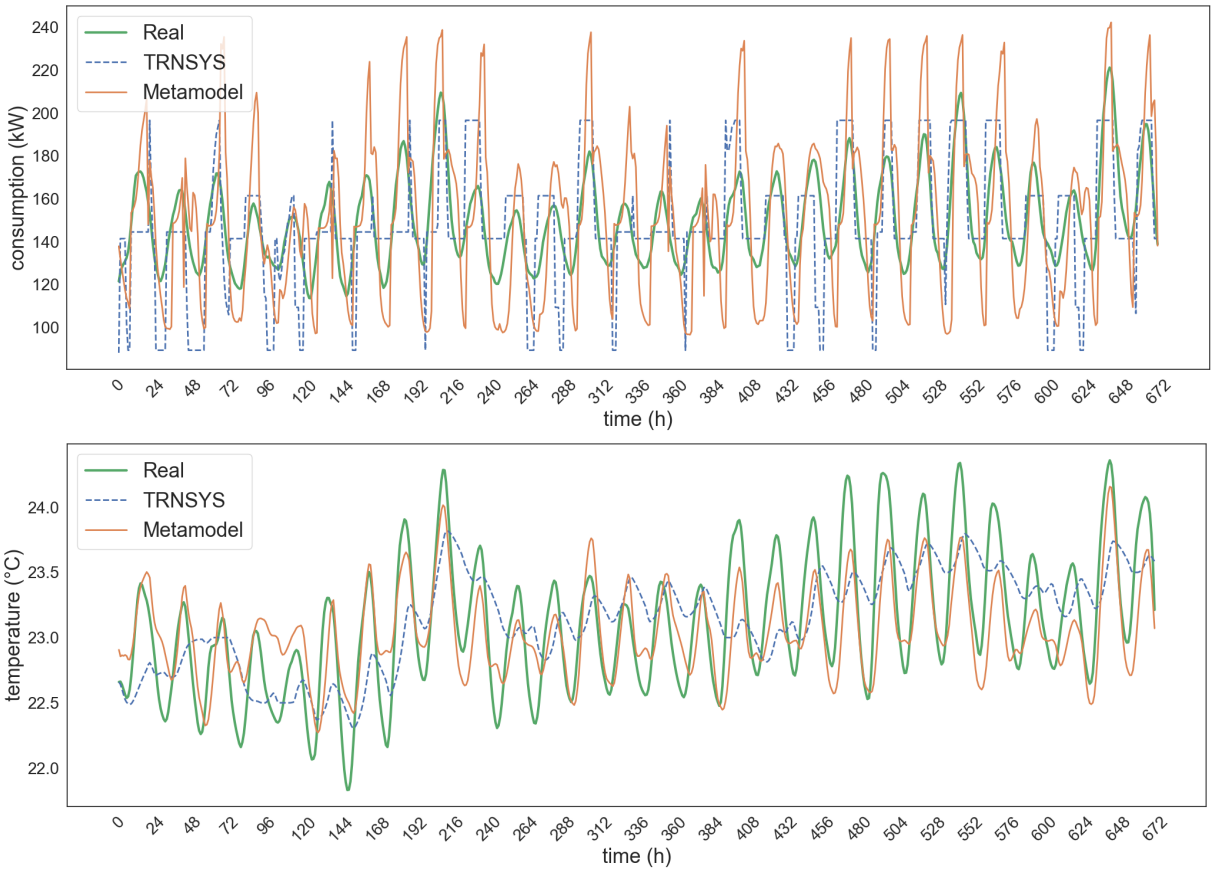


FIGURE 3.6 – Consumption and temperature simulations after calibration, for both the metamodel and TRNSYS, for Livingstone.



weeks, in order to capture the real building behavior in a situation as close as possible to the future period we aim to match.

We validate the calibration phase using two successive weeks, by applying the calibrated settings to the two following weeks, with fresh weather data, and compare the results to the true observed values. The results are displayed in Figure 3.7 and display encouraging results, as the simulation of the metamodel on the two unseen weeks is able to match most trends is both consumption and indoor temperature.

### 3.2.2 Optimization

After a successful calibration, the metamodel is supposed to have correctly estimated building parameters $\lambda$, enabling it to accurately reproduce the thermal exchanges of the real building, as confirmed by the validation step.

The parameters $(\psi_k)_{1\leq k\leq T}$ associated with the HVAC system can then be optimized for a given set of weather data $(\varphi_k)_{1\leq k\leq T}$. The optimization tasks consists in finding a set a usage related parameters that reduce consumption while keeping the same level of comfort. Optimizing energy consumption requires minimizing two conflicting objectives, making it impossible to find a solution that optimizes both objectives simultaneously. Instead, we search for optimal compromises between energy consumption and comfort, and plot each proposition to form a Pareto front, see Figure 3.8. Combinations of energy consumption and comfort are unreachable below the Pareto front, and suboptimal above ; we always aim at sampling points at the intersection. Indeed, for any such optimal compromise, we can always get a higher level of comfort, for the price of a higher consumption. The consumption criteria is the energy load during the month ; the comfort criteria is the gap between indoor temperature and a constant reference temperature $T^*$ :

$$\mathcal{L}^{\texttt{optim}}_{\texttt{temperature}}(T^\star, \tau) = \sum_{t\in T_{\mathrm{Occ}}} (T^\star - \tau_t)^2 \,,$$
$$\mathcal{L}^{\texttt{optim}}_{\texttt{energy}}(\zeta) = \sum_{t=1}^{T} \zeta_t^2 \,,$$

with

$$(\hat{\tau}_{1:T}, \hat{\zeta}_{1:T}) = f_{\texttt{building}}(\lambda, \psi_{1:T}, \delta_{1:T}, \varphi_{1:T}) \,,$$

where $T^* = 22.5°C$ and $T_{\mathrm{occ}}$ is a subset of the daytime hours, where the building is considered to be occupied and the target temperature should be met. Following recent works in building energy optimization, we search for a set of optimal parameters using NSGA-II, see Deb et al. (2000), another evolutionary algorithm, but adapted to multi objective problems. An implementation can be found in the Pygmo [4] library. We run the optimization for 3000 iterations (2 hours). Results can be viewed as a Pareto front which is given in Figure 3.8 for the second month used in the calibration process. As observed during calibration, this process can take a colossal number of iterations before achieving satisfactory results, once again justifying the use of a much faster metamodel. The predicted time series associated with the BMS parameters selected in Figure 3.8 are given in Figure 3.9. The relative gain, as well as the expected energy savings for both building are available in Table 3.3.

4. https ://esa.github.io/pygmo2/

FIGURE 3.7 – Consumption and temperature simulations after calibration on two weeks (top), simulation on the two following weeks for the same parameters (bottom), for Stanley. Although curves for the validation weeks are not matched perfectly, the metamodel is able to capture most trends of both consumption and indoor temperature. The remaining difference can be explained by the absence of real building usage settings $\psi$ for the calibration week. This experiment comforts our assumption that the calibration step leads to a correct estimation of building parameters $\lambda$.

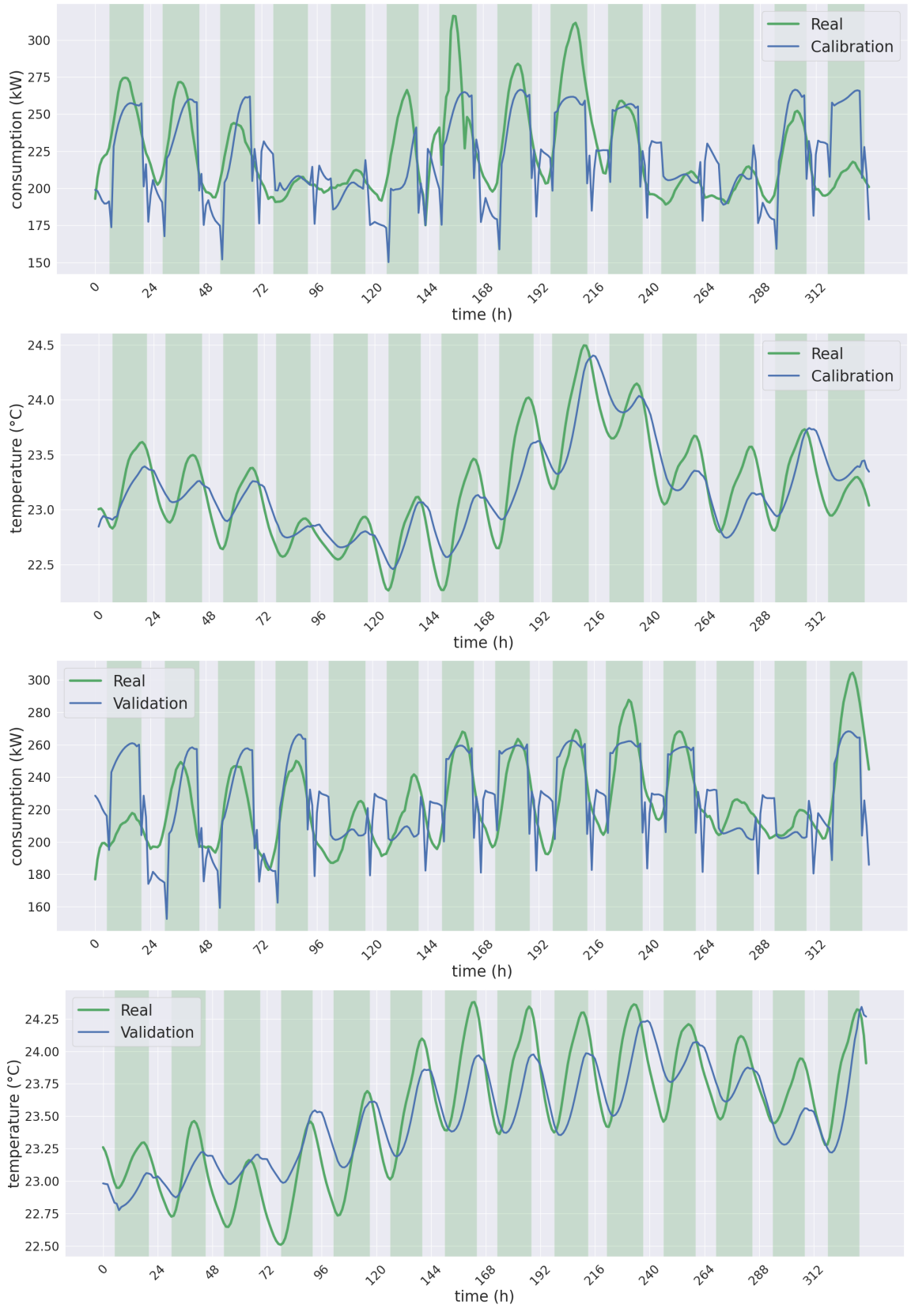


TABLE 3.3 – Energy gain after optimization. Relative gain represents the energy load reduction between calibration and optimization steps, when maintaining the initial level of comfort. We then apply this coefficient to the real monthly consumption to obtain the reduction forecast in MWh. We also provide a more interesting reduction obtained by reducing the comfort criteria by $0.5$°C.

| | relative gain (%) | prevision (MWh) | relative gain / $0.5°$ C (%) | prevision / $0.5°$ C (MWh) |
|---|---|---|---|---|
| Stanley | 5.32 | 8.05 | 10.5 | 15.9 |
| Livingstone | 9.92 | 9.96 | 17.3 | 17.3 |

FIGURE 3.8 – Pareto front after optimization for the Stanley (left) and Livingstone (right) building. We select the point of closest equivalent comfort, corresponding to a 5.3% (Stanley) and 9.9% (Livingstone) reduction in consumption. Combinations of energy consumption and comfort are unreachable below the Pareto front, and suboptimal above ; we always aim at sampling points at the intersection.

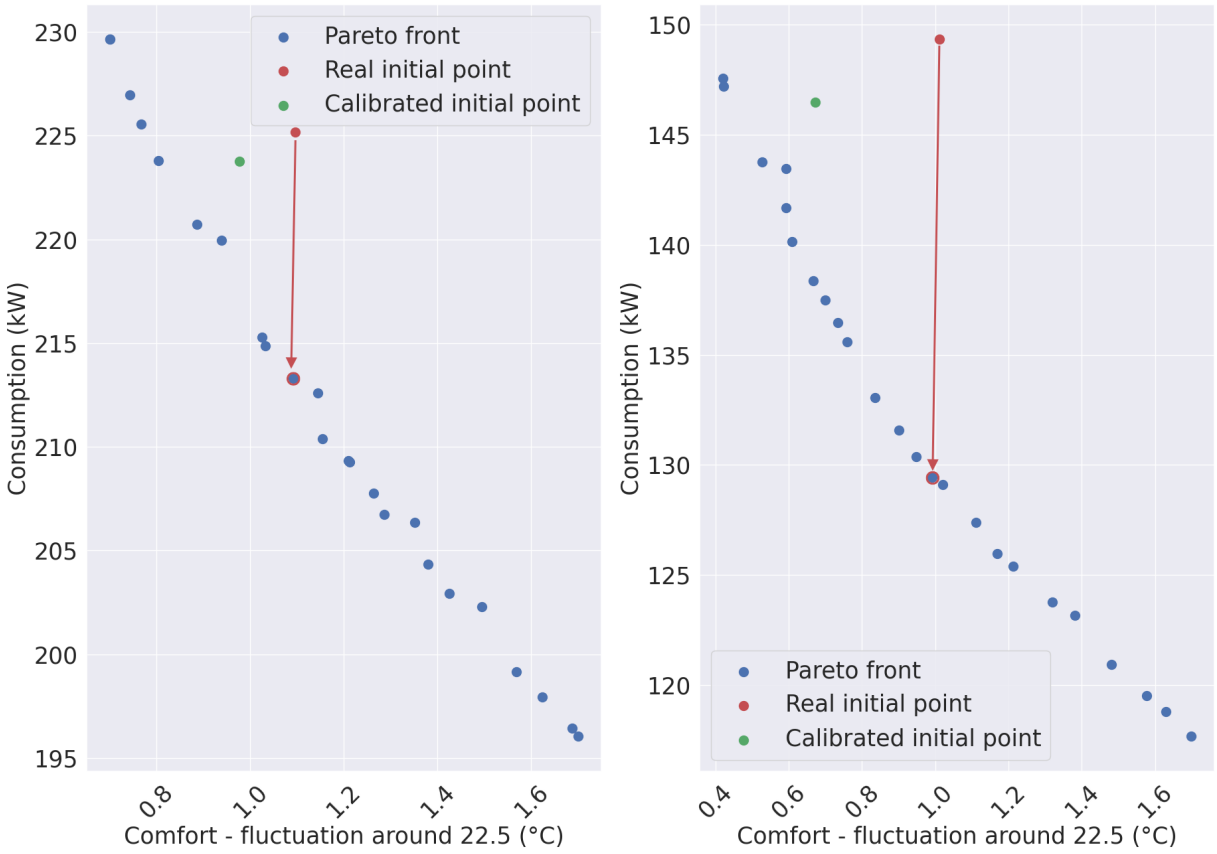


FIGURE 3.9 – Consumption and temperature simulations after optimization (metamodel) for the Stanley building.

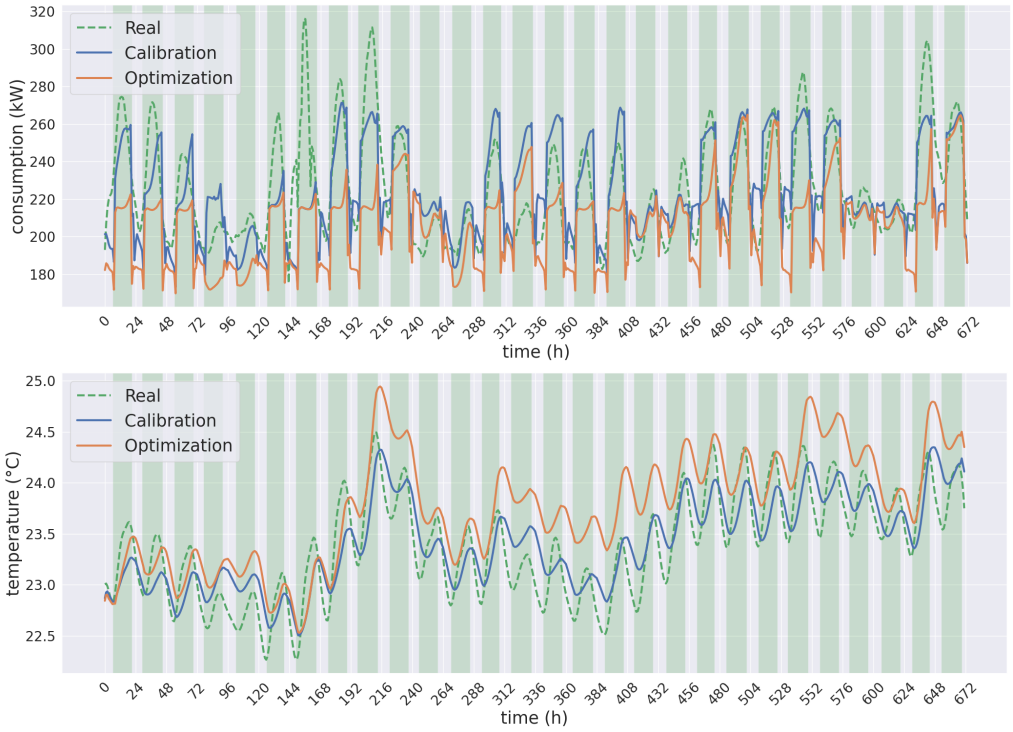
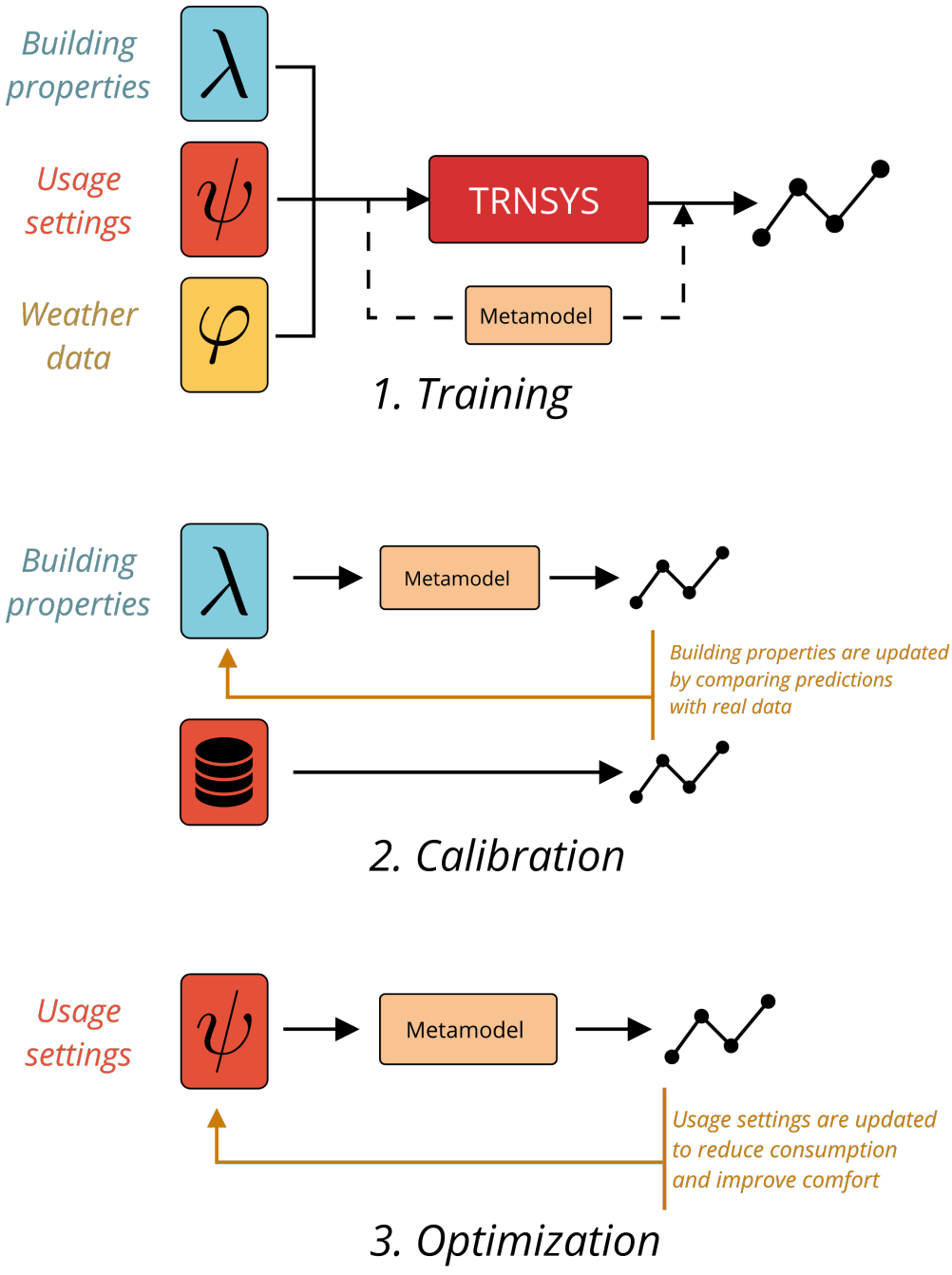


FIGURE 3.10 – Our end-to-end methodology for metamodel training, calibration and optimization.



## 3.3 Air quality modelling

In the previous sections, we proposed an end-to-end methodology to optimize energy loads and improve comfort. Because regulating indoor temperature, through heating and cooling, is by far the most energy hungry aspect of comfort, we neglected all other variables. We now address a more complete analysis of building air quality control, and propose two initial naive approaches. The limitations arising from these methods lead us to develop more complex strategies in the next chapter.

### 3.3.1 Definition

The Indoor Air Quality measures the impact of indoor air conditions on health and well being. A wide array of chemicals and conditions are involved in its analysis, as detailed in Zhang et al. (2021). In this study, we constrain ourselves to the data gathered by Oze-Energies : $CO_2$ levels, relative humidity and indoor temperature.

$CO_2$ concentration in the air is measured by ppm (parts per million) varying between 360 ppm and 412 ppm in fresh air. Indoor, it should be kept below the 1000 ppm threshold Persily (2015), as high $CO_2$ concentrations are linked to symptoms of an air quality disease known as Sick Building Syndrome detailed in Hou et al. (2021). However, it is mainly impacted by the current occupation of the building, as highlighted in Madureira et al. (2016), which is unknown to us. Air quality regulation can still be achieved by tracking $CO_2$ evolution in the building, through sensors such as the ones described in Section 2.2.1, and acting whenever these levels get too high. In all following experiments regarding air quality in this thesis, we therefore consider $CO_2$ levels as a surrogate variable for the occupation of the building.

Humidity has an impact on both health and well being. Results in Hou et al. (2021) show that a higher relative humidity in Chinese homes were associated with higher percentage of perceived moldy odor and humid air, while lower levels lead to perceived dry air. From a health perspective, maintaining relative humidity between the recommended ranges prevents the spread of bacteria, viruses, and fungi among others, as described in the Sterling Chart in Figure 2.4. Indoor humidity has been a known health hazard for quite some time, as the Bible states that living in buildings with dampness problems ('plague of leprosy') is dangerous to your health (Leviticus 14, 34–57). Yet these concerns are relevant even for newly built dwellings, which may be well isolated, but still not allow for optimal relative humidity management, as showed in Ade and Rehm (2021).

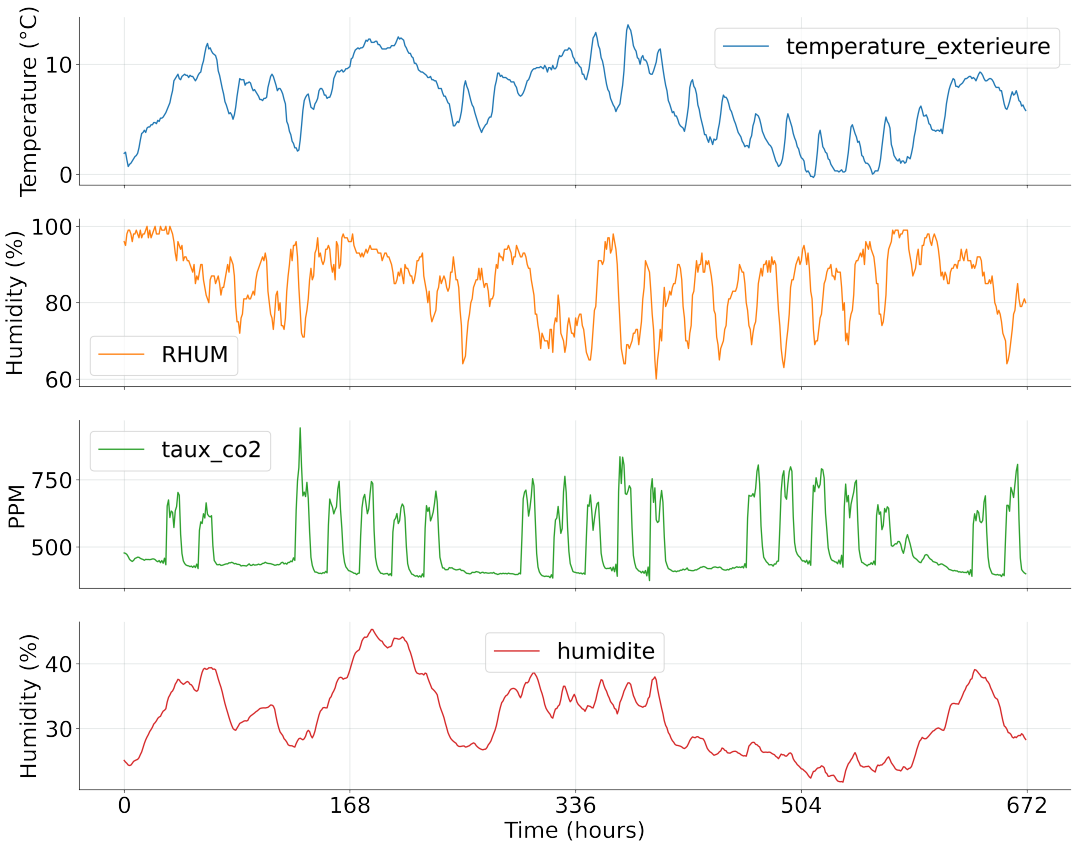
### 3.3.2 Experiments

In the subsequent paragraphs, we benchmark two statistical models applied to relative humidity forecast. Consider the regression task where the objective is to estimate the hourly indoor humidity, given by a sequence of observations $y_t$, using as inputs $u_t$, a combination of weather data as described in Section 2.2.1 and $CO_2$ levels, for $t \in \{1 \cdots T\}$. In Figure 3.11 we present a visualization of this dataset.

### 3.3.3 Deterministic neural network

We start by evaluating a traditional deterministic neural network for time series, by using the same LSTM architecture as presented for indoor temperature and consumption in Section 3.1.4. At each time step $t > 1$, a hidden state $h_t$ and a cell state $c_t$ are computed from the inputs : $c_t, h_t = \text{LSTM}_{\texttt{hidden}}(u_t, h_{t-1}, c_{t-1})$, with $c_0 \equiv h_0 \equiv 0$. Then, we apply a non linear transformation to map hidden states to observations : $y_t = \text{LSTM}_{\texttt{observation}}(h_t) + \epsilon_t$, where $(\epsilon_t)_{1 \leq t \leq T}$ are independent identically distributed centered Gaussian random variable with fixed variance. The parameters of the model are estimated by gradient descent, by minimizing the MSE.

FIGURE 3.11 – Relative Humidity dataset. Visualization of a 4-week sample of the dataset. We aim at modelling indoor humidity based on the outdoor temperature, humidity (RHUM), aux indoor $CO_2$ levels.

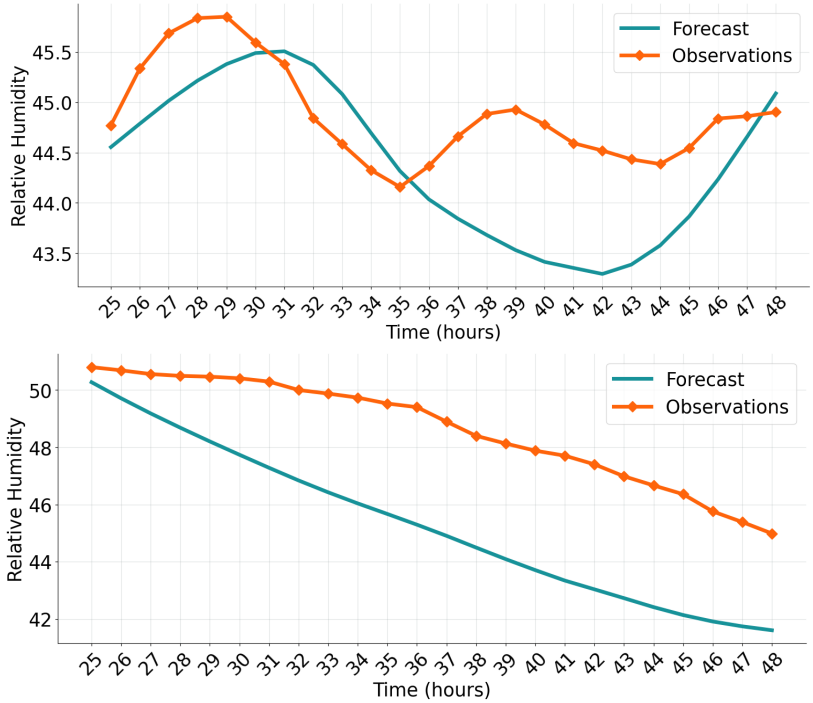



Unlike the metamodel training of the previous section, the LSTM applied to relative humidity is not able to model local variations of the observations. Although the general trend of the data was learned, hour to hour changes cannot be reconstructed by the model, as seen in Figure 3.12 where we compared two samples from the validation set to the corresponding prediction of the model. Additional samples can be found in Appendix A.1.3. We believe that this discrepancy can be explained by external factors, not represented in the input data, such as ventilation settings, or an occupant opening a window. In this context, it is not surprising that our model does not provide predictions perfectly matching the historic ground truth ; however, when inferring future weeks, it is important to have access to a measure of the model's uncertainty. For this reason, we turn to statistical models taking into account this uncertainty, in order to model the distribution of the observation, rather than simply providing a single prediction.

### 3.3.4 Hidden Markov Model

We compare our initial LSTM with a discrete Hidden Markov Model (HMM), where hidden states $(z_t)_{t=1}^T$ take values in a finite set $\{1, \cdots, K\}$. We consider that the number $K$ of states is fixed. At each time step $t$, we assume that the observation $y_t$, conditionally on the hidden state $z_t = k \in \{1, \cdots, K\}$, is Gaussian distributed with mean $\mu_k \in \mathbb{R}$ and standard deviation $\sigma_k > 0$. The hidden states are modelled as a homogeneous Markov chain, characterized by the transition matrix $A$ such that for all $t > 1$, and for all $1 \leq k, j \leq K$, $p(z_t = k | z_{t-1} = j) = a_{kj}$ with $\sum_{j=1}^K a_{kj} = 1$ and $a_{kj} > 0$. The initial probabilities are defined as $p(z_1 = k) = \pi_k$ with $\sum_{k=1}^K \pi_k = 1$ and $\pi_k > 0$. The observations

FIGURE 3.12 – Prediction of Relative Humidity given observations on two 24 hour samples. The neural network is able to model general trends, but fails at grasping hour to hour behaviors. Aggregated results on the entire validation set for the RMSE and MAE criteria can be found in Table 3.4. Additional samples can be found in Appendix A.1.3.




$y_{1:T}$ are independent conditionally on the hidden states $z_{1:T}$ so that

$$p(y_{1:T}|z_{1:T}) = \prod_{t=1}^{T} p(y_t|z_t).$$

We can derive the log-likelihood expression of the complete distribution :

$$\ell(y_{1:T}, z_{1:T}; \theta) = \sum_{t=1}^{T}\sum_{k=1}^{K} \log p(y_t; \mu_k, \sigma_k)\mathbb{1}\{z_t = k\} + \sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{j=1}^{K} \log(a_{kj})\mathbb{1}\{z_t = j, z_{t-1} = k\} + \sum_{k=1}^{K} \log \pi_k \mathbb{1}\{z_0 = k\},$$

where $\theta = \{\mu_k, \sigma_k, a_{kj}, \pi_k\}$ denote the parameters of the model, estimated through the Expectation Maximization (EM, Dempster et al. (1977)) algorithm.

**Evaluation protocol.** Unlike deterministic neural networks, HMMs allow to model the distribution of the observations easily. One common way to approximate this distribution consists in drawing multiple samples from the trained model, and computing the associated relative humidity predictions. From there, we can derive various metrics about the uncertainty of the model.

In order to benchmark the precision of the HMM to the neural network, we average $N = 100$ independent predictions, and compute the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE) criteria over the validation set. Results are displayed in Table 3.4 and are discussed in the next paragraph.

TABLE 3.4 – Comparison of RMSE, MAE and computation time of our model against the benchmarked HMM. This table provide aggregated results on the entire validation set. Mean values of the estimators are displayed along with their variance.

| | RMSE | MAE | Computation time |
|---|---|---|---|
| LSTM | $0.31 \pm 0.18$ | $0.27 \pm 0.16$ | $21ms$ |
| HMM | $0.46 \pm 0.19$ | $0.40 \pm 0.17$ | $99m$ |

### 3.3.5 Discussion

We evaluated two statistical models on a relative humidity forecasting task. The first one, a deterministic neural network, was able to provide good quality predictions based on the input variables, however lacked any information regarding its own uncertainty toward those predictions. When modelling historic data, we rarely have access to all relevant input variables : we cannot simply rely on the model being correct. In contrast, the Hidden Markov Model can be used to quantify the uncertainty, for instance by computing confidence intervals based on identical independent draws from the observation distribution. However, because it can only model linear interactions on the latent space, we report a higher bias for the HMM than the LSTM.

The approximation quality of the neural network can be combined with the distribution modelling of the HMM. In the next chapter, we propose two generative models with low bias on the Relative Humidity dataset, while providing uncertainty estimation through confidence intervals. We compare two of the main approaches to parameter estimation for non linear time series models, Sequential Monte Carlo methods and Variational Inference.

# Bibliographie


Ade, R. and Rehm, M. (2021). Is green certification the solution to substandard indoor air quality (humidity) ? a case study of old, new and green-certified dwellings. *Australasian Journal of Environmental Management*, 28(2) :126–148.

Ajib, B. (2018). Data-driven building thermal modeling using system identification for hybrid systems.

Bai, Y. (2022). Relu-function and derived function review. *SHS Web Conf.*, 144 :02006.

Bengio, Y., Simard, P. Y., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2 :157–66.

Bre, F., Roman, N. D., and Fachinotti, V. D. (2020). An efficient metamodel-based method to carry out multi-objective building performance optimizations. *Energy and Buildings*, 206 :109576.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation : Nsga-ii. In *PPSN*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society : Series B*, 39 :1–38.

Fabrizio, E. and Monetti, V. (2015). Methodologies and advancements in the calibration of building energy models. *Energies*, 8 :2548–2574.

Hansen, N., Müller, S., and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11 :1–18.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9 :1735–1780.

Hou, J., Sun, Y., Dai, X., Liu, J., Shen, X., Tan, H., Yin, H., Huang, K., Gao, Y., Lai, D., Hong, W., Zhai, X., Norbäck, D., and Chen, Q. (2021). Associations of indoor carbon dioxide concentrations, air temperature, and humidity with perceived air quality and sick building syndrome symptoms in chinese homes. *Indoor Air*, 31(4) :1018–1028.

Józefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *ArXiv*, abs/1602.02410.

Kim, Y., Jernite, Y., Sontag, D. A., and Rush, A. M. (2016). Character-aware neural language models. In *AAAI*.

Kingma, D. P. and Ba, J. (2015). Adam : A method for stochastic optimization. *CoRR*, abs/1412.6980.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Li, L., Yang, L., and Zeng, Y. (2021). Improving sentiment classification of restaurant reviews with attention-based bi-gru neural network. *Symmetry*, 13(8).

Madureira, J., Paciência, I., Rufo, J., Severo, M., Ramos, E., Barros, H., and de Oliveira Fernandes, E. (2016). Source apportionment of co2, pm10 and vocs levels and health risk assessment in naturally ventilated primary schools in porto, portugal. *Building and Environment*, 96 :198–205.

Magnier, L. and Haghighat, F. (2010). Multiobjective optimization of building design using trnsys simulations, genetic algorithm, and artificial neural network. *Building and Environment*, 45 :739–746.

Mozer, M. C. (1989). A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems*, 3.

Ohta, Y., Sasakawa, T., and Sato, H. (2020). Evolutionary air-conditioning optimization using an lstm-based surrogate evaluator. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Persily, A. (2015). Indoor carbon dioxide concentrations in ventilation and indoor air quality standards. 36th Air Infiltration and Ventilation Centre Conference, Madrid , -1.

Proust, M. (1921). *Sodome et Gomorrhe*.

Reynolds, J., Rezgui, Y., Kwan, A., and Piriou, S. (2018). A zone-level, building energy optimisation combining an artificial neural network, a genetic algorithm, and model predictive control. *Energy*, 151 :729–739.

Roman, N. D., Bre, F., Fachinotti, V. D., and Lamberts, R. (2020). Application and characterization of metamodels based on artificial neural networks for building performance simulation : a systematic review. *Energy and Buildings*, page 109972.

Sendra-Arranz, R. and Gutiérrez, Á. (2020). A long short-term memory artificial neural network to predict daily hvac consumption in buildings. *Energy and Buildings*, 216 :109952.

Seyedzadeh, S., Rahimian, F., Oliver, S., Rodriguez-Trejo, S., and Glesk, I. (2020). Machine learning modelling for predicting non-domestic buildings energy performance : A model to support deep energy retrofit decision-making. *Applied Energy*, 279 :115908.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Zhang, H., Srinivasan, R. S., and Ganesan, V. (2021). Low cost, multi-pollutant sensing system using raspberry pi for indoor air quality monitoring. *Sustainability*.

Zhang, Z., Robinson, D., and Tepper, J. (2018). Detecting hate speech on twitter using a convolution-gru based deep neural network. In Gangemi, A., Navigli, R., Vidal, M.-E., Hitzler, P., Troncy, R., Hollink, L., Tordai, A., and Alam, M., editors, *The Semantic Web*, pages 745–760, Cham. Springer International Publishing.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer : Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12) :11106–11115.

# Chapitre 4

# Uncertainty modelling through random latent variables

Neural networks architectures are able to leverage huge amounts of data in order to infer millions of parameters, and produce accurate predictions. Yet we cannot simply rely on their predictions being accurate when making decisions impacting well being and health. In the previous chapter, we experimented with Hidden Markov Models (HMM) in an attempt to model the uncertainty associated with each prediction. This approach is able to model the distribution of the observations instead of simply producing a prediction. By analyzing it, we can derive information about the uncertainty of the model. However, unlike neural networks, the training procedure of HMM does not scale well to numerous parameters in deep architectures, leading to degraded accuracy.

The main challenge we address in this chapter consists in combining the accuracy of neural networks with uncertainty modelling. For this, we propose two approaches. In Section 4.4, we propose to decouple representation learning from uncertainty modelling, in a two step training procedure. The unknown parameters are estimated by minimizing a deterministic cost function, then the last layer of the architecture is finetuned using Sequential Monte Carlo (SMC) methods. The results presented are adapted from the following contribution : *Last layer state space model for representation learning and uncertainty quantification*, Cohen, M., Charbit, M. and Le Corff, S., 2023 IEEE Statistical Signal Processing Workshop. In Section 4.5, we develop a model with a discrete latent representation of the data. We show that discrete regimes allow better interpretability of the model. Additionally, parameter estimation does not require the complex approximations that come with continuous latent vectors, and is achieved through Variational Inference. The results presented are adapted from the following contribution : *Variational Discrete Latent Representation for Time Series Modelling*, Cohen, M., Charbit, M. and Le Corff, S., 2023 IEEE Statistical Signal Processing Workshop.

## 4.1 Motivations

The metamodel has proven to be an accurate surrogate of TRNSYS for indoor temperature and consumption modelling and has provided interesting practical performance for several buildings, see Chapter 3. However, as real buildings behaviors are often much more complex than the simplified models written using TRNSYS, our model is still not able to closely match a huge variety of historic building data, during calibration for instance. This can be explained by many factors : a monozone simplified model is not realistic to describe all buildings, the noise incurred from sensors and captors are not taken into account in the metamodel training, and the building faces many random solicitations such as the actions of its occupants. The same factors can explain the limited performance when

modelling relative humidity in the previous chapter.

Although neural networks have been developed to solve many predictive tasks, they often solely output a single-point estimate, so that no uncertainty measure is available to provide statistical guarantees on their predictions. Designing deep models and training associated procedures to estimate predictive distributions of future observations is a long standing challenge. These distributions could naturally be used to compute several uncertainty measures and validate the model or metamodel predictions, or highlight situations in which the predictions cannot be trusted for critical practical applications. On the other hand, understanding better the uncertainty would help in introducing new models or new training procedures to improve the performance in large-scale real-world machine learning applications.

In this chapter, we propose a new approach to provide a flexible statistical framework that can be combined with deep learning architectures, taking into account stochastic latent states and noisy observations. One of the main challenges of this approach resides in the dimension of our statistical models : in order to compute abstract features from the input data, deep neural networks often rely on hundred of thousands of parameters. While there exists various methods for uncertainty estimation, fitting such large networks requires additional steps, that we introduce in the following sections.

## 4.2 Review of the literature

Recurrent Neural Networks (RNN) were first introduced as an efficient and convenient architecture to address short time dependencies problems. They have been consistently improved to develop longer term memory, and optimize their implementations Bengio et al. (1994); Hochreiter and Schmidhuber (1997). Current deep learning frameworks allow stacking arbitrary high number of recurrent layers, whose parameters are estimated by gradient descent through automated differentiation procedures, as shown in Graves et al. (2013). Fostering the dissemination of deep learning-based algorithms to such fields requires to design new approaches for uncertainty quantification.

**Stochastic Recurrent Neural Networks.** Bayesian statistics are able to approximate the distributions of future observations and to provide uncertainty estimation Hinton and Neal (1995). Several architectures inspired by Variational Inference (see Jordan et al. (2004) for an introduction) emerged by considering latent states as random variables and approximating their posterior distribution. The authors of Chung et al. (2015); Fraccaro et al. (2016) built on a traditional recurrent architecture by modelling temporal dependencies between these latent random states. Results presented in Fortunato et al. (2017) yield improved performance when considering local gradient information for computing the posterior.

Sequential Monte Carlo (SMC) methods have also been successfully applied to Recurrent Neural Networks. Instead of computing a single latent vector at each time step, a set of particles representing the distribution of the latent space are propagated, and associated with importance weights. In Maddison et al. (2017); Naesseth et al. (2018), the authors were able to model complex distributions on dependant data. The parameters of the RNN are estimated through Variational Inference.

In Blundell et al. (2015), the authors considered weights as random variables and proposed approximations of their posterior distributions allowing more robust predictions. Such Bayesian neural networks have been proposed and studied in a variety of works, see for instance Hernández-Lobato and Adams (2015); Khan et al. (2018); Teye et al. (2018). However, these methods are computationally intensive for high dimensional models and we do not have statistical guarantees on their ability to capture the target posterior distribution, see Foong et al. (2020a).

**Ensemble methods.** Monte Carlo Dropout (MC Dropout) methods offer to capture uncertainty by leveraging Dropout during both training and evaluation tasks, producing variable predictions from a single trained recurrent model, see Gal and Ghahramani (2016). In the recent years, MC Dropout methods have been applied in many industrial fields, such as flight delay prediction Vandal et al. (2018) or molecular simulations Wen and Tadmor (2020). Alternatively, ensemble methods consist in training distinct networks to obtain a combined prediction, as shown in Pearce et al. (2018); Lakshminarayanan et al. (2017). However, these frequentist approaches fail to guarantee proper calibration of the model, as highlighted by Ashukha et al. (2020), and suffer various limitations, see Foong et al. (2020b).

**Decoupled architectures.** In an effort to provide an alternative strategy with limited computation overhead, Brosse et al. (2020) suggests splitting training in two stages to solve classification problems for independent data : representation learning and uncertainty estimation. The two steps proceed as follows : (i) the algorithm first trains a deep classifier to obtain accurate task-dependent representations of the data, and then (ii) ensemble models are trained using these representations and the output. Their experiments indicate that last-layer algorithms outperform baseline networks and that a single last layer is an appealing trade-off between computational cost and uncertainty quantification. However, this method is restricted to independent and identically distributed data and cannot be directly applied to time series.

## 4.3 Problem definition

Before considering uncertainty quantification for the end-to-end process associated with training a metamodel and solving calibration and optimization tasks, we propose in this chapter a solution to model predictive distributions for generic deep recurrent neural networks. We model the distribution of a series of observations $(y_k)_{k=1}^T$, with the objective of both improving on the results of the previous chapter, presented in Table 3.4, and providing uncertainty quantification. At each time step $k$, we store exogenous variables in a deterministic vector $u_k$. The sample length $T$ is fixed.

Additionally to the Relative Humidity dataset, aggregated by Oze-Energies over two years, we also consider a publicly available dataset in order to propose reproducible results. Both datasets are treated similarly, in terms of model definition, training procedure and evaluation metrics.

**Relative Humidity dataset.** This dataset is composed of hourly records of indoor Relative Humidity (RH), from a building managed by Oze-Energies. Our input data is comprised of the outdoor humidity, temperature and current occupancy. Although occupancy is not directly observed by Oze-Energies, we do have access to $CO_2$ concentration at each time step, which we use as a surrogate variable. We chose to split the dataset in day long samples, where $T = 24$ .

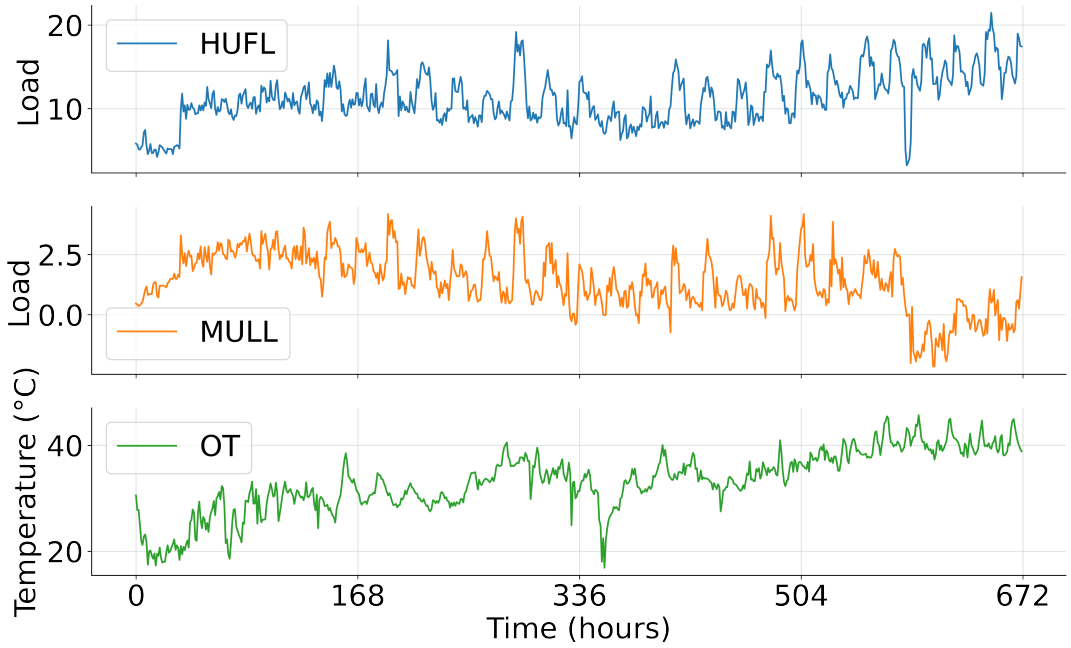
Two years worth of data, namely for 2020 and 2021, are available. Note that the lockdown from the COVID-19 happened in April 2020, and had a strong impact on both $CO_2$ and RH as the building was completely empty during this period. In Figure 3.11, where we plotted the entirety of the dataset for both input variables and target observations, we can see how occupation slowly rose after the lockdown, as highlighted by the growing $CO_2$ levels during occupancy times. For this reason, the first year and a half was used for training, and the remaining of the dataset was saved for validation.

**Electricity Transformer Dataset.** The Electricity Transformer Dataset (ETDataset) records the evolution of oil temperature in electricity transformers in order to better understand and model their state. A good estimation of the

future oil temperature could help reduce energy waste. The dataset was introduced in Zhou et al. (2021), and was released publicly [1]. Oil temperature records, which we aim to model, are associated with six power load features : High Useful Load (HUFL), High Useless Load (HULL), Middle Useful Load (MUFL), Middle Useless Load (MULL), Low Useful Load (LUFL) and Low Useless Load (LULL).

Once again, we have two years worth of hourly data, displayed in Figure 4.1 where we plotted inputs and outputs. Following the original paper, we train our models on samples from the first year, and validate on samples from the following four months.

FIGURE 4.1 – Visualization of a 4-week sample of the Electricity Transformer dataset from Zhou et al. (2021), and presented in Paragraph 4.3. We only plotted the input variables HUFL and MULL for better clarity, as well as the observations (Oil Temperature).




## 4.4 Monte Carlo approach for continuous latent

Inspired by Brosse et al. (2020), we propose a last layer approach to split uncertainty quantification from representation learning, in the context of dependent data. This new method for uncertainty estimation combines high expressivity, quality uncertainty estimations and ease of training. Our proposed architecture is composed of an arbitrary sequential model, followed by a decoupled state space model layer. Using a state space model in the last layer allows to introduce complex predictive distributions for the observations based on task-dependent latent data. However, because the loglikelihood of the observations is not available explicitly in such a setting, the second stage training requires using approximate sampling methods.

Estimating the parameters of potentially high-dimensional models with unobserved (i.e. noisy) layers is a challenging task. We therefore propose to first train an input model following traditional deep learning approaches, then use Sequential Monte Carlo methods in a lower dimensional state space to account for uncertainty, with tractable and computationally efficient simulation-based methods. The two-stage training algorithm is presented in Algorithm 1, and the architecture of the model is described in Figure 4.3

1. https ://github.com/zhouhaoyi/ETDataset/

### 4.4.1 Related works

#### Sequential Monte Carlo Methods for recurrent neural networks

They are many approaches to model uncertainty in statistical models. We present an architecture based on Sequential Monte Carlo methods, an application of Monte Carlo algorithms for time dependent data. They have been successfully combined with recurrent neural networks to tackle variable sequential problems, see for instance Naesseth et al. (2018); Maddison et al. (2017); Ma et al. (2020). We turn to Martin et al. (2020) for an example using more complex neural architectures, such as the Transformer. Particle filters have also proven reliable in other fields, such as presented in Liu et al. (2020) for object tracking. These methods however do not scale to high dimension latent space, where computations quickly become intractable and inefficient.

#### Two-stage training

Our two stage training procedure is inspired by Transfer Learning, a well known deep learning technique which consists in producing a first estimate of the parameters using a computationally cheap related task.

With deeper and deeper neural network architectures, come the need for bigger datasets and more complex training procedures, which themselves come at a high price : obtaining data can be expensive, and may require additional labeling ; heavy computations for extended periods usually incur costly infrastructure. Transfer learning first appeared with the realization that the first layers are often almost identical from one model to the next, regardless of the precise task they were trained for, as shown in Yosinski et al. (2014). Another major highlight from this paper is that initializing parameters with an auxiliary task can improve generalization performance.

Many works have shown the practical use of Transfer Learning in diverse fields of application. A first estimate of the parameters is obtained from publicly available models, trained on generic tasks, or by training a new model on a public dataset. Then, the parameters - or a subset - are finetuned on a smaller scale dataset, see Pan and Yang (2010) for a review. This method is flexible, as it allows changing the architecture of the model, as well as removing or adding new layers between the pre-training and fine-tuning steps, see Chouhan et al. (2020) for an example applied to Computer Vision or Raffel et al. (2022) to Natural Language Processing.

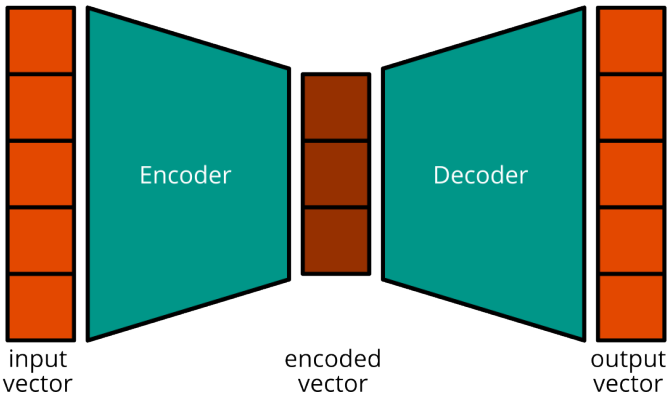
Transfer Learning is an essential part of our proposed architecture. Since estimating all parameters of a deep neural network using Sequential Monte Carlo methods would be computationally prohibitive, we first train the first layers of our model on a simple regression task. In a second stage, the parameters of the last layer can be finetuned on a more complex uncertainty modelling task.

#### Representation learning

Our objective is to take advantage of the recent efforts in deep neural network design, and training, to abstract huge amounts of complex, intricate data. The authors of Hinton and Salakhutdinov (2006) demonstrated that high dimensional data can be represented in a much lower space using deep neural architectures, and proposed an autoencoder model to this end, see Figure 4.2. In Bengio (2007), it is further shown that more complex data may require deeper architectures to obtain low dimensional representations. On a practical experimentation with samples in a 220 dimension space, the authors of Alkhayrat et al. (2020) concluded that deep neural networks outperformed traditional algorithms for dimensionality reduction, such as the Principal Component Analysis. In all these references, the huge number of parameters arising from deep neural network architectures where estimated using gradient descent.

By implementing such models, we aim at representing our potentially high dimensional input data in a smaller latent space. There, our Sequential Monte Carlo algorithm is be able to efficiently model uncertainty.

FIGURE 4.2 – An autoencoder architecture for dimensionality reduction using deep neural networks. The input vectors are mapped to a lower dimension space using the Encoder, and represented as encoded vectors, here in the middle of the diagram. The Decoder learns the inverse transformation, in order to reconstruct the original input vectors. Both the Encoder and Decoder are non linear parametric functions, usually a combination of well known deep learning layers.




### 4.4.2 Proposed architecture

In the following, for any sequence $(a_m, \dots, a_n)$ with $n \geq m$, we use the short-hand notation $a_{m:n} = (a_m, \dots, a_n)$.

#### Input model

We first consider an arbitrary multi-layer neural network $h_\phi$ with unknown parameters $\phi$, responsible for transforming the input time series $u_{1:T}$ into high level features $\widetilde{u}_{1:T}$, in a small dimension latent space :

$$\widetilde{u}_{1:T} = h_\phi(u_{1:T})\,, \qquad \text{input model.}$$

We produce an estimate $\hat{\phi}$ during the first training stage, by introducing an auxiliary function $\kappa_\psi$ to model the observations as follows : for all $1 \leq t \leq T$, $y_t = \kappa_\psi(y_{t-1}, \widetilde{u}_t) + \epsilon_t$ and $y_0 = \kappa_\psi(\widetilde{u}_0) + \epsilon_0$, where $(\epsilon_t)_{t \leq 0}$ are i.i.d. centered Gaussian random variables with unknown variance $\Sigma$. The input model is trained on a simple deterministic regression task, by performing gradient descent on the mean squared error, leading to a first estimate of $\phi$ and $\psi$. We keep the estimated parameters $\hat{\phi}$ while the auxiliary function $\kappa_\psi$ and its parameters, only designed to model the observations, are discarded.

#### State space model

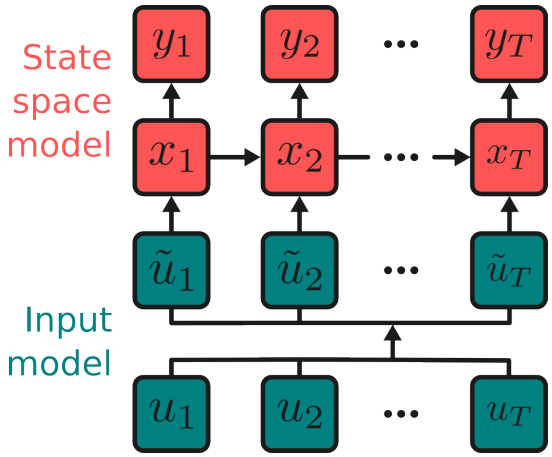
The next step is to define a state space model taking as input the previously extracted features $\widetilde{u}_{1:T}$. Let $x_{1:T}$ a sequence of stochastic hidden states computed recursively and $y_t$ their associated predictions. For all $t \geq 1$, the model is defined as :

$$\begin{aligned} x_t &= g_\theta(x_{t-1}, \widetilde{u}_t) + \eta_t\,, && \text{state model,} \\ y_t &= f_\theta(x_t) + \varepsilon_t\,, && \text{observation model,} \end{aligned}$$

where $\theta$ are the unknown real-valued parameters of the network (weights and biases) and $f_\theta$ and $g_\theta$ are nonlinear parametric functions. We chose $(\eta_t)_{t \geq 1}$ and $(\varepsilon_t)_{t \geq 1}$ as two independent sequences of i.i.d. centered Gaussian random variables with covariance matrices $\Sigma_x$ and $\Sigma_y$, although any distribution can be substituted to better match the noise distribution.

This decoupled approach aims at reducing the number of parameters in $\theta$, compared to $\phi$, in order to estimate them using Sequential Monte Carlo methods. In the next section, we describe this second training procedure for estimating the last layer parameters $\theta$ only, by keeping $\hat{\phi}$ fixed. All implementation and training details are postponed to Section 4.4.5.

FIGURE 4.3 – Our architecture combining a generic input model with a state space model on the last layer.




### 4.4.3 Sequential Monte Carlo Layer

In this section, we detail how to estimate the parameter $\theta$ of the model introduced in Section 4.4.2, from a record of observations $y_{1:T}$. Unlike deterministic neural networks, the likelihood $L$ of the observations is not available explicitly, as it would require integrating over the distribution of the hidden states $x_{1:T}$ :

$$L : \theta \mapsto \int p_\theta(y_{1:T}, x_{1:T}) \mathrm{d}x_{1:T},$$

where $p_\theta$ is the joint probability density function of the observations and the hidden states. Consequently, the score function $\theta \mapsto \nabla_\theta \log L(\theta)$ is intractable, and automated differentiation cannot be used directly. We propose to optimize instead a Monte Carlo estimator of this score function, using Fisher's identity, see for instance Douc et al. (2013) :

$$\nabla_\theta \log p_\theta(y_{1:T}) = \mathbb{E}_\theta \left[ \nabla_\theta \log p_\theta(x_{1:T}, y_{1:T}) | y_{1:T} \right] , \tag{4.1}$$

where $\mathbb{E}_\theta$ designs the expectation under the model parameterized by $\theta$ (the dependency on the input $u_{1:T}$ is kept implicit here for better clarity). The conditional distribution of $x_{1:T}$ given $y_{1:T}$ is not available explicitly for a nonlinear state space model, but it can be approximated using a sequential Monte Carlo smoother. A classical approach consists in first iteratively approximating the filtering distributions $p_\theta(x_t|y_{1:t})$, for $t \leq T$, by a set of $N$ particles $(\xi_t^\ell)_{1 \leq t \leq T, 1 \leq \ell \leq N}$. Then, the smoothing distribution $p_\theta(x_{1:T}|y_{1:T})$ can be approximated by associating importance weights $(\omega_T^\ell)_{\ell=1}^N$ with each particle genealogy $(\xi_{1:T}^\ell)_{\ell=1}^N$. In the following paragraphs, we denote by $\Psi_{\mu,\Sigma}$ the Gaussian probability density function with mean vector $\mu$ and covariance matrix $\Sigma$.

**Particle filter**

We now present a standard particle filter. At $t = 0$, $(\xi_0^\ell)_{\ell=1}^N$ are sampled independently from $\rho_0 = \Psi_{0,\Sigma_x}$, and each particle $\xi_0^\ell$ is associated with the standard importance sampling weight :

$$\omega_0^\ell \propto \Psi_{Y_0,\Sigma_y}(f_\theta(\xi_0^\ell)) \,.$$

Then, for $t \geq 1$, using $\{(\xi_{t-1}^\ell, \omega_{t-1}^\ell)\}_{\ell=1}^N$, pairs $\{(I_t^\ell, \xi_t^\ell)\}_{\ell=1}^N$ of indices and particles are sampled from the instrumental distribution :

$$\pi_t(\ell, x) \propto \omega_{t-1}^\ell p_t(\xi_{t-1}^\ell, x) \,.$$

In this application we use for $p_t(\xi_{t-1}^\ell, \cdot)$ the prior kernel $\Psi_{g_\theta(\xi_{t-1}^\ell, \tilde{u}_t), \Sigma_x}$. For $\ell \in \{1, \ldots, N\}$, $\xi_t^\ell$ is associated with the importance weight $\omega_t^\ell \propto \Psi_{Y_t,\Sigma_y}(f_\theta(\xi_t^\ell))$. An application of this algorithm is presented in Figure 4.4 on a toy model. Such a particle filter with multinomial resampling is referred to as the bootstrap algorithm, see Gordon et al. (1993). It has been extended and analyzed in many directions in the past decades, see Pitt and Shephard (1999); Douc and Cappé (2005); Chopin and Papaspiliopoulos (2020). In other lines of works, the adaptive tuning of the Monte Carlo effort has been analyzed in order to adapt the number of particles on-the-fly, see Elvira et al. (2016, 2021).

**Path-space particle smoother**

Any particle smoother can be used to estimate (4.1) such as the Path-space smoother Kitagawa (1996), the Forward Filtering Backward Smoothing Doucet et al. (2000) or the Forward Filtering Backward Simulation algorithm Godsill et al. (2004). Additionally, because estimating (4.1) amounts to computing a smoothed expectation of an additive functional, we can also use very efficient forward-only SMC smoothers such as the PaRIS algorithm and its pseudo-marginal extensions Olsson and Westerborn (2014); Gloaguen et al. (2022).

In order to illustrate our approach with the simplest (and computationally cheapest) SMC smoother, consider first the Path-space smoother. The genealogical trajectories are defined recursively and updated at each time step with the particles and indices $(\xi_{t+1}^\ell, I_{t+1}^\ell)$ : for all $0 \leq t \leq T-1$,

$$\xi_{0:t+1}^\ell = \left(\xi_{0:t}^{I_{t+1}^\ell}, \xi_{t+1}^\ell\right)$$

In Figure 4.4, the $N = 3$ genealogical trajectories are :

$$\xi_{0:4}^1 = (\xi_0^3, \xi_1^2, \xi_2^2, \xi_3^3, \xi_4^1),\ \xi_{0:4}^2 = (\xi_0^3, \xi_1^1, \xi_2^3, \xi_3^2, \xi_4^2),\ \xi_{0:4}^3 = (\xi_0^3, \xi_1^2, \xi_2^2, \xi_3^3, \xi_4^3).$$

The score function (4.1) can then be estimated as follows :

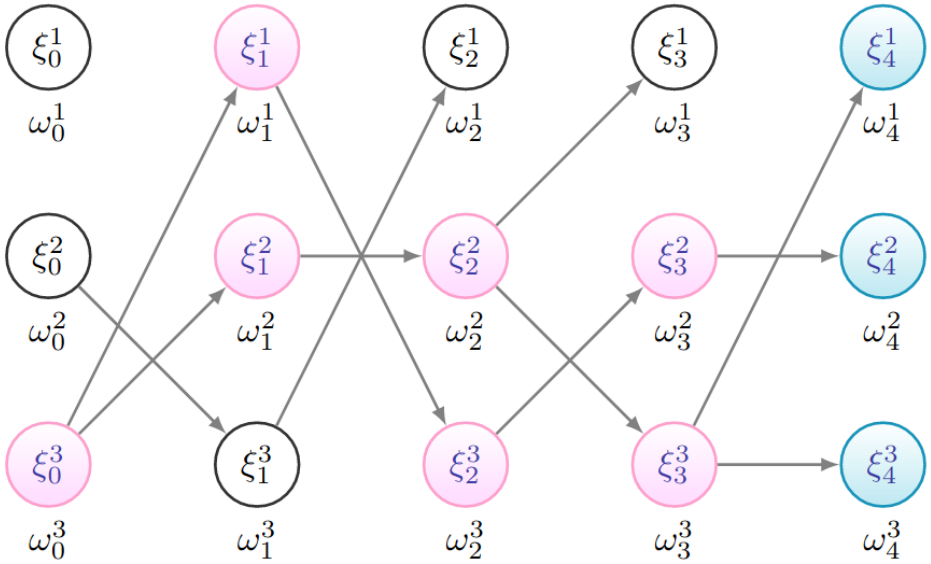
$$\widehat{S}_\theta^N(y_{1:T}) = \sum_{\ell=1}^N \omega_T^\ell \nabla_\theta \log p_\theta(\xi_{1:T}^\ell, y_{1:T}) \,,$$

where $p_\theta$ is the joint probability density function of $(x_{1:T}, y_{1:T})$ for the model described in Section 4.4.2. Using automated differentiation, we can perform gradient descent on the parameter $\theta$.

**Online extensions and the PaRIS**

Although simple and computationally cheap, the smoother based on the ancestral lines of each particle may provide poor estimators. The resampling steps deplete the ancestor lines and, as time increases, two different

FIGURE 4.4 – The auxiliary particle filter for $N = 3$ and $T = 4$. In blue are the particles sampled for the last time step. Each particle from a previous time step belonging to their genealogy is colored in pink. The others, in white, will be discarded.



particles are likely to originate from the same ancestors in the first time steps. The Monte Carlo estimator then suffers generally from high variance when estimating joint-smoothing expectations as in the score estimation problem.

This degeneracy relative to the smoothing problem can be overcome using *backward sampling*. It is specifically designed for additive functionals so it is well suited to our setting (4.1) where

$$\nabla_\theta \log p_\theta(x_{1:T}, y_{1:T}) = \sum_{s=1}^{T} \{\nabla_\theta \log m_\theta(x_{t-1}, \widetilde{u}_t; x_t) + \nabla_\theta \log r_\theta(x_t, y_t)\} \ ,$$

where $m_\theta(x_{t-1}, \widetilde{u}_t; \cdot)$ is the transition density of the state model and $r_\theta(x_t, \cdot)$ is the density of the conditional distribution of $y_t$ given $x_t$ and by convention $m_\theta(x_0, \widetilde{u}_1; \cdot) = \rho_0(\cdot)$.

Appealingly, using the path-space smoother described in the previous section, a Monte Carlo estimator of the score function can be obtained online by setting,

$$\widehat{S}_\theta^N(y_{1:T}) = \sum_{i=1}^{N} \omega_T^i \tau_T^i, \tag{4.2}$$

where the statistics $\{\tau_s^i\}_{i=1}^N$ satisfy the recursion

$$\tau_{s+1}^i = \tau_s^{I_{s+1}^i} + \tilde{h}_s(\xi_s^{I_{s+1}^i}, \xi_{s+1}^i), \tag{4.3}$$

where

$$\tilde{h}_s(x_s, x_{s+1}) = \nabla_\theta \log m_\theta(x_s, \widetilde{u}_{s+1}; x_{s+1}) + \nabla_\theta \log r_\theta(x_{s+1}, y_{s+1}) \, .$$

Note that the conditional probability that $I_{s+1}^i = j$ given the offspring $\xi_{s+1}^i$ and the ancestors $\{\xi_s^\ell\}_{\ell=1}^N$ is given by

$$\mathbf{\Lambda}_s(i, j) = \frac{\omega_s^j m_\theta(\xi_s^j, \xi_{s+1}^i)}{\sum_{\ell=1}^N \omega_s^\ell m_\theta(\xi_s^\ell, \xi_{s+1}^i)}. \tag{4.4}$$

It is straightforward to note that $\mathbf{\Lambda}_s$ is a Markov transition kernel on $\{1, \ldots, N\} \times \{1, \ldots, N\}$. The particle-path

degeneracy of the path-space smoother can be overcome by computing the expectation under the law of this kernel :

$$\tau_{s+1}^i = \sum_{j=1}^N \mathbf{\Lambda}_s(i,j)\{\tau_s^j + \tilde{h}_s(\xi_s^j, \xi_{s+1}^i)\}. \tag{4.5}$$

This approach, first proposed in Del Moral et al. (2010), avoids the path degeneracy as is eliminates the ancestral connection between the particles by means of averaging. Furthermore, it is entirely online. Still, a significant drawback is the overall $\mathcal{O}(N^2)$ complexity. Following Olsson and Westerborn (2014), we propose to sample $M \ll N$ conditionally independent indices $\{J_s^{i,j}\}_{j=1}^M$ from the distribution $\mathbf{\Lambda}_s(i,\cdot)$ and to update the statistics according to

$$\tau_{s+1}^i = M^{-1} \sum_{j=1}^M \left(\tau_s^{J_s^{i,j}} + \tilde{h}_s(\xi_s^{J_s^{i,j}}, \xi_{s+1}^i)\right). \tag{4.6}$$

If the state transition density is uniformly bounded from above and below, an accept-reject approach allows the sampling-based update to be performed for $i \in \{1, \ldots, N\}$ at an $\mathcal{O}(N(M+1))$ overall complexity if a pre-initialized multinomial sampler is used. A key aspect of this approach is that the number $M$ of sampled indices at each step can be very small ; indeed, for any fixed $M \geq 2$, the algorithm, which is referred to as the PaRIS, can be shown to be stochastically stable with an $\mathcal{O}(t)$ variance (see (Olsson and Westerborn, 2014, Section 1) for details), and setting $M$ to $2$ or $3$ yields typically fully satisfying results.

**Stochastic Gradient Descent for online estimation**

An appealing application of the last layer approach is recursive maximum likelihood estimation, i.e., where new observations are used only once to update the estimator of the unknown parameter $\theta$. In Brosse et al. (2020), the authors used in particular Stochastic Gradient Descent (SGD) and Stochastic Gradient Langevin Dynamics to update the estimation of $\theta$ and perform uncertainty quantification. In state space models, recursive maximum likelihood estimation produces a sequence $\{\theta_t\}_{t\geq 0}$ of parameter estimates writing, for each new observation $y_t$, $t \geq 1$,

$$\theta_t = \theta_{t-1} + \gamma_t \nabla_\theta \ell_\theta(y_t | y_{0:t-1}),$$

where $\ell_\theta(y_t|y_{0:t-1})$ is the loglikelihood for the new observation given all the past, and $\{\gamma_t\}_{t\geq 1}$ are positive step sizes such that $\sum_{t\geq 1}\gamma_t = \infty$ and $\sum_{t\geq 1}\gamma_t^2 < \infty$. A practical implementation of such an algorithm, where $\nabla_\theta \ell_\theta(y_t|y_{0:t-1})$ is approximated using the weighted samples $\{(\xi_t^\ell, \omega_t^\ell)\}_{\ell=1}^N$ can be found for instance in Gloaguen et al. (2022). The PaRIS algorithm proposed in Olsson and Westerborn (2014) allows to use the weighted samples $\{(\xi_t^\ell, \omega_t^\ell)\}_{\ell=1}^N$ and the statistics $\{\tau_t^\ell\}_{\ell=1}^N$ on-the-fly to approximate $\nabla_\theta \ell_\theta(y_t|y_{0:t-1})$ either using rejection sampling approaches or importance sampling steps. The update of the recursive maximum likelihood algorithm is based on :

$$\nabla_\theta \ell_\theta(y_t|y_{0:t-1}) = \frac{\pi_{t;\theta}[\nabla_\theta r_{t;\theta}] + \eta_{t;\theta}[r_{t;\theta}]}{\pi_{t;\theta}[r_{t;\theta}]}, \tag{4.7}$$

where $\pi_{t;\theta}$ is the predictive distribution at time $t$, i.e. the law of $x_t$ given $y_{0:t-1}$, $r_{t;\theta} = r_\theta(\cdot, y_t)$ and

$$\eta_{t;\theta}[r_{t;\theta}] = \phi_{0:t;\theta|t-1}[h_{0:t;\theta} r_{t;\theta}] - \pi_{t;\theta}[r_{t;\theta}]\phi_{0:t;\theta|k-1}[h_{0:t;\theta}],$$

with $\phi_{0:t;\theta|t-1}$ the distribution of $x_{0:t}$ given $y_{0:t-1}$ and

$$h_{0:t;\theta}(x_{0:t}) = \sum_{s=0}^{t-1} \nabla_\theta \log q_{s,\theta}(x_s, x_{s+1}), \tag{4.8}$$

with $q_{s,\theta}(x_s, x_{s+1}) = r_{s,\theta}(x_{s+1}) m_\theta(x_s, \widetilde{u}_{s+1}; x_{s+1})$. The signed measure $\eta_{t;\theta}$ is known as the *tangent filter*. Recursive maximum likelihood algorithms rely on the following straightforward decomposition of the normalized loglikelihood :

$$\frac{1}{n}\nabla_\theta \ell_\theta(y_{0:t-1}) = \frac{1}{t}\sum_{k=0}^{t-1} \nabla_\theta \ell_\theta\left(y_k \mid y_{0:k-1}\right),$$

with the convention $\ell_\theta(y_0 \mid y_{0:-1}) = \ell_\theta(y_0)$. Under strong mixing assumptions, for all $\theta$, $\{(X_n, Y_n, \pi_n, \eta_n)\}_{n\geqslant 0}$ is an ergodic Markov chain and the normalized score $\nabla_\theta \ell_\theta(y_{0:t-1})/t$ converges almost surely to a limiting quantity $\lambda(\theta, \theta_\star)$ such that, under identifiability constraints, $\lambda(\theta_\star, \theta_\star) = 0$. A gradient ascent algorithm cannot be designed as the limiting function $\theta \mapsto \lambda(\theta, \theta_\star)$ is not available explicitly. However, *Robbins-Monro algorithm* can be used to solve approximately the equation $\lambda(\theta_\star, \theta_\star) = 0$ with iterative updates

$$\theta_t = \theta_{t-1} + \gamma_t \zeta_t, \quad t \geq 0, \tag{4.9}$$

where $\zeta_t$ is a noisy observation of $\lambda(\theta_{t-1}, \theta_\star)$, equal to (4.7).

The objective is therefore to approximate the key quantity (4.7). Using the particle filter, we can compute the Monte Carlo estimators :

$$\pi_t^N[r_{t;\theta}] = \frac{1}{N}\sum_{\ell=1}^{N} r_{t;\theta}(\xi_t^\ell) \quad \text{and} \quad \pi_t^N[\nabla_\theta r_{t;\theta}] = \frac{1}{N}\sum_{\ell=1}^{N} \nabla_\theta r_{t;\theta}(\xi_t^\ell).$$

In addition, the tangent filter can be approximated as follows :

$$\eta_{t;\theta}^N[r_{t;\theta}] = \frac{1}{N}\sum_{\ell=1}^{N} \tau_t^\ell r_{t;\theta}(\xi_k^\ell) - \left(\frac{1}{N}\sum_{\ell=1}^{N} \tau_t^\ell\right)\left(\frac{1}{N}\sum_{\ell=1}^{N} r_{t;\theta}(\xi_t^\ell)\right). \tag{4.10}$$

Plugging these estimates in equation (4.7) allows to perform the online recursive algorithm.

Although this algorithm is very efficient to update parameters recursively, it is computationally intensive and therefore **fits particularly well our last layer approach** as it would be **intractable for very high dimensional latent states**.

### Discussion

The proposed approach allows to adapt widespread Monte Carlo techniques to deep learning architectures, yet several questions arise from this methodology, due to the nature of particle smoothers and their implementation.

The Path-space smoother is known to quickly degenerate due to the successive resampling steps of the auxiliary filter. During this step, if one of the particle is not selected to be propagated, i.e. for each time step $t$, if there exists $i \in \{1, \cdots N\}$ such that for all $1 \leq j \leq N, I_t^j \neq i$, then the particle's genealogy is discarded for the rest of the process. Over many time steps, the smoother degenerates as most particles share the same genealogy. In Figure 4.4, the trajectories $\xi_{0:4}^1$ and $\xi_{0:4}^3$ share the same history. Although this limitation can be mitigated by using recent alternatives such as the PaRIS, this raises the question of trade-off between the additional complexity (coming from the $M$ backward samples) and the accuracy of the score estimate.

In addition even the Path-space smoother comes at a heavy computation cost with simulating $N$ trajectories at once. Even for modern processing units, such as the GPU able to parallelize hundreds of matrix multiplications, computation times are bound to increase. It is also noteworthy that the smoothing algorithm, although relatively lightweight compared to running inferences on the neural network or computing gradient descent, cannot be parallelized. Lastly, the number of particles $N$ is an additional hyper parameter to tune during the optimization of the

model.

**Algorithm 1:** Two-stage learning

**Input:** $y_{1:T}, u_{1:T}$
**Output:** $(\hat{\phi}, \hat{\theta})$
$\hat{\phi} \leftarrow$ Train the input model $h_\phi$;
$\widetilde{u}_{1:T} \leftarrow h_{\hat{\phi}}(u_{1:T})$;
Initialize parameter estimate $\widehat{\theta}_0$;
**for** $p \leftarrow 1$ **to** $\mathrm{MaxIt}$ **do**
  $\xi_0^\ell \sim \rho_0$ and $\omega_0^\ell \propto \Psi_{y_0,\Sigma_y}(f_{\widehat{\theta}_{p-1}}(\xi_0^\ell))$;
  **for** $t \leftarrow 1$ **to** $T$ **do**
    **for** $j \leftarrow 1$ **to** $N$ **do**
      $I_t^j \sim \mathbb{P}(I_t^j = m) = \omega_{t-1}^m$;
      $\xi_t^j \sim p_t(\xi_{t-1}^{I_t^j}, \widetilde{u}_t, \cdot)$;
      $\omega_t^j \propto \Psi_{y_t,\Sigma_y}(f_{\widehat{\theta}_{p-1}}(\xi_t^j))$;
      Set $\xi_{0:t}^j = (\xi_{0:t-1}^{I_t^j}, \xi_t^j)$.
  Update the parameter estimate using gradient descent with estimated gradient $\widehat{S}^N_{\widehat{\theta}_{p-1}}(y_{1:T})$.

### 4.4.4 Benchmarked models

#### Hidden Markov Model

We benchmark our model against the Hidden Markov Model defined in Section 3.3.4. Parameters are estimated through the Expectation Maximization algorithm. Compared to our proposed architecture based on a deep neural network, the HMM performs poorly, as the data is strongly non linear, and present complex long term dependencies.

#### Monte Carlo Dropout

Dropout is a common method for regularising neural networks, that consists in randomly - and temporally - discarding connections between neurons during training, see Srivastava et al. (2014). It has been showed to prevent units from co-adapting. During inference, while dropout is usually disabled, it is possible to utilize it to measure the uncertainty associated with a prediction, see Brosse et al. (2020) for a benchmark. Multiple samples are generated independently from the network, and interpreted as a distribution.

While this idea seems promising for independent data, one does not simply apply dropout to Recurrent Neural Networks. In Pham et al. (2014), the authors were not able to successfully utilise dropout in the recurrent connections of their network, as it lead to instability for long sequences. Faced with the same limitation, Moon et al. (2015) proposed to apply a fixed dropout mask to the entire sequence of latents, in order to regularise the model. By applying a mask on the weights, rather than the latent vectors, Gal and Ghahramani (2016) proposed a theoretically grounded application of dropout for recurrent networks, that extends to LSTM and GRU layers. We benchmarked their methodology, named the Monte Carlo Dropout (MC Dropout). Compared to our proposed approach, the training procedure is straightforward, however the model still suffers from overconfidence.

### 4.4.5 Experiments

In this section, we detail the implementation choices of the model described above as well as the evaluation procedure. Results on both the Relative Humidity and the Electricity Transfer datasets, presented in Section 4.3, are reported.

#### Models

The Input model is a $L = 3$ layered GRU model, as defined in the deep learning framework PyTorch[2] : for all $1 \leq \ell \leq L$ and all $1 \leq t \leq T$,

$$
\begin{aligned}
r_t^\ell &= \sigma(W_{ir} u_t^{\ell-1} + b_{ir} + W_{hr} u_{t-1}^\ell + b_{hr}) \,, \\
z_t^\ell &= \sigma(W_{iz} u_t^{\ell-1} + b_{iz} + W_{hz} u_{t-1}^\ell + b_{hz}) \,, \\
n_t^\ell &= \tanh(W_{in} u_t^{\ell-1} + b_{in} + r_t^\ell (W_{hn} u_{t-1}^\ell + b_{hn})) \,, \\
\tilde{u}_t^\ell &= (1 - z_t^\ell) n_t^\ell + z_t^\ell u_{t-1}^\ell \,,
\end{aligned}
$$

where $\phi = \{(W_{is}, b_{is}, W_{hs}, b_{hs}), s \in \{r, z, n\}\}$ are unknown parameters, and $\sigma : x \mapsto 1/(1 + \mathrm{e}^{-x})$ is the sigmoid function. The first layer of the network is assimilated to the input vectors, $\widetilde{u}_t^0 \equiv u_t$ and $\widetilde{u}_0^\ell \equiv 0$. The input dimension $d_{\text{in}}$ corresponds to the number of exogenous input variables of the dataset : $d_{\text{in}} = 3$ for the Relative Humidity dataset and $d_{\text{in}} = 6$ for the Electricity Transformer dataset. The output dimension is set to $6$ for both. In order to estimate the parameters $\phi$, we introduce an auxiliary GRU layer responsible for modelling observations. During the training, we minimize the cost function $\mathcal{L}_{\text{input}}(\phi) = \sum_{i=1}^{N_{\texttt{sample}}} \|\texttt{model}_\phi(u_{1:T}^i) - y_{1:T}^i\|^2$ where for each sample $1 \leq i \leq N_{\texttt{sample}}$ in the dataset, $\texttt{model}_\phi(u_{1:T}^i)$ is the prediction associated with $y_{1:T}^i$ obtained with this deterministic model.

The State Space model is implemented using PyTorch implementations of RNN and Linear layers, in order to use auto differentiation. We chose the following form for $f_\theta$ and $g_\theta$ :

$$
\begin{aligned}
g_\theta &: x_{t-1}, \widetilde{u}_t \mapsto \tanh(W_{gx} x_{t-1} + b_{gx} + W_{gu} \widetilde{u}_t + b_{gu}) \,, \\
f_\theta &: x_t \mapsto \sigma(W_f x_t + b_f) \,,
\end{aligned}
$$

where $\theta = \{W_{gx}, b_{gx}, W_{gu}, b_{gu}, W_f, b_f\}$ are unknown parameters.

In these settings, the formula for updating both $\Sigma_x$ and $\Sigma_y$ is available explicitly, by directly deriving 4.1. Instead of relying on gradient descent, we simply update both matrix at each time step by computing the following :

$$
\begin{aligned}
\Sigma_x &= T^{-1} \sum_{\ell=1}^{N} \sum_{t=1}^{T} \omega_T^\ell (\tanh^{-1}(\xi_t^\ell) - g_{\hat{\theta}}(\xi_{t-1}^\ell, \widetilde{u}_t))^2 \,, \\
\Sigma_y &= T^{-1} \sum_{\ell=1}^{N} \sum_{t=1}^{T} \omega_T^\ell (y_t - f_{\hat{\theta}}(\xi_t^\ell))^2 \,.
\end{aligned}
$$

All following experiments were conducted with $N = 100$ particles, using the Adam optimizer introduced in Kingma and Ba (2015). The learning rate and batch size were chosen using a Tree-structured Parzen Estimator algorithm, see Bergstra et al. (2011). We train models for a maximum of 50 epochs, and employ early stopping to prevent overfit.

2. https://pytorch.org/docs/stable/generated/torch.nn.GRU.html

### Evaluations

In this section, we illustrate the ability of our model to capture the distribution of future observations, by evaluating the benchmarked models using the following protocol. We draw 48 hours long samples $(u_{1:48}, y_{1:48})$ from the validation dataset, composed of a 24 hour long lookback window $(u_{1:24}, y_{1:24})$, containing historic commands and observations, and a predictions window where only future commands are available $(u_{25:48})$. Each model produces a 24 hour long forecast, which is compared to the ground truth to compute the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) criteria reported in Table 4.2 for the Relative Humidity dataset and Table 4.1 for the Electricity Transformer dataset.

Predictions can be performed by approximating the predictive density $p_{\theta,\phi}(y_{t+1}|u_{1:t+1}, y_{1:t})$ by :

$$p^N_{\widehat{\theta},\widehat{\phi}}(y_{t+1}) = \sum_{i=1}^{N} \omega^i_t p_{\widehat{\theta},\widehat{\phi}}(y_{t+1}|\xi^i_t, u_{t+1}) \,,$$

where $p_{\widehat{\theta},\widehat{\phi}}(y_{t+1}|\xi^i_t, u_{t+1})$ is the predictive distribution of $y_{t+1}$ described in Section 4.4.2. Although predictions given the previous time step provide good performance, it is limited as many applications require multi-steps forecasts. In order to explore longer ranges, we can simply run our model to get $N$ samples for any time horizon $p$. Since re sampling of particles is no longer available at that point, the uncertainty grows for our model, as shown in Figure 4.5. We report RMSE and MAE criteria by averaging the forecasts of these $N$ draws. The associated intervals containing 95% of the samples are displayed in Figure 4.7 and Figure 4.6, for $1 \leq p \leq 24$.

We compared our model with MC Dropout methods, by implementing recurrent dropout layers as described in Gal and Ghahramani (2016), with dropout values of $p_{\text{drop}} = 0.05$ and $p_{\text{drop}} = 0.01$. The training procedure is similar to traditional recurrent models ; during inference, we draw $100$ samples from the dropout layers, and compute the same average forecasts and intervals as for our model. Despite being based on the same deep learning architecture, the MC dropout model is still largely overconfident, while our proposed model provide more credible empirical confidence intervals. It also outperforms the Hidden Markov Model introduced in Section 3.3.4, see Table 4.2.

FIGURE 4.5 – Prediction of Relative Humidity given observations in the lookback window ($t < 24$) and without ($t > 24$). Since re sampling of particles is no longer available after $t = 24$, the uncertainty grows as the confidence intervals get larger over time. In the absence of specific initialization rules, the first time step is highly uncertain, without posing a problem in the later steps.

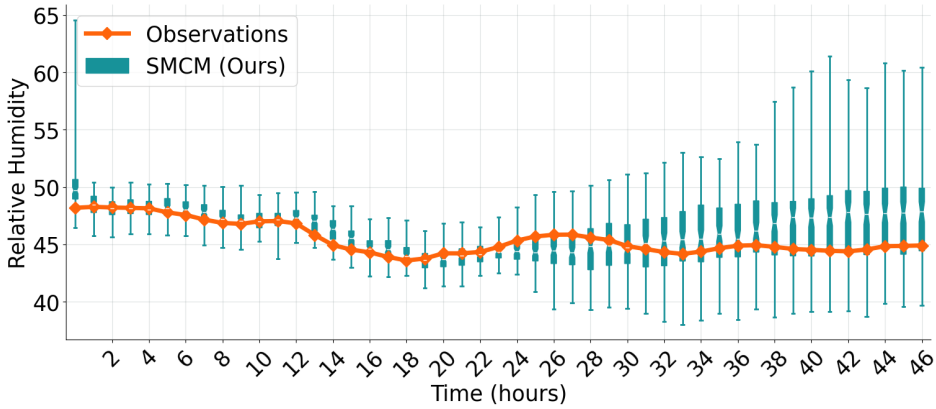


FIGURE 4.6 – Prediction of Oil temperature (ETDataset) given observations in the lookback window ($t < 24$). For better clarity, we only plotted the forecast window ($t > 24$). As a comparison, we plotted the confidence intervals produced by the MC Dropout model (for $p_{\text{drop}} = 0.01$), whose performance are on par with our model but is still largely overconfident. Aggregated results on the entire validation set for the RMSE and MAE criteria can be found in Table 4.1.

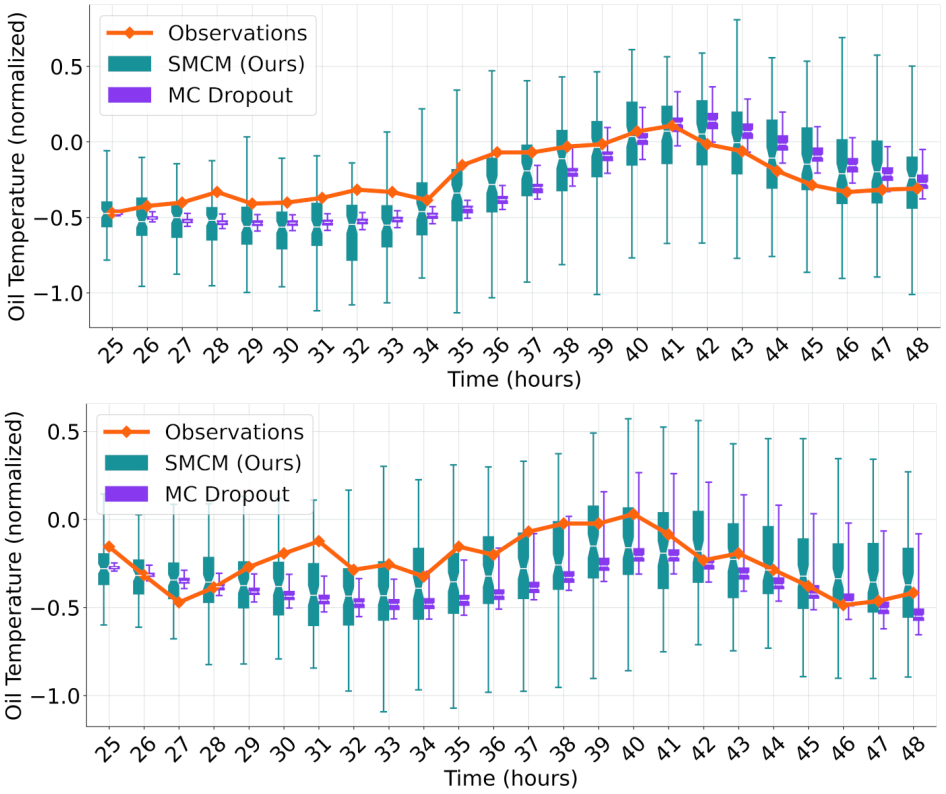



TABLE 4.1 – **Electricity Transformer dataset.** Comparison of RMSE, MAE and computation time of our model against the benchmarked MC Dropout methods and HMM. This table provides aggregated results of the predictions on the entire validation set. Two versions of the dropout model were evaluated, with dropout values $p_{\text{drop}} = 0.05$ and $p_{\text{drop}} = 0.01$. Our model performs similarly to state of the art benchmarks in terms of metrics, while mitigating the overconfidence of neural networks such as the MC Dropout methods. Mean values of the estimators, taken over the validation samples of the dataset, are displayed along with their variance.

| | RMSE | MAE | Computation time |
|---|---|---|---|
| SMCL (ours) | $0.24 \pm 0.13$ | $0.21 \pm 0.12$ | $21ms$ |
| MCD $p = 0.01$ | $0.25 \pm 0.15$ | $0.21 \pm 0.14$ | $19ms$ |
| MCD $p = 0.05$ | $0.28 \pm 0.15$ | $0.24 \pm 0.13$ | $19ms$ |

FIGURE 4.7 – Prediction of Relative Humidity given observations in the lookback window ($t < 24$). For better clarity, we only plotted the forecast window ($t > 24$). As a comparison, we plotted the confidence intervals produced by the MC Dropout model (for $p_{\text{drop}} = 0.01$), whose performance are on par with our model but is still largely overconfident. Aggregated results on the entire validation set for the RMSE and MAE criteria can be found in Table 4.2.

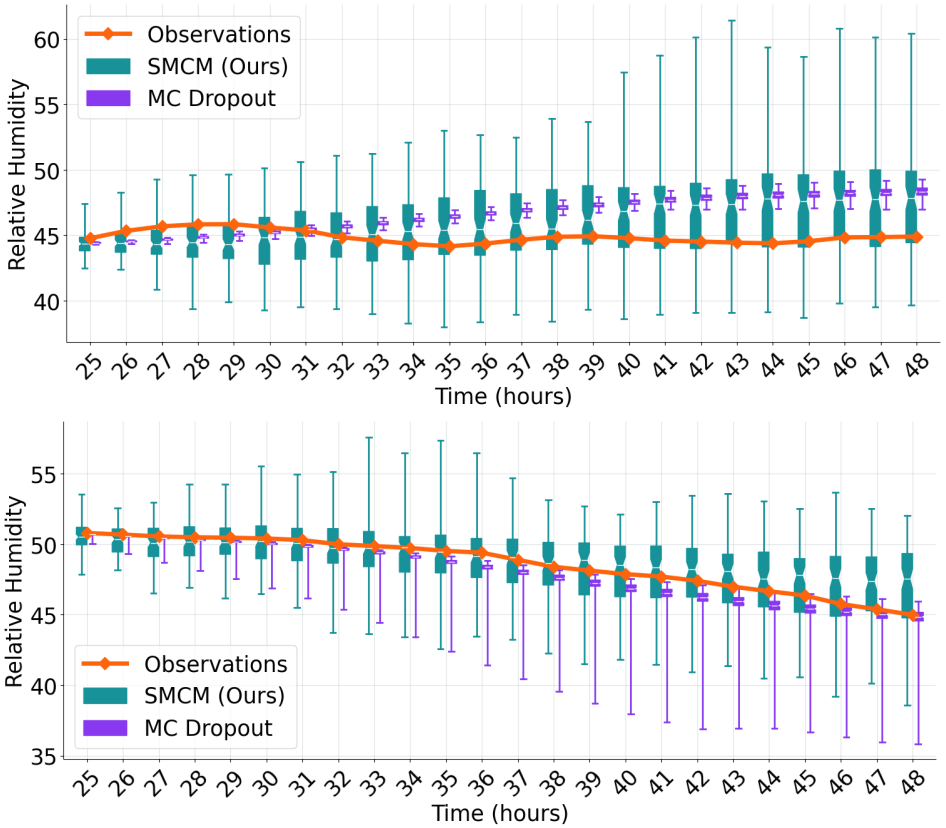



TABLE 4.2 – **Relative Humidity dataset.** Comparison of RMSE, MAE and computation time of our model against the benchmarked MC Dropout methods and HMM. This table provides aggregated results of the predictions on the entire validation set. Two versions of the dropout model were evaluated, with dropout values $p_{\text{drop}} = 0.05$ and $p_{\text{drop}} = 0.01$. Our model performs similarly to state of the art benchmarks in terms of metrics, while mitigating the overconfidence of neural networks such as the MC Dropout methods. Mean values of the estimators, taken over the validation samples of the dataset, are displayed along with their variance.

| | RMSE | MAE | Computation time |
|---|---|---|---|
| SMCL (ours) | $0.30 \pm 0.19$ | $0.26 \pm 0.16$ | $21ms$ |
| MCD $p = 0.01$ | $0.29 \pm 0.18$ | $0.25 \pm 0.16$ | $19ms$ |
| MCD $p = 0.05$ | $0.35 \pm 0.22$ | $0.30 \pm 0.20$ | $19ms$ |
| HMM | $0.46 \pm 0.19$ | $0.40 \pm 0.17$ | $99ms$ |

### 4.4.6 Discussion

We introduced a decoupled architecture for uncertainty estimation applied to dependant data, and evaluated our methodology on two time series datasets. Our deep neural network backbone is able to extract high level features, while particle filtering allows modelling recurrent non linear uncertainty. Our proposed model improves confidence interval quality, compared to MC Dropout methods, without degrading RMSE or MAE criteria.

We demonstrated the potential behind implementing latent space models as a modified RNN cell ; more complex architectures, such as the GRU network used in the input model, or LSTM cells, have the potential to model more complex and long term uncertainty. Likewise, the Path-space smoother could be replaced by more recent alternatives, such as the Forward Filtering Backward Smoothing Doucet et al. (2000) or the Forward Filtering Backward Simulation algorithm Godsill et al. (2004), which mitigate particle degeneracy at the cost of heavier computations. These potential improvements are left for future works.

Our decoupled architecture also enables incorporating uncertainty estimation to an already trained network, by simply adding our last layer. Estimating its parameters would be cheaper than training an entire new uncertainty estimation model, as only the second stage training would be required. In a context where our computational budget is constraint, online smoother such as the PaRIS algorithm Olsson and Westerborn (2014); Gloaguen et al. (2022), could also offer a very efficient method for recalibrating our parameters over large period of time, such as in the case of Oze-Energies.

**Applications for Oze-Energies.** The data gathered by Oze-Energies is quite complex. Even considering Relative Humidity only, our architecture still struggles to precisely model the daily evolution of the building, as seen in the previous section. This is made even more complex as data is scarce, and building's behavior greatly vary from season to season.

A promising methodology to address this complex modelling problem consists in training a global model on the entire available dataset, then estimating fine-tuned models for each month, or season. This way, while the global behavior over the year can be assimilated in a first stage training, specific aspects could be fine-tuned on restricted fractions of the dataset. Our decoupled architecture shines is this context, as different uncertainty levels could easily be derived for different parts of the year.

### 4.4.7 Limitations of a continuous latent space for time series

The main limitation of SMC in our use case is the difficulty to infer parameters. Even in the case of the simple Path-space smoother, computation costs are much heavier than traditional neural network architectures such as MC Dropout, and better alternatives to this smoother would likely only increase this gap. Additionally, each modification of the model requires rewriting the loss function.

For this reason, we decide to experiment on discrete latent, which are much simpler to model for time series as they do not require sampling and re sampling at each time step. The models we present in the following section offer lighter training procedures, and more appealing computational costs, without loosing in precision.

## 4.5 Variational approaches for discrete latent states

### 4.5.1 Introduction

Discrete latent space models aim at representing data through a finite set of features. Recent advances in generative models have pushed towards these representations, as they fit data with naturally discrete hidden states.

For instance, defining a classification task over a dataset often implies a partition of an unobserved latent space, which could be modelled using a categorical random variable, as presented in Kingma et al. (2014). In attention-based models, representing the focus location as a discrete variable, i.e. the right place to focus in the past for predicting the next observation, has proven efficient, and can help interpreting prediction errors, see for instance Xu et al. (2015). When analyzing time series data, the evolution of a discrete latent variable can be interpreted as a switch in regime, see Dzunic and Fisher III (2014). In Ajib et al. (2020), the authors model the indoor temperature of a building by identifying discrete regimes, such as the opening of a window, the presence of occupants or shade.

As one of the most widespread expression of discrete latent space models, Hidden Markov Models have been successfully applied in various fields (Wilks (1998); Gales and Young (2008); Patterson et al. (2017)). Despite modeling hidden states as a discrete Markov chain, they are able to handle complex data structure, see Cappé et al. (2005); Douc et al. (2014) and the references therein for a complete overview. However, dealing with latent data leads to models that are computationally expensive to train, for instance using Expectation Maximization based approaches, and still struggle to handle large scale datasets, in particular when the models contain high-dimensional additional latent states.

In contrast, deep learning methods are able to infer millions of parameters from huge amounts of data - at the cost of much more complex models - through automatic differentiation and gradient computation to optimize a well chosen loss function. However, discrete variables usually prevent us from using gradient propagation, and in turn straightforward trainings. For instance, the Vector-Quantized Variational AutoEncoder (VQ-VAE, van den Oord et al. (2017)) popularized discrete latent spaces for variational inference, by introducing a new family of generative models using posterior distribution over discrete latent variables. This approach requires various approximations in order to propagate the gradient through the model. The reparametrization trick, only recently introduced for categorical variables with the Gumbel Softmax approximation in Jang et al. (2017a), offers an appealing solution to overcome this problem.

We propose the following contributions :

— We introduce a generative model for time series, where the latent space is modelled as a discrete Markov chain, and, conditionally on the latent states, the observations follow a simple autoregressive process. Parameters are jointly estimated by Variational Inference.

— We compared the impact of different prior models to extract high-level features from the input data (based on convolutional and recurrent architectures) on the quality of samples.

— Our model outperforms the state of the art VQ-VAE both in accuracy and computation time.

### 4.5.2 Background

#### Discrete latent representation

One of the most straightforward application of discrete latent representation is derived from semi or unsupervised classification problems, where the data presents distinct semantic classes. The authors of Kingma et al. (2014) propose to model such data as generated by both a continuous and a categorical class variable. By integrating a classification mechanism directly in the model, they are able to outperform continuous latent models, while allowing conditional generation. This idea was transposed to Generative Adversarial Networks by the authors of Chen et al. (2016), by adding a discrete random variable as the input of the generator. Trainings on the MNIST dataset, introduced in LeCun et al. (1998), show how the model associates each value with a class of digits, even without supervision.

Discrete latent space models have proven relevant even when there is no explicit classification task. The authors of Dzunic and Fisher III (2014) modelled climate data by inferring a state space switching interaction model, where the transitions between a finite set of regimes are treated as interactions. Their approach allows for exploratory

pattern discovery and post-analysis by human experts. In Nguyen et al. (2017), a unifying framework for Rao-Blackwellized particle smoother is introduced, where commodity market prices are modeled by a latent Markov chain. Unlike the continuous state space model we introduced previously (see Section 4.4.2), the particle smoother is able to exploit the discrete nature of the latent space. Examples arise in attention-based models too, which have dominated the Natural Language Processing and Computer Vision fields for the last years. Xu et al. (2015) presents an image captioning model where attention scores parametrize a categorical random variable ; by either sampling or computing its expectation, the model extracts visual features used to generate words.

Discrete representations have also successfully replaced continuous latent spaces in existing models. In this way, the authors of Sun et al. (2020) quantized the latent features of an existing variational recurrent neural network applied to text-to-speech synthesis, and reported improved quality in generated audio samples.

#### Introduction to Variational Inference

Variational Inference (VI, see Jordan et al. (2004) for an introduction) is another widely used method for estimating complex statistical models. Their posterior distribution, usually intractable, is approximated by a family of surrogate functions whose parameters are tuned along side the model parameters. The family of functions can be chosen to be very computationally efficient, making Variational Inference an interesting alternative to Sequential Monte Carlo methods for instance. The main challenge of VI resides in a trade-off between ease of computation and quality of the approximation.

**Review of the literature.** Variational Auto-Encoder (VAE), introduced in Kingma and Welling (2014), popularized Variational Inference to statistical model inferring parameters from large datasets. This paper notably derived an Evidence Lower Bound for the likelihood, that can be optimized through gradient descent, taking advantage once again of the automated differentiation offered by deep learning. Its second major contribution consists in re-parametrizing samples from the posterior in order to reduce the variance of the computed gradients, see Kingma and Welling (2019) for a complete overview of the model. Several models have since improved on the original VAE, such as the Importance Weighted Auto-Encoder introduced in Burda et al. (2016), where the authors are able to derive a tighter bound using importance sampling, or the beta-VAE Higgins et al. (2017). In this paper, a new hyper parameter $\beta$ allows to adjust latent channel capacity and independence constraints with reconstruction accuracy.

VI methods have also been successfully applied to discrete latent models. In Salakhutdinov and Hinton (2009), the authors present a methodology for training deep Boltzmann machines through variational inference using Markov chains. Mnih and Rezende (2016) extends the Importance Weighted Auto-Encoder model to discrete latent variables. Petetin et al. (2021) proposes a regime switching Bayesian model, where the Kullback-Leibler Divergence between the posterior and the discrete prior distribution can be computed exactly.

**Definition.** In the following, it is assumed that the distribution of $y$ depends on a latent random variable $z$ with prior distribution denoted $\mathsf{z} \mapsto p_\theta(\mathsf{z})$. Therefore, the likelihood of the observation can be written, for all $y$, $p_\theta(y) = \int p_\theta(\mathsf{z}) p_\theta(y|\mathsf{z}) dz$. Because the posterior distribution $p_\theta(z|y)$ is not tractable, we approximate it using a parametric family of tractable distributions $\{q_\phi, \phi \in \Phi\}$. A lower bound for the loglikelihood, referred to as Evidence Lower BOund (ELBO) $\mathcal{L}$, can be expressed as an expectation under this new distribution $q_\phi$ :

$$\begin{aligned}\log p_\theta(y) &= \log \mathbb{E}_{q_\phi}\left[\frac{p_\theta(y, Z)}{q_\phi(Z|y)}\right] \\ &\leq \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(y|Z) p_\theta(Z)}{q_\phi(Z|y)}\right] = \mathcal{L}_{\theta,\phi}(y)\,.\end{aligned}$$

All is left is to optimize $\mathcal{L}$ with respect to $\theta$ and $\phi$. We can compute a Monte Carlo estimator of this ELBO by sampling $M > 0$ latent variables $(z^{(i)})_{i=1}^{M}$ under the posterior distribution $q_\phi(z|y)$ :

$$\hat{\mathcal{L}}_{\theta,\phi}(y) = M^{-1} \sum_{i=1}^{M} \log \frac{p_\theta(y|z^{(i)}) p_\theta(z^{(i)})}{q_\phi(z^{(i)}|y)} \,.$$

In practice, many numerical applications settle for a single sample from the posterior, $M = 1$. Then, we reparametrize samples $z$ from the posterior in order to propagate the gradient through the model. We want to express $z$ as a differentiable function $g$ of the parameter $\phi$ and some independent random variable $\epsilon$, such that $z = g_\phi(y, \epsilon)$. The expectation under the posterior can now be written under the distribution $p_\epsilon$ of $\epsilon$, and the gradient operator becomes commutative. For any function $f$, we have :

$$\begin{aligned} \nabla_\phi \mathbb{E}_{q_\phi}[f(Z)] &= \nabla_\phi \mathbb{E}_{p_\epsilon}[f(Z)] \\ &= \mathbb{E}_{p_\epsilon} \nabla_\phi [f(Z)] \,. \end{aligned}$$

Note that this reparametrization constraints the choice of parametric posterior distributions.

**Comparison with Sequential Monte Carlo methods for time series.** Sequential Monte Carlo and Variational Inference offer two different path towards approximating a complex posterior distribution. The main advantage of VI methods reside in its computationally appealing surrogate family, and the use of the reparametrization trick allowing to reduce the variance when computation the gradient. The trade-off for this low variance is a higher bias, as the true posterior may not be expressed within the chosen parametric family.

### Application to discrete latent models

The training methodology presented in the previous chapter only applies when the reparametrized posterior function is differentiable. As sampling discrete random variables prevent gradient propagation through the model, a workaround is required for discrete latent models to take advantage of automatic differentiation offered by modern deep learning frameworks.

Recently, a lot of new approaches to discrete latent models have been proposed, influenced by the Vector Quantized Variational Auto Encoders (VQ-VAE) introduced in van den Oord et al. (2017). In this paper, the discrete posterior distribution is defined by associating, with each observation, one of a finite set of codebooks, whose positions are learned during training in order to partition the latent space. The main contribution of this paper is two-fold : first, it is shown that a rough approximation of the gradient of the posterior distribution allows to estimate all parameters using gradient descent ; second, the training procedure is decomposed into steps. In order to easily learn the auto encoder parameters, an uninformative prior is first considered. Samples generated after this step cannot catch the complex dependencies that lie in the latent space. To produce coherent samples, a complex prior model is trained on the latent space, while keeping the previously learnt parameters frozen. Because these prior models are usually already challenging to fit on their own, most applications of the VQ-VAE have not been able to jointly train both parts of the network, see Sun et al. (2020).

We now review alternative methods for defining a discrete posterior distribution, as well as an appropriate prior model.

**Differentiation of the posterior.** Sampling discrete variables from the posterior prevents gradient propagation through the model ; while the authors of the VQ-VAE proposed a straight-through approximation, other approaches have addressed this issue. In Lorberbom et al. (2019), the model is optimized through direct loss minimization, a

method that introduces additional bias and hyper parameter tuning. Another method, introduced in Bartler et al. (2019), leverage importance sampling to sample from the posterior, without introducing bias or new parameters. However, a new differentiable distribution close to the posterior must be introduced, limiting its usage in general settings. The authors of Jang et al. (2017a) proposed a new differentiable distribution, approximating samples from a categorical law while being differentiable. Although this method does introduce a new hyper parameter, it offers a very appealing trade-off.

**Prior.** During the second stage of the VQ-VAE training, an autoregressive prior is trained on the latent discrete space. In the original paper, the authors chose very resource intensive models for both image van den Oord et al. (2016) and audio van den Oord et al. (2018) datasets. However, more parsimonious prior models already yield encouraging results. For instance, we can find in Sun et al. (2020) a comparison of different autoregressive priors on a text-to-speech task, associated with state of the art performance, using a single layered Long Short Term Memory (LSTM) network. On the same task, the authors of Yasuda et al. (2021) were able to perform end-to-end training. Additionally, this prior model was conditioned on text data to produce speech samples.

### 4.5.3 Our model

We believe the methodology of the VQ-VAE to be pertinent, however the performance presented seem to heavily depend on the quality, and complexity of the prior model. As recent research on the choice of prior models have shown encouraging results with a variety of different architectures, we propose to model our latent discrete state with a Markovian prior. We hope to highlight the potential of modelling data with discrete states, by proposing a Hidden Markov Model whose behaviors are well known.

We define a latent state model, where we assume that the observations $(y_t)_{t=1}^T$ are independent of the commands $(u_t)_{t=1}^T$ conditionally on hidden variables $(\mathsf{z}_t)_{t=1}^T$, which take values in a discrete set of codebooks $\mathcal{E} = \{\mathrm{e}_1, \dots, \mathrm{e}_K\}$. For each of $1 \leq k \leq K$, $\mathrm{e}_k \in \mathbb{R}^D$. We then consider the following family of probability density functions :

$$p_\theta(y_{1:T}|u_{1:T}) = \int p_\theta(y_{1:T}|\mathsf{z}_{1:T}) p_\theta(\mathsf{z}_{1:T}|u_{1:T}) d\mathsf{z}_{1:T}\,, \tag{4.11}$$

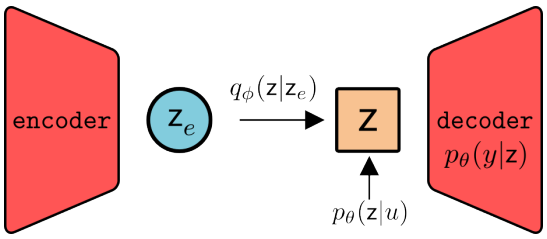
depending on an unknown parameter $\theta \in \mathbb{R}^m$.

Conditionally on the commands, we assume that the latent states $\mathsf{z}_{1:T}$ are Markovian and that the conditional law of the observations depends on past latent states, so that our model is more general than Hidden Markov Models. It also differs from other extensions of HMMs with dependencies between the observations, such as the autoregressive processes as described in Douc et al. (2004). In the following paragraphs, we detail the structure of the observation and prior models, as well as the choice of posterior family. An architecture of our model is displayed in Figure 4.8.

#### Codebooks

The codebooks represents the discrete regimes our model can switch between. Their positions in $\mathbb{R}^D$ is inferred during training, in order to provide a good partition of the space. In our definition of the model, the number of codebooks $K$ is fixed, as it is in most related works, and considered a hyper parameter. We kept the same value $K = 32$ as from the VQ-VAE paper van den Oord et al. (2017).

FIGURE 4.8 – Architecture of our proposed model. In order to generate samples conditional on the commands $u$, we draw from the prior model $p_\theta(\mathsf{z}_{1:T}|u_{1:T})$, then compute the associated prediction through the likelihood distribution $p_\theta(y_{1:T}|\mathsf{z}_{1:T})$.




**Observation model**

We consider a Gaussian observation model, and estimate at each time step its mean and variance :

$$p_\theta(y_{1:T}|\mathsf{z}_{1:T}) = \prod_{t=1}^{T} \Psi_{\mu_t,\sigma_t^2}(y_t)\,,$$

where $\Psi_{\mu,\sigma^2}$ is the probability density function of a Gaussian random variable with mean vector $\mu$ and covariance matrix $\sigma^2 I_d$. For all $1 \leq t \leq T$, $\mu_t = g_\theta^\mu(\mu_{t-1}, \mathsf{z}_{1:t-1})$ and $\sigma_t = g_\theta^\sigma(\sigma_{t-1}, \mathsf{z}_{1:t-1})$, with $\mu_0 \equiv \sigma_0 \equiv 0$. We discuss our choice for the parametric functions $g_\theta^\mu$ and $g_\theta^\sigma$ in Section 4.5.5.

**Prior model**

We assume that, conditionally on the commands $u_{1:T}$, the latent state is a discrete Markov chain. Write, for all $1 \leq \ell \leq K$, $p_{\theta,1}^\ell = p_\theta(\mathsf{z}_1 = \mathrm{e}_\ell|u_{1:T})$, and for all $2 \leq t \leq T$ and $1 \leq k, j \leq K$, $p_{\theta,t|t-1}^{k,j} = p_\theta(\mathsf{z}_t = \mathrm{e}_k|\mathsf{z}_{t-1} = \mathrm{e}_j, u_{1:T})$. The prior distribution is then defined as :

$$\log p_\theta(\mathsf{z}_{1:T}|u_{1:T}) = \sum_{k=1}^{K} \mathbb{1}_{\mathsf{z}_1=\mathrm{e}_k} \log p_{\theta,1}^k + \sum_{t=2}^{T}\sum_{j,k=1}^{K} \mathbb{1}_{\mathsf{z}_{t-1}=\mathrm{e}_j}\mathbb{1}_{\mathsf{z}_t=\mathrm{e}_k} \log p_{\theta,t|t-1}^{k,j}$$

**Posterior distribution**

As the posterior distribution of $\mathsf{z}_{1:T}$ given $y_{1:T}$ is intractable, we use a variational approach to estimate $\theta$. In the original approach proposed by van den Oord et al. (2017), the authors use an encoding function $f_\phi$, depending on an unknown parameter $\phi \in \mathbb{R}^p$, mapping the observations to a series of encoded latent variables $\mathsf{z}_{1:T}^e = f_\phi(y_{1:T})$. Then, the posterior is approximated by $q_\phi(\mathsf{z}_t = \mathrm{e}_k|y_{1:T}) = \mathbb{1}_{\mathrm{e}_*=\mathrm{e}_k}$, where $\mathrm{e}_* = \texttt{argmin}_{\mathrm{e}_\ell\in\mathcal{E}}\|\mathsf{z}_t^e - \mathrm{e}_\ell\|_2$. We propose, for $1 \leq k \leq K$ :

$$\begin{aligned} q_{\phi,t}^k = q_\phi(\mathsf{z}_t = \mathrm{e}_k|y_{1:T}) &= \texttt{softmax}(\{-\|\mathsf{z}_t^e - \mathrm{e}_\ell\|_2^2\}_{1\leq\ell\leq K})_k \\ &\propto \exp\{-\|\mathsf{z}_t^e - \mathrm{e}_k\|_2^2\}\,. \end{aligned}$$

In other words, instead of selecting the closest codebook, we sample from all available codebooks with probability proportional to their distance to the current encoded latent variable. During both training and inference, this discrete distribution encourages the exploration of the entire set of codebooks. Under the variational distribution, the latent data are assumed to be independent conditionally on the observations.

### 4.5.4 Inference procedure

The Evidence Lower BOund (ELBO) can be decomposed in three terms : for all $(\theta, \phi)$,

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi}\left[\log p_\theta(y_{1:T}|\mathsf{z}_{1:T})\right] + \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathsf{z}_{1:T}|u_{1:T})\right] - \mathbb{E}_{q_\phi}\left[\log q_\phi(\mathsf{z}_{1:T}|y_{1:T})\right] ,$$

where $\mathbb{E}_{q_\phi}$ designs the expectation under the posterior distribution $q_\phi(\mathsf{z}_{1:T}|y_{1:T})$. The last term of the ELBO can be computed explicitly as follows :

$$\mathbb{E}_{q_\phi}[\log q_\phi(\mathsf{z}_{1:T}|y_{1:T})] = \sum_{t=1}^{T}\sum_{k=1}^{K} q_{\phi,t}^k \log q_{\phi,t}^k \, .$$

We approximate the two other terms by drawing $M > 0$ samples under $q_\phi$ and computing a Monte Carlo estimator. In order for this operation to be differentiable, we use a reparametrization designed for categorical variables based on the Gumbel-Softmax distribution.

In Tocher (1955), it is shown that we can draw samples under $q_\phi$ by computing $\texttt{argmax}_{k=1}^K(log(q_\phi^k) + g_k)$, where $(g_k)_{k=1}^K$ are independent identically distributed samples from the Gumbel distribution $G(0, 1)$ with probability density $g(x) = e^{-(x+e^{-1})}$. The Gumbel-Softmax distribution Jang et al. (2017b) aims at using the `softmax` function as a continuous and differentiable approximation to the `argmax` operator. We sample $(g_1, \dots, g_K)$ independently from the Gumbel distribution and define, for all $1 \leq k \leq K$, $\pi_{k,t} \propto \exp((\log q_{\phi,t}^k + g_k)/\tau_t)$, where $\tau_t > 0$ is the softmax temperature, allowing for a smooth interpolation between a Uniform distribution (for large values of $\tau_t$), and a Categorical distribution (for small values of $\tau_t$). We propose to approximate the variational posterior distribution of $\mathsf{z}_t$ by the Dirac mass at $\widetilde{\mathsf{z}}_t = \sum_{k=1}^K \pi_{k,t}\mathrm{e}_k$. Through re-parametrization, this method allows for differentiation of the sampled latent vector $\widetilde{\mathsf{z}}_t$, with respect to the codebooks $\mathrm{e}_k$, $1 \leq k \leq K$, and the encodings $\mathsf{z}_{1:T}^e$. The first term of the ELBO can now be approximated by $(\theta, \phi) \mapsto M^{-1}\sum_{i=1}^M \log p_\theta(\widetilde{\mathsf{z}}_{1:T}|u_{1:T})$, and the second by $(\theta, \phi) \mapsto M^{-1}\sum_{i=1}^M \log p_\theta(y_{1:T}|\widetilde{\mathsf{z}}_{1:T})$.

Estimating $\theta$ and $\phi$ jointly can induce instability at the beginning of the training, leading to diminished performance after convergence. As shown in Ramesh et al. (2021), we can perform end-to-end training by penalizing the prior and posterior terms by a factor $\beta$ initialized close to zero, and then slowly increasing its value until reaching $\beta = 1$.

### 4.5.5 Experiments

We now report our choice of models and hyper parameters, and their performance on the Electricity Transformer and Relative Humidity datasets.

#### Chosen architectures

Our prior model can be decomposed in two sub-networks : an input model is responsible for extracting high level features from the commands, while the autoregressive kernel computes the Markov chain transition probabilities. This disentanglement allows us to keep the kernel simple, while still working with high level features :

$$\begin{aligned} \tilde{u}_{1:T} &= f_\theta^{\text{input\_model}}(u_{1:T}) \, , \\ h_t &= \mathrm{ker}_\theta(h_{t-1}, \tilde{u}_{t-\Delta:t}), \, \forall 1 \leq t \leq T, \, h_0 \equiv 0 \, . \end{aligned}$$

For the input model, we implemented a 3-layered LSTM, with the same latent dimension as the commands. We then compared several auto regressive architectures.

— A simple RNN cell was used as a benchmark, as they struggle to model long term dependencies.

— A Gated Recurrent Unit (GRU) cell, following results in Sun et al. (2020). These architectures have a more refined memory representation, and have shown encouraging results in our previous benchmarks.

— A kernel based on causal convolutions, as proposed in van den Oord et al. (2018), where memory is replaced by an explicit dependency on the last $\Delta = 25$ time steps.

For the encoder $f_\phi$ and decoder $g_\theta^{\mu,\sigma}$ parametric functions, we chose 3-layered LSTM networks as well.

Finally, we cross validated the number of codebooks using the Root-Mean-Square Error (RMSE), as described in Section 4.5.5 of the model, and settled for $K = 8$. The influence of each codebook on the sample generation is illustrated in Fig 4.10.

We experimented with various schedules increasing $\beta$ from 0 to 1, and found that, although this penalization is necessary to estimate all parameters jointly, the choice of schedule had little influence on the model performance. Therefore, we increase $\beta$ linearly between epochs 1 to 100, and keep $\beta = 1$ for the rest of the training, which amounts to 1500 total epochs. In all following simulations, $D = 32$.

**Benchmarked architectures.** We compared our model to the linear Gaussian Hidden Markov Model introduced in Section 3.3.4, with the same number of hidden states ($K = 8$). Transition parameters of the Markov chain, as well as mean and variance of each Gaussian are estimated through Expectation Maximization, using the Baum-Welch algorithm.

We also compared our approach to the original VQ-VAE. The architecture of the model is kept similar, to highlight the difference of methodology and training. The main differences are therefore :

— the auto encoder is trained while considering an uninformative prior ; then, we freeze the parameters of the autoencoder to estimate the parameters of the prior model.

— samples from the posterior distribution are drawn under a Dirac mass, as shown above. In order to compute the gradient, we use the straight-through estimator.

### Evaluation

Provided a sequence of commands, our model produces samples predicting the observations. By averaging them over each time step, we compute the RMSE of our model : let $\hat{x}^i_{1:T}, 1 \leq i \leq N$ be $N$ independent sequences of predictions of $x_{1:T}$, then RMSE $= (T^{-1}\sum_{t=1}^{T}(x_t - N^{-1}\sum_{i=1}^{N}\hat{x}_t^i)^2)^{1/2}$. Similarly, we report the MAE. In the following experiments, we set $N = 100$. Results on the entire validation set are displayed in Table 4.3 for the Electricity Transformer dataset and Table 4.4 for the Relative Humidity dataset, compared to the HMM, the original VQ-VAE as well as the decoupled architecture presented in Section 4.4.

We visualize the uncertainty of our model by plotting confidence intervals at each time step. We draw $N = 100$ samples $(\mathsf{z}_1, \cdots, \mathsf{z}_T)$ under the prior model, conditioned on a set of commands, then compute the associated predictions by drawing under $p_\theta(y_{1:T}|\mathsf{z}_{1:T})$. The boxplots presented in Figure 4.9 contain 95% of the generated samples, much like in Section 4.4.

TABLE 4.3 – **Electricity Transformer dataset.** Comparison of RMSE, MAE and computation time of our model against the benchmarked VQ-VAE, HMM, as well as the decoupled architecture from the previously section. This table provides aggregated results of the predictions on the entire validation set. Our model performs similarly to state of the art benchmarks in terms of metrics. Mean values of the estimators, taken over the validation samples of the dataset, are displayed along with their variance.

| | RMSE | MAE | Computation time |
|---|---|---|---|
| SMCL | $0.24 \pm 0.13$ | $0.21 \pm 0.12$ | $21ms$ |
| Ours (gru) | $0.21 \pm 0.10$ | $0.17 \pm 0.09$ | $39ms$ |
| Ours (rnn) | $0.21 \pm 0.10$ | $0.17 \pm 0.09$ | $39ms$ |
| Ours (cnn) | $0.20 \pm 0.10$ | $0.16 \pm 0.09$ | $131ms$ |
| VQ-VAE | $0.28 \pm 0.12$ | $0.24 \pm 0.12$ | $39ms$ |

TABLE 4.4 – **Relative Humidity dataset.** Comparison of RMSE, MAE and computation time of our model against the benchmarked VQ-VAE, HMM, as well as the decoupled architecture from the previously section. This table provides aggregated results of the predictions on the entire validation set. Our model performs similarly to state of the art benchmarks in terms of metrics. Mean values of the estimators, taken over the validation samples of the dataset, are displayed along with their variance.

| | RMSE | MAE | Computation time |
|---|---|---|---|
| SMCL | $0.30 \pm 0.19$ | $0.26 \pm 0.16$ | $21ms$ |
| Ours (gru) | $0.30 \pm 0.19$ | $0.24 \pm 0.15$ | $39ms$ |
| Ours (rnn) | $0.30 \pm 0.20$ | $0.25 \pm 0.17$ | $39ms$ |
| Ours (cnn) | $0.31 \pm 0.20$ | $0.25 \pm 0.16$ | $131ms$ |
| VQ-VAE | $0.55 \pm 0.33$ | $0.51 \pm 0.32$ | $39ms$ |
| HMM | $0.46 \pm 0.19$ | $0.40 \pm 0.17$ | $99ms$ |

FIGURE 4.9 – Prediction of Relative Humidity on two samples from the validation dataset. Compared to the Sequential Monte Carlo approach, our discrete latent model is less biased, as confirmed by the results presented in Table 4.4. This new approach also allows to produce coherent confidence intervals, whose sizes are stable regardless of the length of the predicted time series.

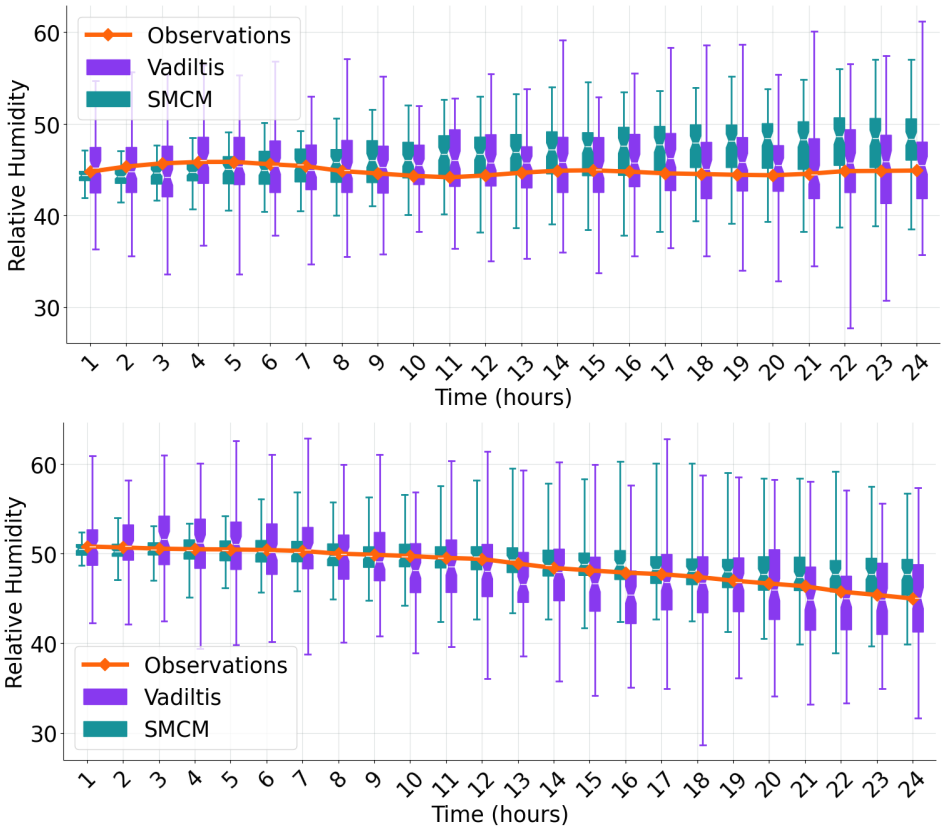
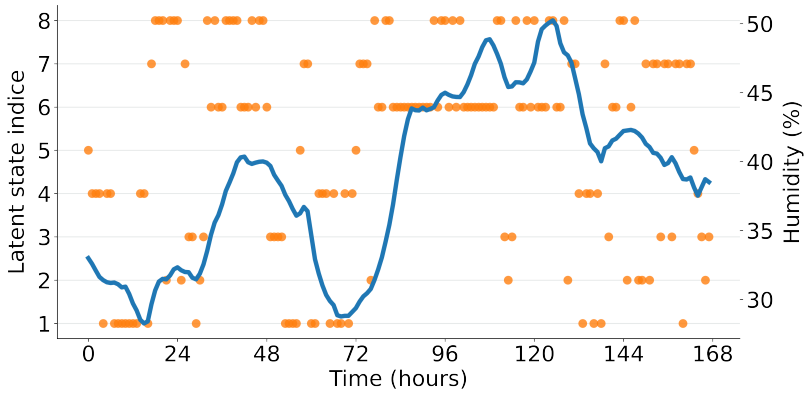


FIGURE 4.10 – Codebook usage for a week long sample. We sampled $N = 100$ trajectories from the prior model, and plotted the associated codebook index at each time step. The full line represent the observed relative humidity. Some of the codebooks appear to be linked to particular behaviors of the building, for instance codebook 6 matches an increase of relative humidity at the beginning of the day. We could now combine such a representation with segmentation algorithms in an unsupervised setting.




### 4.5.6 Discussion

In this section, we explored time series modelling with discrete latent space models. A review of the recent contributions showed that a discrete latent representation of the data can be meaningful in a wide variety of scenarios. These models offer performance on par with their continuous counterparts, as well new ways of interpreting the model, for instance through regime segmentation.

We proposed a generative model based on the Variational Auto Encoder (VAE) architecture, where the discrete posterior distribution is approximated by a family of neural network functions. Benchmarks were performed on two time series dataset, and show that this new model is able to consistently learn a relevant representation of the data. We showed that the choice of prior model does impacts performance, yet even simple architectures are able to model dependencies in the latent space. Through reparametrization, we are able to learn all parameters jointly, regardless of the number or nature of the layers composing the auto encoder and prior models. While choosing a suitable number of codebook remains an open question, the results presented in this section confirm the relevance of a discrete representation of the data, even for complex nonlinear modelling such as indoor air quality.

The experiments conducted in this chapter also present a good comparison of Sequential Monte Carlo and Variational Inference methods in the context of dependant data. Both allowed us to train complex networks, with thousands of parameters, and reach state of the art performance on two complex datasets. Sequential Monte Carlo methods fit perfectly with a decoupled model architecture, which has shown its potential in many uses cases ; additionally, SMC seamlessly fit in complex real time estimation, such as online filtering. On the other side, Variational Inference propose a computationally efficient way to train models with an absurdly high number of parameters. In our particular case, VI allows to define a single training procedure regardless of the architecture of the auto encoder or prior model. It also offer new research perspectives around the choice of prior models, which is explored in the next chapter.

# Bibliographie


Ajib, B., Lefteriu, S., Caucheteux, A., and Lecoeuche, S. (2020). Predicting the air temperature of a building zone by detecting different configurations using a switched system identification technique. *Journal of Building Engineering*, 31 :100995.

Alkhayrat, M., Aljnidi, M., and Aljoumaa, K. (2020). A comparative dimensionality reduction study in telecom customer segmentation using deep learning and pca. *Journal of Big Data*, 7 :1–23.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. (2020). Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv :2002.06470*.

Bartler, A., Wiewel, F., Mauch, L., and Yang, B. (2019). Training variational autoencoders with discrete latent variables using importance sampling. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5.

Bengio, Y. (2007). Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2 :1–127.

Bengio, Y., Simard, P. Y., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2 :157–66.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1613–1622. JMLR.org.

Brosse, N., Riquelme, C., Martin, A., Gelly, S., and Moulines, É. (2020). On last-layer algorithms for classification : Decoupling representation from uncertainty estimation. *ArXiv*, abs/2001.08049.

Burda, Y., Grosse, R. B., and Salakhutdinov, R. (2016). Importance weighted autoencoders. *CoRR*, abs/1509.00519.

Cappé, O., Moulines, É., and Rydén, T. (2005). Inference in hidden markov models. In *Springer series in statistics*.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan : Interpretable representation learning by information maximizing generative adversarial nets. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Chopin, N. and Papaspiliopoulos, O. (2020). *An Introduction to Sequential Monte Carlo*. Springer International Publishing.

Chouhan, V., Singh, S. K., Khamparia, A., Gupta, D., Tiwari, P., Moreira, C., Damaševičius, R., and de Albuquerque, V. H. C. (2020). A novel transfer learning based approach for pneumonia detection in chest x-ray images. *Applied Sciences*, 10(2) :559.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Del Moral, P., Doucet, A., and Singh, S. S. (2010). A backward interpretation of Feynman–Kac formulae. *ESAIM : Mathematical Modelling and Numerical Analysis*, 44 :947–975.

Douc, R. and Cappé, O. (2005). Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 64–69. IEEE.

Douc, R., Moulines, É., and Stoffer, D. (2013). *Nonlinear Time Series : Theory, Methods and Applications with R Examples*. Chapman & Hall.

Douc, R., Moulines, E., and Stoffer, D. (2014). *Nonlinear time series : Theory, methods and applications with R examples*. CRC press.

Douc, R., Éric Moulines, and Rydén, T. (2004). Asymptotic properties of the maximum likelihood estimator in autoregressive models with Markov regime. *The Annals of Statistics*, 32(5) :2254 – 2304.

Doucet, A., Godsill, S. J., and Andrieu, C. (2000). On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10 :197–208.

Dzunic, Z. and Fisher III, J. (2014). Bayesian switching interaction analysis under uncertainty. In Kaski, S. and Corander, J., editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 220–228. PMLR.

Elvira, V., Míguez, J., and Djurić, P. M. (2016). Adapting the number of particles in sequential monte carlo methods through an online scheme for convergence assessment. *IEEE Transactions on Signal Processing*, 65(7) :1781–1794.

Elvira, V., Miguez, J., and Djurić, P. M. (2021). On the performance of particle filters with adaptive number of particles. *Statistics and Computing*, 31(6) :1–18.

Foong, A., Burt, D., Li, Y., and Turner, R. (2020a). On the expressiveness of approximate inference in bayesian neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15897–15908. Curran Associates, Inc.

Foong, A., Burt, D., Li, Y., and Turner, R. (2020b). On the expressiveness of approximate inference in bayesian neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15897–15908. Curran Associates, Inc.

Fortunato, M., Blundell, C., and Vinyals, O. (2017). Bayesian recurrent neural networks. *arXiv preprint arXiv :1704.02798*.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. (2016). Sequential neural models with stochastic layers. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, page 2207–2215, Red Hook, NY, USA. Curran Associates Inc.

Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Gales, M. and Young, S. (2008). The application of hidden Markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3) :195–304.

Gloaguen, P., Le Corff, S., and Olsson, J. (2022). A pseudo-marginal sequential monte carlo online smoothing algorithm. *Bernoulli*, 28(4) :2606–2633.

Godsill, S. J., Doucet, A., and West, M. A. (2004). Monte carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99 :156 – 168.

Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F*, volume 140, pages 107–113. IET.

Graves, A., rahman Mohamed, A., and Hinton, G. E. (2013). Speech recognition with deep recurrent neural networks. pages 6645–6649.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-VAE : Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Hinton, G. E. and Neal, R. (1995). Bayesian learning for neural networks.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786) :504–507.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9 :1735–1780.

Jang, E., Gu, S. S., and Poole, B. (2017a). Categorical reparameterization with gumbel-softmax. *ArXiv*, abs/1611.01144.

Jang, E., Gu, S. S., and Poole, B. (2017b). Categorical reparameterization with gumbel-softmax. *ArXiv*, abs/1611.01144.

Jordan, M. I., Ghahramani, Z., Jaakkola, T., and Saul, L. K. (2004). An introduction to variational methods for graphical models. *Machine Learning*, 37 :183–233.

Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. (2018). Fast and scalable bayesian deep learning by weight-perturbation in Adam. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Kingma, D. P. and Ba, J. (2015). Adam : A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kingma, D. P., Mohamed, S., Jimenez Rezende, D., and Welling, M. (2014). Semi-supervised learning with deep generative models. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *International Conference on Learning Representations.*

Kingma, D. P. and Welling, M. (2019). An introduction to variational autoencoders. *ArXiv*, abs/1906.02691.

Kitagawa, G. (1996). Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1) :1–25.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324.

Liu, Y., Cheng, J., Zhang, H., Zou, H., and Xiong, N. (2020). Long short-term memory networks based on particle filter for object tracking. *IEEE Access*, 8 :216245–216258.

Lorberbom, G., Gane, A., Jaakkola, T., and Hazan, T. (2019). Direct optimization through argmax for discrete variational auto-encoder. In Wallach, H., Larochelle, H., Beygelzimer, A., d 'Alche-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Ma, X., Karkus, P., and Hsu, D. (2020). Particle filter recurrent neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34 :5101–5108.

Maddison, C. J., Lawson, J., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Teh, Y. (2017). Filtering variational objectives. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*.

Martin, A., Ollion, C., Strub, F., Corff, S. L., and Pietquin, O. (2020). The monte carlo transformer : a stochastic self-attention model for sequence prediction. *ArXiv*, abs/2007.08620.

Mnih, A. and Rezende, D. (2016). Variational inference for monte carlo objectives. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2188–2196, New York, New York, USA. PMLR.

Moon, T., Choi, H., Lee, H., and Song, I. (2015). Rnndrop : A novel dropout for rnns in asr. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 65–70.

Naesseth, C., Linderman, S., Ranganath, R., and Blei, D. (2018). Variational Sequential Monte Carlo. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Nguyen, N. M., Le Corff, S., and Moulines, É. (2017). Particle rejuvenation of rao-blackwellized sequential monte carlo smoothers for conditionally linear and gaussian models. *EURASIP Journal on Advances in Signal Processing*, 2017(1) :1–15.

Olsson, J. and Westerborn, J. (2014). Efficient particle-based online smoothing in general hidden markov models : the paris algorithm. *Bernoulli*, 23 :1951–1996.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10) :1345–1359.

Patterson, T. A., Parton, A., Langrock, R., Blackwell, P. G., Thomas, L., and King, R. (2017). Statistical modelling of individual animal movement : an overview of key methods and a discussion of practical challenges. *AStA Advances in Statistical Analysis*, 101(4) :399–438.

Pearce, T., Brintrup, A., Zaki, M., and Neely, A. (2018). High-quality prediction intervals for deep learning : A distribution-free, ensembled approach. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4075–4084. PMLR.

Petetin, Y., Janati, Y., and Desbouvries, F. (2021). Structured variational bayesian inference for gaussian state-space models with regime switching. *IEEE Signal Processing Letters*, 28 :1953–1957.

Pham, V., Bluche, T., Kermorvant, C., and Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 285–290.

Pitt, M. K. and Shephard, N. (1999). Filtering via simulation : Auxiliary particle filters. *J. Amer. Statist. Assoc.*, 94(446) :590–599.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2022). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR.

Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In van Dyk, D. and Welling, M., editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455. PMLR.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56) :1929–1958.

Sun, G., Zhang, Y., Weiss, R. J., Cao, Y., Zen, H., Rosenberg, A., Ramabhadran, B., and Wu, Y. (2020). Generating diverse and natural text-to-speech samples using a quantized fine-grained vae and autoregressive prosody prior. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6699–6703.

Teye, M., Azizpour, H., and Smith, K. (2018). Bayesian uncertainty estimation for batch normalized deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Tocher, K. D. (1955). Statistical theory of extreme values and some practical applications : A series of lectures. *Journal of the Royal Statistical Society. Series A (General)*, 118(1) :106–106.

van den Oord, A., Kalchbrenner, N., Espeholt, L., kavukcuoglu, k., Vinyals, O., and Graves, A. (2016). Conditional image generation with pixelcnn decoders. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. (2018). Parallel wavenet : Fast high-fidelity speech synthesis. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3918–3926. PMLR.

van den Oord, A., Vinyals, O., and kavukcuoglu, k. (2017). Neural discrete representation learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vandal, T., Livingston, M., Piho, C., and Zimmerman, S. (2018). Prediction and uncertainty quantification of daily airport flight delays. In Hardgrove, C., Dorard, L., and Thompson, K., editors, *Proceedings of The 4th International Conference on Predictive Applications and APIs*, volume 82 of *Proceedings of Machine Learning Research*, pages 45–51. PMLR.

Wen, M. and Tadmor, E. (2020). Uncertainty quantification in molecular simulations with dropout neural network potentials. *npj Computational Materials*, 6 :1–10.

Wilks, D. (1998). Multisite generalization of a daily stochastic precipitation generation model. *Journal of Hydrology*, 210(1) :178–191.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell : Neural image caption generation with visual attention. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057. PMLR.

Yasuda, Y., Wang, X., and Yamagishd, J. (2021). End-to-end text-to-speech using latent duration based on vq-vae. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5694–5698.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks ? In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer : Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12) :11106–11115.

# Chapitre 5

# Prior models for discrete latent states

In the previous chapter, we explored latent space models applied to the forecasting of hourly relative humidity records in a building. We showed that a discrete representation of this data allows for simpler parameter estimation and better interpretability, without degrading the performance of the model, in terms of precision and quality of confidence intervals. Whereas we limited ourselves to simple Markovian prior models, in this chapter we experiment with recent advances in complex distribution modelling, such as diffusion bridges.

This chapter is organized as follows : in Section 5.1, we introduce the limitation behind using autoregressive priors, as well as the appeal of diffusion probabilistic models, before proposing a new mathematical framework for discrete latent models allowing estimating all parameters jointly. We review related works in Section 5.2, before presenting our methodology in Section 5.3. In Section 5.4, we present a set of experiments conducted on a toy problem, two image datasets and the relative humidity forecasting task. Due to limitations arising during the training phase, we propose an extension of diffusion applied to discrete data in Section 5.5, and discuss the overhaul results of this new approach in Section 5.6.

The results presented in this chapter are adapted from the following contribution : *Diffusion Bridges Vector Quantized Variational Autoencoders*, Cohen, M., Quispe, Q., Le Corff, S., Ollion, C., Moulines, E., Proceedings of the 39th International Conference on Machine Learning, Volume 162.

## 5.1 Introduction

Vector Quantized-Variational AutoEncoders (VQ-VAE) are generative models based on discrete latent representations of the data, where inputs are mapped to a finite set of learned embeddings. To generate new samples, an autoregressive prior distribution over the discrete states must be trained separately. This prior is generally very complex and leads to slow generation. Additionally, the implementation of VQ-VAE relies on many practical tricks, already highlighted in the previous chapter, see Section 4.5.2. Despite these limitations, the VQ-VAE and its derived models have been successfully applied in image and speech generation Oord et al. (2017); Esser et al. (2021); Ramesh et al. (2021). One particular limitation we now address revolves around the prior model.

Firstly, when training the VQ-VAE, the prior distribution of the discrete variables is initially assumed to be uninformative, in practice uniform. Only in a subsequent training step, high-dimensional autoregressive models such as the PixelCNN van den Oord et al. (2016); Salimans et al. (2017); Chen et al. (2018) or WaveNet Oord et al. (2016) are estimated to obtain a complex prior distribution. Joint training of the prior and the Auto-Encoder is a challenging task for which no satisfactory solution exists yet.

Secondly, the autoregressive nature of the proposed prior models (PixelCNN and WaveNet) has several draw-

backs in the general case, which are the same in the observation and latent space. The data is assumed to have a fixed sequential order, which forces the generation to start at a certain point. For instance, when modelling an image, such a prior would typically start modelling the upper left corner, then span the image in an arbitrary way. At each step, a new latent variable is sampled using the previously sampled pixels. This is also the case when modelling a complex, high dimensional lattice of latent vectors. During inference, the model may then accumulate prediction errors. Additionally, the runtime process, which depends mainly on the number of network evaluations, is sequential and depends on the size of the image or the multi-dimensional lattice. The influence of the prior is further explored in Razavi et al. (2019), where VQ-VAE is used to sample images on a larger scale, using two layers of discrete latent variables, and Willetts et al. (2021) use hierarchical discrete VAEs with numerous layers of latent variables. Other works such as Esser et al. (2021); Ramesh et al. (2021) have used Transformers to autoregressively model a sequence of latent variables : while these works benefit from the recent advances of Transformers for large language models, their autoregressive process still suffers from the same drawbacks as PixelCNN-like priors.

A promising class of models that depart from autoregressive models are Diffusion Probabilistic Models Sohl-Dickstein et al. (2015); Ho et al. (2020) and closely related Score-Matching Generative Models Song and Ermon (2019); De Bortoli et al. (2021). The general idea is to apply a corrupting Markovian process on the data through $T$ corrupting steps and learn a neural network that gradually *denoises* or reconstructs the original samples from the noisy data. For example, when sampling images, an initial sample is drawn from an uninformative distribution and reconstructed iteratively using the trained Markov kernel. This process is applied to all pixels simultaneously, so no fixed order is required and the sampling time does not depend on sequential predictions that depend on the number of pixels, but on the number of steps $T$. While this number of steps can be large ($T = 1000$ is typical), simple improvements enable to reduce it dramatically and obtain $\times 50$ speedups Song et al. (2021). These properties have led diffusion probability models to receive much attention in the context of continuous input modelling.

Our work addresses both limitations by introducing a new mathematical framework that extends the VQ-VAE to non autoregressive priors. The main claim of our proposed approach is that using diffusion bridges in a continuous space is a very efficient way to learn complex discrete distributions, with support on a large space. We propose the following contributions :

— We develop a new mathematical framework for quantized latent models, that extends and generalizes the standard VQ-VAE. Our method enables end-to-end training and, in particular, bypasses the separate training of an autoregressive prior.

— To this end, we build a diffusion bridge between a continuous coded vector and a non-informative prior distribution. The latent discrete states are then given as random functions of these continuous vectors.

— We show that our model is competitive with the autoregressive prior on the mini-Imagenet and CIFAR dataset and is efficient in both optimization and sampling. We also demonstrate an application for time series, by improving on the variational model presented previously, see Section 4.5.3.

Figure 5.1 describes the complete architecture of our model.

## 5.2 Related Works

**Discrete Generative denoising.** There exists approaches to diffusion probabilistic models, such as presented for the diffusion bridges, in the context of discrete data. In Hoogeboom et al. (2021), the authors propose multinomial diffusion to gradually add categorical noise to discrete samples for which the generative denoising process is learned. Unlike alternatives such as normalizing flows (see Kobyzev et al. (2021) for a review), the diffusion proposed by the authors for discrete variables does not require gradient approximations because the parameter of the diffusion is fixed.

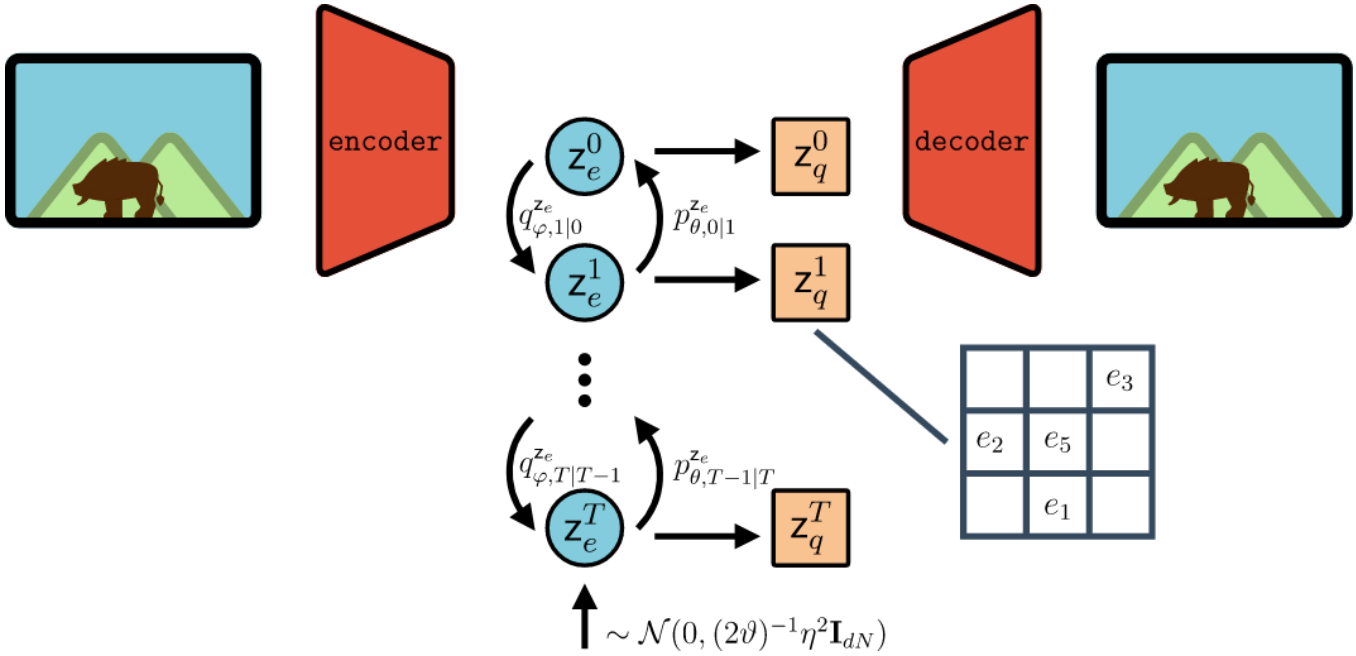



FIGURE 5.1 – Our proposed architecture, for a prior based on a Ornstein-Uhlenbeck bridge. The top pathway from *input image* to $z_e^0$, to $z^0$, to *reconstructed image* resembles the original VQ-VAE model. The vertical pathway from $(z_e^0, z^0)$ to $(z_e^T, z^T)$ and backwards is based on a denoising diffusion process. See Section 5.3.2 and Algorithm 3 for the corresponding sampling procedure.

Such diffusion models are optimized using variational inference to learn the denoising process, i.e., the bridge that aims at inverting the multinomial diffusion. In Hoogeboom et al. (2021), the authors propose a variational distribution based on bridge sampling. In Austin et al. (2021), the authors improve the idea by modifying the transition matrices of the corruption scheme with several tricks. The main one is the addition of absorbing states in the corruption scheme by replacing a discrete value with a MASK class, inspired by recent Masked Language Models like BERT. In this way, the corrupted dimensions can be distinguished from the original ones instead of being uniformly sampled. One drawback of their approach, mentioned by the authors, is that the transition matrix does not scale well for a large number of embedding vectors, which is typically the case in VQ-VAE.

Compared to discrete generative denoising, our approach takes advantage of the fact that the discrete latent distribution depends solely on an auxiliary continuous distribution. We derive a novel model based on continuous-discrete diffusion that we believe is simpler and more scalable than the models mentioned in this section.

**Generative denoising applied to a latent space.** Instead of modelling the data directly, Vahdat et al. (2021) propose to perform score matching in a latent space. The authors propose a complete generative model and are able to train the encoder/decoder and score matching end-to-end. Their method also achieve excellent visual patterns and results but relies on a number of optimization heuristics necessary for stable training. In Mittal et al. (2021), the authors have also applied such an idea in a generative music model. Instead of working in a continuous latent space, our method is specifically designed for a discrete latent space as in VQ-VAEs.

In the model proposed by Gu et al. (2021), the autoregressive prior is replaced by a discrete generative denoising process, which is perhaps closer to our idea. However, the authors focus more on a text-image synthesis task where the generative denoising model is traine based on an input text : it generates a set of discrete visual tokens given a sequence of text tokens. They also consider the VQ-VAE as a trained model and focus only on the generation of latent variables. This work focuses instead on deriving a full generative model with a sound probabilistic

interpretation that allows it to be trained end-to-end.

## 5.3 Diffusion bridges VQ-VAE

### 5.3.1 Model and loss function

Assume that the distribution of the input $y \in \mathbb{R}^m$ depends on a hidden discrete state $\mathsf{z} \in \mathcal{E} = \{\mathrm{e}_1, \ldots, \mathrm{e}_K\}$ with $\mathrm{e}_k \in \mathbb{R}^d$ for all $1 \leqslant k \leqslant K$. Let $p_\theta$ be the joint probability density of $(\mathsf{z}, y)$

$$(\mathsf{z}, y) \mapsto p_\theta(\mathsf{z}, y) = p_\theta(\mathsf{z}) p_\theta(y|\mathsf{z}) \,,$$

where $\theta \in \mathbb{R}^p$ are unknown parameters. Consider first an encoding function $f_\phi$ and write $\mathsf{z}_e(y) = f_\phi(y)$ the encoded data. In the original VQ-VAE, the authors proposed $q_\phi(\mathsf{z}|y) = \delta_{\mathrm{e}_{k_y^*}}(\mathsf{z})$ , as the variational distribution to approximate $p_\theta(\mathsf{z}|y)$, where $\delta$ is the Dirac mass and $k_y^* = \mathrm{argmin}_{1 \leqslant k \leqslant K} \{\|\mathsf{z}_e(y) - \mathrm{e}_k\|_2\}$, where $\phi \in \mathbb{R}^r$ are all the variational parameters.

In this paper, we introduce a diffusion-based generative VQ-VAE. This model allows to propose a VAE approach with an efficient joint training of the prior and the variational approximation. Assume that $\mathsf{z}$ is a sequence, i.e. $\mathsf{z} = \mathsf{z}^{0:T}$, where the superscript refers to the time in the diffusion process. Consider the following joint probability distribution

$$p_\theta(\mathsf{z}^{0:T}, y) = p_\theta^{\mathsf{z}}(\mathsf{z}^{0:T}) p_\theta^y(y|\mathsf{z}^0) \,.$$

The latent discrete state $\mathsf{z}^0$ used as input in the decoder is the final state of the chain $(\mathsf{z}^T, \ldots, \mathsf{z}^0)$. We further assume that $p_\theta^{\mathsf{z}}(\mathsf{z}^{0:T})$ is the marginal distribution of

$$p_\theta(\mathsf{z}^{0:T}, \mathsf{z}_e^{0:T}) = p_{\theta,T}^{\mathsf{z}_e}(\mathsf{z}_e^T) p_{\theta,T}^{\mathsf{z}}(\mathsf{z}^T|\mathsf{z}_e^T) \prod_{t=0}^{T-1} p_{\theta,t|t+1}^{\mathsf{z}_e}(\mathsf{z}_e^t|\mathsf{z}_e^{t+1}) p_{\theta,t}^{\mathsf{z}}(\mathsf{z}^t|\mathsf{z}_e^t) \,.$$

In this setting, $\{\mathsf{z}_e^t\}_{0 \leqslant t \leqslant T}$ are continuous latent states in $\mathbb{R}^{d \times N}$ and conditionally on $\{\mathsf{z}_e^t\}_{0 \leqslant t \leqslant T}$ the $\{\mathsf{z}^t\}_{0 \leqslant t \leqslant T}$ are independent with discrete distribution with support $\mathcal{E}^N$. This means that we model jointly $N$ latent states as this is useful for many applications such as image generation or time series forecasting. The continuous latent state is assumed to be a Markov chain and at each time step $t$ the discrete variable $\mathsf{z}^t$ is a random function of the corresponding $\mathsf{z}_e^t$. Although the continuous states are modeled as a Markov chain, the discrete variables arising therefrom have a more complex statistical structure (and in particular are not Markovian).

The prior distribution of $\mathsf{z}_e^T$ is assumed to be uninformative ; the density of the final latent state $\mathsf{z}_e^0$ can be factorized as a sequence of denoising transition densities $\{p_{\theta,t|t+1}^{\mathsf{z}_e}\}_{0 \leqslant t \leqslant T-1}$. Only this last state is mapped to the embedding space and decoded to compute the conditional law of the data given the latent states.

Since the conditional law $p_\theta(\mathsf{z}^{0:T}, \mathsf{z}_e^{0:T}|y)$ is not available explicitly, this work focuses on variational approaches to provide an approximation. Then, consider the following variational family :

$$q_\phi(\mathsf{z}^{0:T}, \mathsf{z}_e^{0:T}|y) = \delta_{\mathsf{z}_e(y)}(\mathsf{z}_e^0) q_{\phi,0}^{\mathsf{z}}(\mathsf{z}^0|\mathsf{z}_e^0) \prod_{t=1}^{T} \left\{ q_{\phi,t|t-1}^{\mathsf{z}_e}(\mathsf{z}_e^t|\mathsf{z}_e^{t-1}) q_{\phi,t}^{\mathsf{z}}(\mathsf{z}^t|\mathsf{z}_e^t) \right\} \,.$$

The family $\{q_{\phi,t|t-1}^{\mathsf{z}_e}\}_{1 \leqslant t \leqslant T}$ of forward "noising" transition densities are chosen to be the transition densities of a continuous-time process $(Z_t)_{t \geqslant 0}$ with $Z_0 = \mathsf{z}_e(y)$. Sampling the diffusion bridge $(\tilde{Z}_t)_{t \geqslant 0}$, i.e. the law of the process $(Z_t)_{t \geqslant 0}$ conditioned on $Z_0 = \mathsf{z}_e(y)$ and $Z_T = \mathsf{z}_e^T$ is a challenging problem for general diffusions, see for instance Beskos et al. (2008); Lin et al. (2010); Bladt et al. (2016). By the Markov property, the marginal density at time $t$ of

this conditioned process is given by :

$$\tilde{q}^{\mathsf{z}_e}_{\phi,t|0,T}(\mathsf{z}^t_e|\mathsf{z}^0_e,\mathsf{z}^T_e) = \frac{q^{\mathsf{z}_e}_{\phi,t|0}(\mathsf{z}^t_e|\mathsf{z}^0_e)q^{\mathsf{z}_e}_{\phi,T|t}(\mathsf{z}^T_e|\mathsf{z}^t_e)}{q^{\mathsf{z}_e}_{\phi,T|0}(\mathsf{z}^T_e|\mathsf{z}^0_e)}\,. \tag{5.1}$$

The Evidence Lower BOund (ELBO) is then defined, for all $(\theta,\phi)$, as

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi}\left[\log\frac{p_\theta(\mathsf{z}^{0:T},\mathsf{z}^{0:T}_e,y)}{q_\phi(\mathsf{z}^{0:T},\mathsf{z}^{0:T}_e|y)}\right], \tag{5.2}$$

where $\mathbb{E}_{q_\phi}$ is the expectation under $q_\phi(\mathsf{z}^{0:T},\mathsf{z}^{0:T}_e|y)$.

**Lemma 5.3.1.** *For all $(\theta,\phi)$, the ELBO $\mathcal{L}(\theta,\phi)$ is :*

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi}\left[\log p^y_\theta(y|\mathsf{z}^0)\right] + \sum_{t=0}^{T}\mathcal{L}_t(\theta,\phi) + \sum_{t=0}^{T}\mathbb{E}_{q_\phi}\left[\log\frac{p^{\mathsf{z}}_{\theta,t}(\mathsf{z}^t|\mathsf{z}^t_e)}{q^{\mathsf{z}}_{\phi,t}(\mathsf{z}^t|\mathsf{z}^t_e)}\right],$$

*where, for $1\leqslant t\leqslant T-1$,*

$$\begin{aligned}
\mathcal{L}_0(\theta,\phi) &= \mathbb{E}_{q_\phi}\left[\log p^{\mathsf{z}_e}_{\theta,0|1}(\mathsf{z}^0_e|\mathsf{z}^1_e)\right],\\
\mathcal{L}_t(\theta,\phi) &= \mathbb{E}_{q_\phi}\left[\log\frac{p^{\mathsf{z}_e}_{\theta,t-1|t}(\mathsf{z}^{t-1}_e|\mathsf{z}^t_e)}{q^{\mathsf{z}_e}_{\phi,t-1|0,t}(\mathsf{z}^{t-1}_e|\mathsf{z}^0_e,\mathsf{z}^t_e)}\right],\\
\mathcal{L}_T(\theta,\phi) &= \mathbb{E}_{q_\phi}\left[\log\frac{p^{\mathsf{z}_e}_{\theta,T}(\mathsf{z}^T_e)}{q^{\mathsf{z}_e}_{\phi,T|0}(\mathsf{z}^T_e|\mathsf{z}^0_e)}\right].
\end{aligned}$$

*Démonstration.* The proof is standard and postponed to Appendix A.2.1. □

The three terms of the objective function can be interpreted as follows :

$$\mathcal{L}(\theta,\phi) = \mathcal{L}^{rec}(\theta,\phi) + \sum_{t=0}^{T}\mathcal{L}_t(\theta,\phi) + \sum_{t=0}^{T}\mathcal{L}^{reg}_t(\theta,\phi)$$

with $\mathcal{L}^{rec} = \mathbb{E}_{q_\phi}[\log p^y_\theta(y|\mathsf{z}^0)]$ a reconstruction term, $\mathcal{L}_t$ the diffusion term, and an extra term

$$\mathcal{L}^{reg}_t = \mathbb{E}_{q_\phi}\left[\log\frac{p^{\mathsf{z}}_{\theta,t}(\mathsf{z}^t|\mathsf{z}^t_e)}{q^{\mathsf{z}}_{\varphi,t}(\mathsf{z}^t|\mathsf{z}^t_e)}\right], \tag{5.3}$$

which may be seen as a regularization term as discussed in next sections.

### 5.3.2 Application to Ornstein-Uhlenbeck processes

Consider for instance the following Stochastic Differential Equation (SDE) to add noise to the normalized inputs :

$$\mathrm{d}Z_t = -\vartheta(Z_t - z_*)\mathrm{d}t + \eta\mathrm{d}W_t\,, \tag{5.4}$$

where $\vartheta,\eta>0$, $z_*\in\mathbb{R}^{d\times N}$ is the target state at the end of the noising process and $\{W_t\}_{0\leqslant t\leqslant T}$ is a standard Brownian motion in $\mathbb{R}^{d\times N}$. We can define the variational density by integrating this SDE along small step-sizes. Let $\delta_t$ be the time step between the two consecutive latent variables $\mathsf{z}^{t-1}_e$ and $\mathsf{z}^t_e$. In this setting, $q^{\mathsf{z}_e}_{\phi,t|t-1}(\mathsf{z}^t_e|\mathsf{z}^{t-1}_e)$ is a Gaussian probability density function with mean $z_* + (\mathsf{z}^{t-1}_e - z_*)\mathrm{e}^{-\vartheta\delta_t}$ in $\mathbb{R}^{d\times N}$ and covariance matrix $(2\vartheta)^{-1}\eta^2(1-\mathrm{e}^{-2\vartheta\delta_t})\mathbf{I}_{dN}$,

where for all $n \geqslant 1$, $\mathbf{I}_n$ is the identity matrix with size $n \times n$. Asymptotically the process is a Gaussian with mean $z_*$ and variance $\eta^2(2\vartheta)^{-1}\mathbf{I}_{dN}$.

The denoising process amounts then to sampling from the bridge associated with the SDE, i.e. sampling $\mathsf{z}_e^{t-1}$ given $\mathsf{z}_e^0$ and $\mathsf{z}_e^t$. The law of this bridge is explicit for the Ornstein-Uhlenbeck diffusion (5.7). Using (5.1),

$$\tilde{q}^{\mathsf{z}_e}_{\phi,s|0,t}(\mathsf{z}_e^s|\mathsf{z}_e^t,\mathsf{z}_e^0) \propto q^{\mathsf{z}_e}_{\varphi,s|0}(\mathsf{z}_e^{t-1}|\mathsf{z}_e^0) q^{\mathsf{z}_e}_{\varphi,t|s}(\mathsf{z}_e^t|\mathsf{z}_e^s)\,,$$

where $0 \leqslant s \leqslant t$, so that $\tilde{q}^{\mathsf{z}_e}_{\phi,t-1|0,t}(\mathsf{z}_e^{t-1}|\mathsf{z}_e^t,\mathsf{z}_e^0)$ is a Gaussian probability density function with mean

$$\tilde{\mu}_{\phi,t-1|0,t}(\mathsf{z}_e^0,\mathsf{z}_e^t) = \frac{\beta_t}{1-\bar{\alpha}_t}\left(z_* + \sqrt{\bar{\alpha}_{t-1}}(\mathsf{z}_e^0 - z_*)\right) + \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\sqrt{\alpha}_t\left(\mathsf{z}_e^t - (1-\sqrt{\alpha_t})z_*\right)$$

and covariance matrix

$$\tilde{\sigma}^2_{\phi,t-1|0,t} = \frac{\eta^2}{2\vartheta}\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\,\mathbf{I}_{dN}\,,$$

where $\beta_t = 1-\exp(-2\vartheta\delta_t)$, $\alpha_t = 1-\beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. Note that the bridge sampler proposed in Ho et al. (2020) is a specific case of this setting with $\eta = \sqrt{2}$, $z_* = 0$ and $\vartheta = 1$.

**Choice of denoising model $p_\theta$.** Following Ho et al. (2020), we propose a Gaussian distribution for $p^{\mathsf{z}_e}_{\theta,t-1|t}(\mathsf{z}_e^{t-1}|\mathsf{z}_e^t)$ with mean $\mu_{\theta,t-1|t}(\mathsf{z}_e^t,t)$ and variance $\sigma^2_{\theta,t-1|t}\,\mathbf{I}_{dN}$. In the following, we choose

$$\sigma^2_{\theta,t-1|t} = \frac{\eta^2}{2\vartheta}\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

so that the term $\mathcal{L}_t$ of Lemma 5.3.1 writes

$$2\sigma^2_{\theta,t-1|t}\mathcal{L}_t(\theta,\phi) = -\mathbb{E}_{q_\phi}\left[\left\|\mu_{\theta,t-1|t}(\mathsf{z}_e^t,t) - \tilde{\mu}_{\varphi,t-1|0,t}(\mathsf{z}_e^0,\mathsf{z}_e^t)\right\|_2^2\right]\,.$$

In addition, under $q_\phi$, $\mathsf{z}_e^t$ has the same distribution as

$$\mathsf{h}_e^t(\mathsf{z}_e^0,\varepsilon_t) = z_* + \sqrt{\bar{\alpha}_t}(\mathsf{z}_e^0 - z_*) + \sqrt{\frac{\eta^2}{2\vartheta}(1-\bar{\alpha}_t)}\varepsilon_t\,,$$

where $\varepsilon_t \sim \mathcal{N}(0,\mathbf{I}_{dN})$. Then, for instance in the case $z_* = 0$, $\tilde{\mu}_{\phi,t-1|0,t}$ can be reparameterised as follows :

$$\tilde{\mu}_{\phi,t-1|0,t}(\mathsf{z}_e^0,\mathsf{z}_e^t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathsf{h}_e^t(\mathsf{z}_e^0,\varepsilon_t) - \sqrt{\frac{\eta^2}{2\vartheta(1-\bar{\alpha}_t)}}\beta_t\varepsilon_t\right)\,.$$

We therefore propose to use

$$\mu_{\theta,t-1|t}(\mathsf{z}_e^t,t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathsf{z}_e^t - \sqrt{\frac{\eta^2}{2\vartheta(1-\bar{\alpha}_t)}}\beta_t\varepsilon_\theta(\mathsf{z}_e^t,t)\right)\,,$$

which yields

$$\mathcal{L}_t(\theta,\phi) = \frac{-\beta_t}{2\alpha_t(1-\bar{\alpha}_{t-1})}\mathbb{E}\left[\left\|\varepsilon_t - \varepsilon_\theta(\mathsf{h}_e^t(\mathsf{z}_e^0,\varepsilon_t),t)\right\|_2^2\right]\,. \tag{5.5}$$

Several choices can be proposed to model the function $\varepsilon_\theta$. The deep learning architectures considered in the numerical experiments are discussed in Appendix A.2.4 and A.2.5. Similarly to Ho et al. (2020), we use a stochastic version of our loss function : sample $t$ uniformly in $\{0,\dots,T\}$, and consider $\mathcal{L}_t(\theta,\phi)$ instead of the full sum over all $t$.

**Algorithm 2:** Training procedure

**repeat**
- Compute $\mathsf{z}_e^0 = f_\phi(y)$
- Sample $\hat{\mathsf{z}}^0 \sim q_\phi(\mathsf{z}^0|\mathsf{z}_e^0)$
- Compute $\hat{\mathcal{L}}^{rec}(\theta,\phi) = \log p_\theta^y(y|\hat{\mathsf{z}}^0)$
- Sample $t \sim Uniform(\{0,\dots,T\})$
- Sample $\varepsilon_t \sim \mathcal{N}(0, \mathbf{I}_{dN})$
- Sample $\mathsf{z}_e^t \sim q_\phi(\mathsf{z}_e^t|\mathsf{z}_e^0)$ (using $\varepsilon_t$)
- Compute $\hat{\mathcal{L}}_t(\theta,\phi)$ from $\varepsilon_\theta(\mathsf{z}_e^t, t)$ and $\varepsilon_t$ using (5.5)
- Compute $\hat{\mathcal{L}}_t^{reg}(\theta,\phi)$ from $\mathsf{z}_e^t$ *(see text)*
- $\hat{\mathcal{L}}(\theta,\phi) = \hat{\mathcal{L}}^{rec}(\theta,\varphi) + \hat{\mathcal{L}}_t(\theta,\varphi) + \hat{\mathcal{L}}_t^{reg}(\theta,\varphi)$
- Perform SGD step on $-\hat{\mathcal{L}}(\theta,\phi)$

**until** convergence

The final training algorithm is described in Algorithm 2 and the sampling procedure in Algorithm 3.

**Connections with the VQ-VAE loss function.** In the special case where $T = 0$, our loss function can be reduced to a standard VQ-VAE loss function. In that case, write $\mathsf{z} = \mathsf{z}^0$ and $\mathsf{z}_e = \mathsf{z}_e^0$, the ELBO then becomes :

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\varphi}\left[\log p_\theta^y(y|\mathsf{z})\right] + \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta^{\mathsf{z}}(\mathsf{z}|\mathsf{z}_e)}{q_\varphi^{\mathsf{z}}(\mathsf{z}|\mathsf{z}_e)}\right],$$

Then, if we assume that $p_\theta^{\mathsf{z}}(\mathsf{z}|\mathsf{z}_e) = \mathrm{Softmax}\{-\|\mathsf{z}_e - \mathrm{e}_k\|_2^2\}_{1\le k\le K}$ and that $q_\phi^{\mathsf{z}}(\mathsf{z}|\mathsf{z}_e)$ is as in Oord et al. (2017), i.e. a Dirac mass at $\widehat{\mathsf{z}} = \mathrm{argmin}_{1\le k\le K}\|\mathsf{z}_e - \mathrm{e}_k\|_2^2$, up to an additive constant, this yields the following random estimation of $\mathbb{E}_{q_\varphi}[\log p_\theta^{\mathsf{z}}(\mathsf{z}|\mathsf{z}_e)/q_\varphi^{\mathsf{z}}(\mathsf{z}|\mathsf{z}_e)]$,

$$\widehat{\mathcal{L}}_{\mathsf{z}}^{reg}(\theta,\phi) = \|\mathsf{z}_e - \widehat{\mathsf{z}}\|_2 + \log\left(\sum_{k=1}^K \exp\left\{-\|\mathsf{z}_e - \mathrm{e}_k\|_2\right\}\right).$$

The first term of this loss is the loss proposed in Oord et al. (2017) which is then split into two parts using the stop gradient operator. The last term is simply the additional normalizing term of $p_\theta^{\mathsf{z}}(\mathsf{z}|\mathsf{z}_e)$.

**Connecting diffusion and discretisation.** Similar to the VQ-VAE case above, it is possible to consider only the term $\mathcal{L}_0^{reg}(\theta,\phi)$ in the case $T > 0$. However, our framework allows for much flexible parameterisation of $p_{\theta,t}^{\mathsf{z}}(\mathsf{z}^t|\mathsf{z}_e^t)$ and $q_{\varphi,t}^{\mathsf{z}}(\mathsf{z}^t|\mathsf{z}_e^t)$. For instance, the Gumbel-Softmax trick provides an efficient and differentiable parameterisation. A sample $\mathsf{z}^t \sim p_{\theta,t}^{\mathsf{z}}(\mathsf{z}^t|\mathsf{z}_e^t)$ (resp. $\mathsf{z}^t \sim q_{\varphi,t}^{\mathsf{z}}(\mathsf{z}^t|\mathsf{z}_e^t)$) can be obtained by sampling with probabilities proportional to $\{\exp\{(-\|\mathsf{z}_e - \mathrm{e}_k\|_2^2 + G_k)/\tau_t\}\}_{1\le k\le K}$ (resp. $\{\exp\{(-\|\mathsf{z}_e - \mathrm{e}_k\|_2^2 + \tilde{G}_k)/\tau\}\}_{1\le k\le K}$), where $\{(G_k, \tilde{G}_k)\}_{1\le k\le K}$ are i.i.d. with distribution $\mathrm{Gumbel}(0,1)$, $\tau > 0$, and $\{\tau_t\}_{0\le t\le T}$ are positive time-dependent scaling parameters. In practice, the third part of the objective function can be computed efficiently, by using a stochastic version of the ELBO, computing a single $\mathcal{L}_t^{reg}(\theta,\varphi)$ instead of the sum (we use the same $t$ for both parts of the ELBO). The term reduces to :

$$\mathcal{L}_t^{reg}(\theta,\phi) = -\mathrm{KL}(q_\varphi(\mathsf{z}^t|\mathsf{z}_e^t)\|p_\theta(\mathsf{z}^t|\mathsf{z}_e^t)). \tag{5.6}$$

This terms connects the diffusion and quantization parts as it creates a gradient pathway through a step $t$ of the diffusion process, acting as a regularisation on the codebooks and $\mathsf{z}_e^t$. Intuitively, maximizing $\mathcal{L}_t^{reg}(\theta,\phi)$ accounts for pushing codebooks and $\mathsf{z}_e^t$ together or apart depending on the choice of $\tau, \tau_t$. The final end-to-end training algorithm is described in Algorithm 2, and further considerations are provided in Appendix A.2.3.

**Algorithm 3:** Sampling procedure (for $z_* = 0$)

Sample $\mathsf{z}_e^T \sim \mathcal{N}(0, (2\vartheta)^{-1}\eta^2 \mathbf{I}_{dN})$
**for** $t = T$ **to** $1$ **do**
  Set $\mathsf{z}_e^{t-1} = \alpha_t^{-1/2}\left(\mathsf{z}_e^t - \sqrt{\frac{\eta^2}{2\vartheta(1-\bar{\alpha}_t)}}\beta_t \varepsilon_\theta(\mathsf{z}_e^t, t)\right)$
**end for**
Sample $\mathsf{z}^0 \sim p^{\mathsf{z}}_{\theta,0}(\mathsf{z}^0|\mathsf{z}_e^0)$ {*quantization*}
Sample $y \sim p^y_\theta(y|\mathsf{z}^0)$ {*decoder*}

## 5.4 Experiments

### 5.4.1 Toy Experiment

In order to understand the proposed denoising procedure for VQ-VAE, consider a simple toy setting in which there is no encoder nor decoder, and the codebooks $\{\mathrm{e}_j\}_{0\leqslant j\leqslant K-1}$ are fixed. In this case, with $d = 2$ and $N = 5$, $y = \mathsf{z}_e^0 \in \mathbb{R}^{2\times 5}$. We choose $K = 8$ and the codebooks $\mathrm{e}_j = \mu_j \in \mathbb{R}^2$, $0 \leqslant j \leqslant K-1$, are fixed centers at regular angular intervals in $\mathbb{R}^2$ and shown in Figure 5.2; the latent states $(\mathsf{z}^t)_{1\leq t\leq T}$ lie in $\{\mathrm{e}_0, \ldots, \mathrm{e}_7\}^5$. Data generation proceeds as follows. First, sample a sequence of $(q_1, \ldots, q_5)$ in $\{0, \ldots, 7\}$ : $q_1$ has a uniform distribution, and, for $s \in \{0, 1, 2, 3\}$, $q_{s+1} = q_s + b_s \mod 8$, where $b_s$ are independent Bernoulli samples with parameter $1/2$ taking values in $\{-1, 1\}$. Conditionally on $(q_1, \ldots, q_5)$, $y$ is a Gaussian random vector with mean $(\mathrm{e}_{q_1}, \ldots, \mathrm{e}_{q_5})$ and variance $\mathbf{I}_{2\times 5}$.

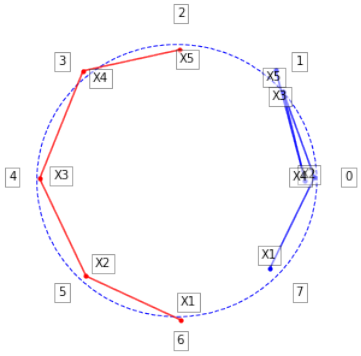



FIGURE 5.2 – Toy dataset, with $K = 8$ centroids, and two samples $y = (y_1, y_2, y_3, y_4, y_5)$ in $\mathbb{R}^{2\times 5}$ each displayed as $5$ points in $\mathbb{R}^2$ (blue and red points), corresponding to the discrete sequences (red) $(6, 5, 4, 3, 2)$ and (blue) $(7, 0, 1, 0, 1)$.

We train our bridge procedure with $T = 50$ timesteps, $\vartheta = 2, \eta = 0.1$, other architecture details and the neural network $\varepsilon_\theta(\mathsf{z}_e^t, t)$ are described in Appendix A.2.5. Forward noise process and denoising using $\varepsilon_\theta(\mathsf{z}_e^t, t)$ are showcased in Figure 5.3, and more illustrations and experiments can be found in Appendix A.2.5.

**End-to-end training.** Contrary to VQ-VAE procedures in which the encoder/decoder/codebooks are trained separately from the prior, we can train the bridge prior alongside the codebooks. Consider a new setup, in which the $K = 8$ codebooks are randomly initialized and considered as parameters of our model (they are no longer fixed to the centers of the data generation process $\mu_j$). The first part of our loss function, in conjunction with the Gumbel-Softmax trick makes it possible to train all the parameters of the model end-to-end. Details of the procedure and results are shown in Appendix A.2.5.

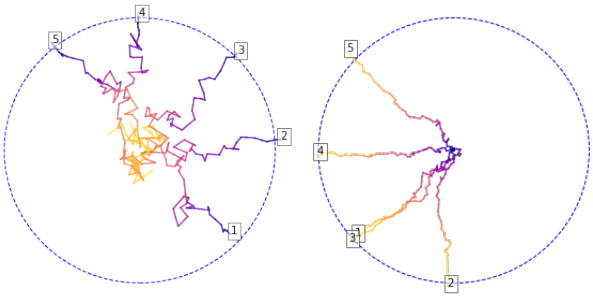


FIGURE 5.3 – (Left) Forward noise process for one sample. First, one data is drawn ($\mathsf{z}_e^0(y) = y$ in the toy example) and then $\{\mathsf{z}_e^t\}_{1\leq t\leq T}$ are sampled under $q_\phi$ and displayed. (Right) Reverse process for one sample $\mathsf{z}_e^T \sim \mathcal{N}(0, (2\vartheta)^{-1}\eta^2\mathbf{I}_{dN})$. As expected, the last sample $\mathsf{z}_e^0$ reaches the neighborhood of $5$ codebooks.

### 5.4.2 Image Synthesis

#### Protocol

In this section, we focus on image synthesis using CIFAR10 and miniImageNet datasets. The goal is to evaluate the efficiency and properties of our model compared to the original PixelCNN. Note that for fair comparisons, the encoder, decoder and codebooks are pretrained and fixed for all models, only the prior is trained and evaluated here. As our goal is the comparison of priors, we did not focus on building the most efficient VQ-VAE, but rather a reasonable model in terms of size and efficiency.

**CIFAR10.** The CIFAR dataset consists of inputs $y$ of dimensions $32\times32$ with 3 channels. The encoder projects the input into a grid of continuous values $\mathsf{z}_e^0$ of dimension $8\times8\times128$. After discretisation, $\{\mathsf{z}^t\}_{0\leqslant t\leqslant T}$ are in a discrete latent space induced by the VQ-VAE which consists of values in $\{1,\dots,K\}^{8\times8}$ with $K = 256$. The pre-trained VQ-VAE reconstructions can be seen in Figure A.7 in Appendix A.2.6.

**miniImageNet.** *mini*ImageNet was introduced by Vinyals et al. (2016) to offer more complexity than CIFAR10, while still fitting in memory of modern machines. 600 images were sampled for 100 different classes from the original ImageNet dataset, then scaled down, to obtain 60,000 images of dimension $84 \times 84$. In our experiments, we trained a VQVAE model to project those input images into a grid of continuous values $\mathsf{z}_e^0$ of dimensions $21 \times 21 \times 32$, see Figure A.9 in Appendix A.2.6. The associated codebook contains $K = 128$ vectors of dimension $32$.

**Prior models.** Once the VQ-VAE is trained on the miniImageNet and CIFAR datasets, the $84\times84\times3$ and $32\times32\times3$ images respectively are passed to the encoder and result in $21 \times 21$ and $8 \times 8$ feature maps respectively. From this model, we extract the discrete latent states from training samples to train a PixelCNN prior and the continuous latent states for our diffusion. Concerning our diffusion prior, we choose the Ornstein-Uhlenbeck process setting $\eta = \sqrt{2}$, $z_* = 0$ and $\vartheta = 1$, with $T = 1000$.

**End-to-End Training.** As an additional experiment, we propose an End-to-End training of the VQ-VAE and the diffusion process. To speed up training, we first start by pretraining the VQ-VAE, then learn the parameters of our diffusion prior alongside all the VQ-VAE parameters (encoder, decoder and codebooks). Note that in this setup, we cannot directly compare the NLL to PixelCNN or our previous diffusion model as the VQ-VAE has changed, but we can compare image generation metrics such as FID and sample quality.

### Quantitative results

We benchmarked our model using three metrics, in order to highlight the performance of the proposed prior, the quality of produced samples as well as the associated computation costs. Results are given as a comparison to the original PixelCNN prior for both the *mini*ImageNet (see Table 5.2) and the CIFAR10 (see Table 5.3) datasets.

**Negative Log Likelihood.** Unlike most related papers, we are interested in computing the Negative Log Likelihood (NLL) directly in the latent space, as to evaluate the capacity of the priors to generate coherent latent maps. To this end, we mask a patch of the original latent space, and reconstruct the missing part, similar to image inpainting, following for instance Van Oord et al. (2016). In the case of our prior, for each sample $y$, we mask an area of the continuous latent state $\mathsf{z}_e^0$, i.e. we mask some components of $\mathsf{z}_e^0$, and aim at sampling the missing components given the observed ones using the prior model. Let $\underline{\mathsf{z}}^0$ and $\underline{\mathsf{z}_e}^0$ (resp. $\overline{\mathsf{z}}^0$ and $\overline{\mathsf{z}_e}^0$) be the masked (resp. observed) discrete and continuous latent variables. The target conditional likelihood is

$$\begin{aligned} p_\theta(\underline{\mathsf{z}}^0|\overline{\mathsf{z}_e}^0) &= \int p_\theta(\underline{\mathsf{z}}^0, \underline{\mathsf{z}_e}^0|\overline{\mathsf{z}_e}^0)\mathrm{d}\underline{\mathsf{z}_e}^0 \,, \\ &= \int p_\theta(\underline{\mathsf{z}}^0|\underline{\mathsf{z}_e}^0)p_\theta(\underline{\mathsf{z}_e}^0|\overline{\mathsf{z}_e}^0)\mathrm{d}\underline{\mathsf{z}_e}^0 \,. \end{aligned}$$

This likelihood is intractable and replaced by a simple Monte Carlo estimate $\hat{p}_\theta(\underline{\mathsf{z}}^0|\underline{\mathsf{z}_e}^0)$ where $\underline{\mathsf{z}_e}^0 \sim p_\theta(\underline{\mathsf{z}_e}^0|\overline{\mathsf{z}_e}^0)$. Note that conditionally on $\underline{\mathsf{z}_e}^0$ the components of $\underline{\mathsf{z}}^0$ are assumed to be independent but $\underline{\mathsf{z}_e}^0$ are sampled jointly under $p_\theta(\underline{\mathsf{z}_e}^0|\overline{\mathsf{z}_e}^0)$. As there are no continuous latent data in PixelCNN, $p_\theta(\underline{\mathsf{z}}^0|\overline{\mathsf{z}}^0)$ can be directly evaluated.

**Fréchet Inception Distance.** We report Fréchet Inception Distance (FID) scores by sampling a latent discrete state $\mathsf{z} \in \mathcal{E}^N$ from the prior, and computing the associated image through the VQ-VAE decoder. In order to evaluate each prior independently from the encoder and decoder networks, these samples are compared to VQ-VAE reconstructions of the dataset images.

**Kullback-Leibler divergence.** In this experiment, we draw $M = 1000$ samples from test set and encode them using the trained VQ-VAE, and then draw as many samples from the pixelCNN prior, and our diffusion prior. We propose then to compute the empirical Kullback Leibler (KL) divergence between original and sampled distribution at each pixel. Figure 5.4 highlights that PixelCNN performs poorly on the latest pixels (at the bottom) while our method remains consistent. This is explained by our denoising process in the continuous space which uses all pixels jointly while PixelCNN is based on an autoregressive model.

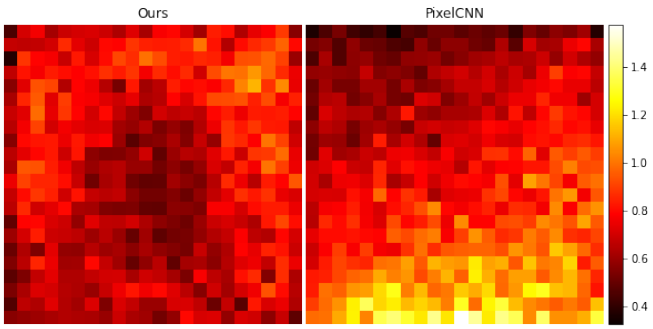



FIGURE 5.4 – KL Distance between the true empirical distribution and both prior distributions in the latent space. Darker squares indicates lower (better) values.

| | KL |
|---|---|
| Ours | **0.713** |
| PixelCNN | 0.809 |

TABLE 5.1 – Averaged KL metric on the feature map.

**Computation times.** We evaluated the computation cost of sampling a batch of 32 images, on a GTX TITAN Xp GPU card. Note that the computational bottleneck of our model consists of the $T = 1000$ sequential diffusion steps (rather than the encoder/decoder which are very fast in comparison). Therefore, a diffusion speeding technique such as the one described in Song et al. (2021) would be straightforward to apply and would likely provide a $\times 50$ speedup as mentioned in the paper.

TABLE 5.2 – Results on *mini*ImageNet. Metrics are computed on the validation dataset. The means are displayed along with the standard deviation in parenthesis.

| | NLL | FID | s/sample |
|---|---|---|---|
| PixelCNN Oord et al. (2017) | 1.00 ($\pm 0.05$) | 98 | 10.6s ($\pm 28ms$) |
| Ours | 0.94 ($\pm 0.02$) | 99 | 1.7s ($\pm 10ms$) |

TABLE 5.3 – Results on CIFAR10. Metrics are computed on the validation dataset. The means are displayed along with the standard deviation in parenthesis. NLL for end-to-end takes into account the full model including the modified VQ-VAE, and therefore is not directly comparable to the two others.

| | NLL | FID | s/sample |
|---|---|---|---|
| PixelCNN Oord et al. (2017) | 1.41 ($\pm 0.06$) | 109 | 0.21 ($\pm 0.8ms$) |
| Ours | 1.33 ($\pm 0.18$) | 104 | 0.05s ($\pm 0.5ms$) |
| Ours (end-to-end) | 1.59 ($\pm 0.27$) | 92 | 0.11s ($\pm 0.5ms$) |

**Qualitative results**

**Sampling from the prior.** Samples from the PixelCNN prior are shown in Figure 5.5b and samples from our prior in Figure 5.5a. Additional samples are given in Appendix A.2.6. Note that contrary to original VQ-VAE prior, the prior is not conditioned on a class, which makes the generation less specific and more difficult. However, the produced samples illustrate that our prior can generate a wide variety of images which show a large-scale spatial coherence in comparison with samples from PixelCNN.

**Conditional sampling.** As explained in Section 5.4.2, for each sample $y$, we mask some components of $\mathbf{z}_e^0(y)$, and aim at sampling the missing components given the observed ones using the prior models. This conditional denoising process is further explained for our model in Appendix A.2.2. To illustrate this setting, we show different conditional samples for 3 images in Figure 5.8 and Figure 5.9 for both the PixelCNN prior and ours. In Figure 5.8, the mask corresponds to a $9 \times 9$ centered square over the $21 \times 21$ feature map. In Figure 5.9, the mask corresponds to a $9 \times 9$ top left square. These figures illustrate that our diffusion model is much less sensitive to the selected masked region than PixelCNN. This may be explained by the use of our denoising function $\varepsilon_\theta$ which depends on all conditioning pixels while PixelCNN uses a hierarchy of masked convolutions to enforce a specific conditioning order. Additional conditional sampling experiments are given in Appendix A.2.6.

(a) Samples from our diffusion prior.

(b) Samples from the PixelCNN prior.

FIGURE 5.5 – Comparison between samples from our diffusion-based prior (top) and PixelCNN prior (bottom).

**Denoising chain.** In addition to the conditional samples, Figure 5.6 shows the conditional denoising process at regularly spaced intervals, and Figure 5.7 shows unconditional denoising. Each image of the chain is generated by passing the predicted $\mathsf{z}^t$ through the VQ-VAE decoder.

### 5.4.3 Relative humidity forecasting

After asserting the quality of the proposed prior model on the traditional image datasets, compared to state of the art approaches, we present results applied on the Relative Humidity dataset.

First, notice that the discrete latent model presented in the previous chapter, see Section 4.5.3, can be expressed in the context of our new proposed framework. We set $N$ as the length of the time series samples ($24$ in the case of our Relative Humidity dataset), and the number of diffusion step $T = 0$. In this context, the lattice of latent quantized vector is one dimensional. Then, we replace our initial autoregressive, Markovian prior with a diffusion

FIGURE 5.6 – Sampling denoising chain from $t = 500$ up to $t = 0$, shown at regular intervals, conditioned on the outer part of the picture. We show only the last $500$ steps of this process, as the first $500$ steps are not visually informative. The sampling procedure is described in Appendix A.2.2.

bridge. Similarly to our previous experiments, this diffusion bridge samples latent vectors conditionally on a set of commands, in order to produce forecasts matching the current state of the building as well as the outside weather. We choose $T = 100$, and optimize the Evidence Lower BOund defined in 5.2, with the encoder function $f_\phi$ proposed in Section 4.5.3.

As presented in Table 5.4, the performance of this new model are encouraging when compared to our benchmark, in particular to the previous discrete latent model approach. We compared the confidence intervals produced by both methods, results are displayed for two samples of the validation set in Figure 5.10. There still remains multiple limitations, in particular the much higher computation time required, which is not unusual for such a novel approach. Methods for reducing the sampling time have already been proposed in the literature, see Song et al. (2021) for instance. In addition, tuning the parameters of the diffusion bridges has been much more challenging than for the previous Markovian priors. In Section 5.5, we discuss an extension of diffusion bridges applied to dicrete data.

TABLE 5.4 – **Relative Humidity dataset.** Comparison of RMSE, MAE and computation time of our model against the previously proposed discrete latent architecture (see Section 4.5.3), the benchmarked VQ-VAE, HMM, as well as our decoupled architecture (see Section 4.4.2). We report performance on par with these benchmarked models. This table also highlights one major limitation of this new approach : the high computation time. We believe that further works toward diffusion bridges will eventually address this problem, as already seen in Song et al. (2021). This table provides aggregated results of the predictions on the entire validation set. Mean values of the estimators, taken over the validation samples of the dataset, are displayed along with their variance.

| | RMSE | MAE | Computation time |
|---|---|---|---|
| HMM | $0.46 \pm 0.19$ | $0.40 \pm 0.17$ | $99ms$ |
| VQ-VAE | $0.55 \pm 0.33$ | $0.51 \pm 0.32$ | $39ms$ |
| SMCL | $0.30 \pm 0.19$ | $0.26 \pm 0.16$ | $21ms$ |
| Vaditlis (gru) | $0.30 \pm 0.19$ | $0.24 \pm 0.15$ | $39ms$ |
| Diffusion bridges | $0.29 \pm 0.19$ | $0.24 \pm 0.16$ | $2,159ms$ |

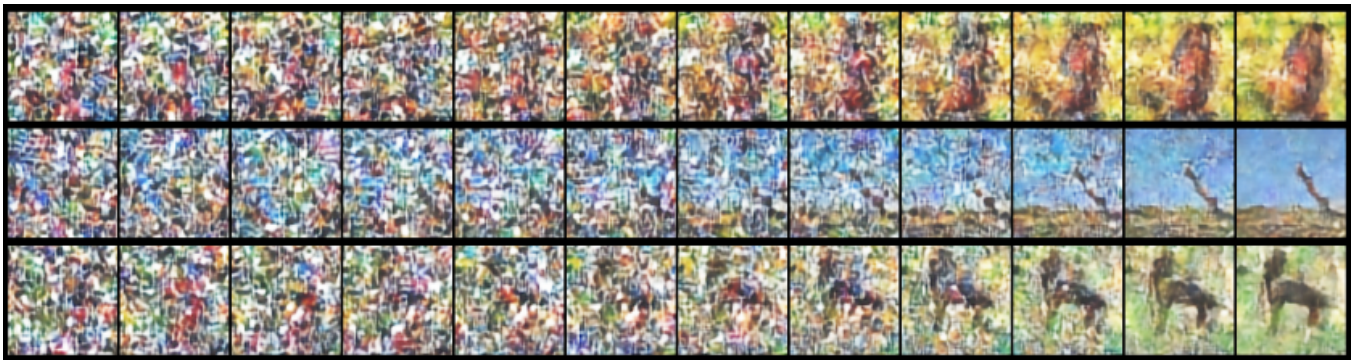


FIGURE 5.7 – Sampling denoising chain from $t = 500$ up to $t = 0$, shown at regular intervals, unconditional. We show only the last $500$ steps of this process, as the first $500$ steps are not visually informative. The sampling procedure is described in Algorithm 3

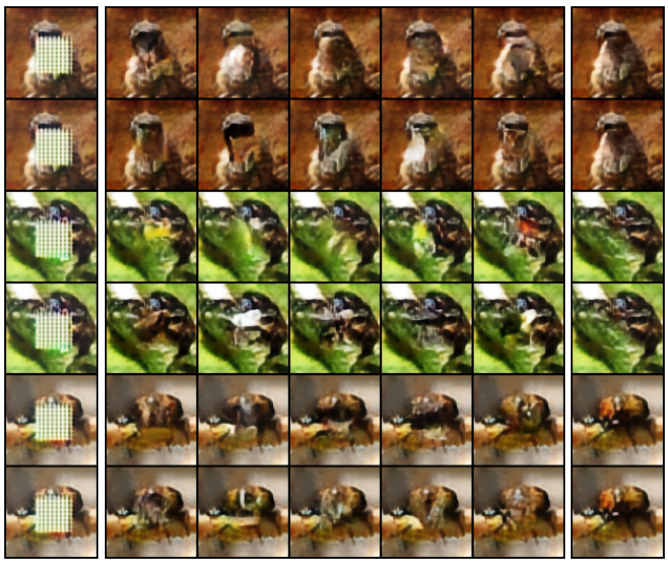


FIGURE 5.8 – Conditional sampling with centered mask : for each of the 3 different images, samples from our diffusion are on top and from PixelCNN on the bottom. For each row : the image on the left is the VQVAE masked reconstruction, the image on the right is the full VQ-VAE reconstruction. Images in-between are independent conditional samples from the models.

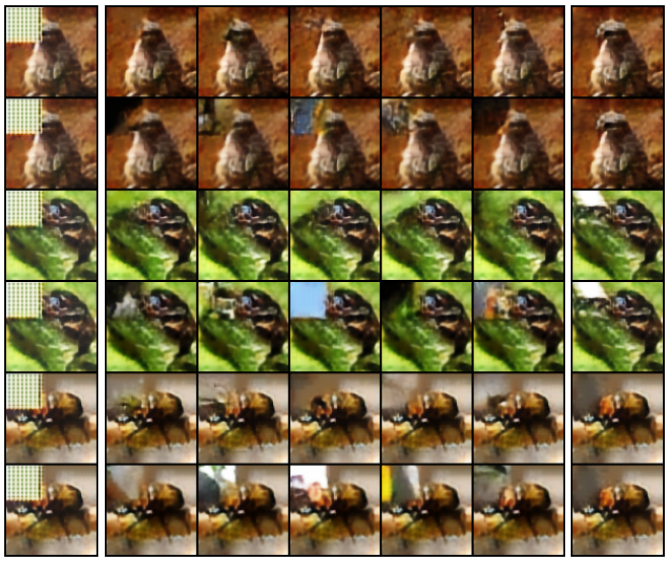
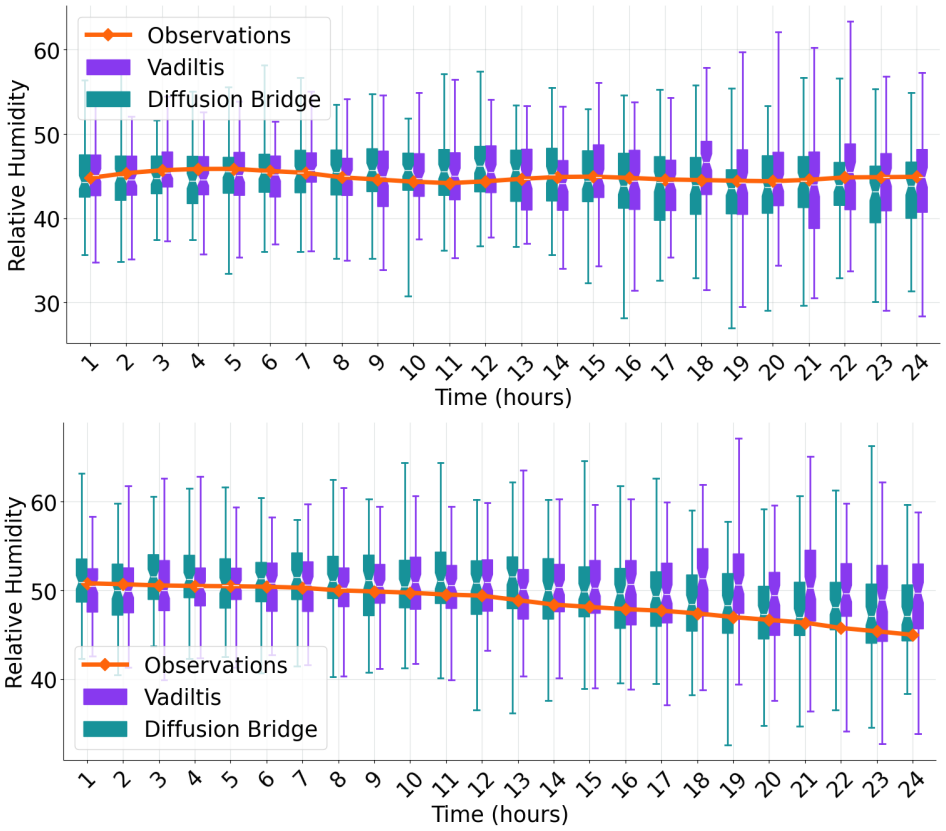


FIGURE 5.9 – Conditional sampling with top left mask : for each of the 3 different images, samples from our diffusion are on top and from PixelCNN on the bottom. For each row : the image on the left is the VQVAE masked reconstruction, the image on the right is the full VQ-VAE reconstruction. Images in-between are independent conditional samples from the models.

FIGURE 5.10 – Prediction of Relative Humidity on two samples from the validation dataset. Each box contains 75% of samples, while the whiskers cover 95%. The produced confidence intervals are coherent with previous approaches.



## 5.5 Extension of discrete diffusion

In this section, we propose an extension to probabilistic diffusion models applied to discrete latent states. We consider a set of discrete observations $\mathsf{z} \in \mathcal{E}^N$, where $\mathcal{E} = \{\mathrm{e}_1, \cdots, \mathrm{e}_K\}$, with $\mathrm{e}_k \in \mathbb{R}^d$ for all $1 \leq k \leq K$. For the sake of visualization, and in order to compare diffusion methods directly on a discrete example, we propose a simple toy dataset of 5 letter words, whereby $K = 26$, $N = 5$. Only a few combinations of 5 letters $(\mathsf{z}_k)_{k=1}^5$ result in an existing word. Once presented, we detail the results of our extended method on our benchmark dataset, the relative humidity forecasting task.

### 5.5.1 Motivations

Diffusion bridges applied directly on discrete random variables have already been proposed, such as the multinomial diffusion Hoogeboom et al. (2021), where the authors work within the Categorical distribution space, modeling the probabilities in a $K \times N$ space. Their forward noise process consists of sampling a random $i \in \{1, \ldots, N\}$ and sampling $\mathsf{z}^i$ from a uniform distribution. This however, seems to suffer from limitations, one of which being the inability to scale with large vocabularies $K$, which is the case in language models ($K \sim 50,000$) or VQ-VAEs ($K \sim 4,000$ or more).

We postulate that using a continuous diffusion in an embedded space $\mathsf{X} \subset \mathbb{R}^{d \times N}$is more appropriate as 1) it leverages recent advances in continuous diffusions and 2) it enables to scale better, as $d$ can remain small compared to $K$, typically in word embeddings $d$ evolves as $\log K$. This embedding space can be given, created or trained. We will focus on a simple application, where $e_k \sim \mathcal{N}(0, \mathbf{I}_d), \forall 1 \leq k \leq K$ are sampled independently and fixed during the process.

Our first approach was to simply embed the discrete data samples $\mathsf{z}$ into the continuous space $\mathsf{X}$ using the embedding projections $e_k, k \in \{1, \ldots, K\}$. Then, within the continuous space $\mathsf{X}$ we would be able to perform standard continuous denoising diffusion Ho et al. (2020). In that case, the diffusion process in defined by choosing a non-informative prior distribution, building a forward noising process gradually corrupting the samples in order to reach the prior distribution, and training a denoising model. In practice, the prior distribution is often a unit Gaussian, the corrupting process a Ornstein-Uhlenbeck and the denoising model a deep neural network. Once trained, the sampling process consists in drawing a sample $\mathsf{z}_e^T$ under the prior distribution, computing $T$ denoising steps $p_{\theta, t|t+1}(\mathsf{z}_e^t | \mathsf{z}_e^{t+1})$ for $t \in \{T-1, \cdots, 0\}$, and the associated quantized vector $\mathsf{z} \sim p_\theta(\mathsf{z}^0 | \mathsf{z}_e^0)$.

Instead of defining the discrete distribution $p_\theta(\mathsf{z}|\mathsf{z}_e)$ as a Dirac on the nearest neighbor embedding, $\delta_{e_{k^\star}}(\mathsf{z})$, where $k^\star = \texttt{argmin}_{1 \leq k \leq K} \{\|\mathsf{z}_e - \mathrm{e}_k\}$, we sampled a codebook from the following Categorical distribution :

$$p_\theta(\mathsf{z} = \mathrm{e}_k | \mathsf{z}_e) = \texttt{softmax}_{1 \leq j \leq K} \{-\|\mathsf{z}_e - \mathrm{e}_j\|_2\}_k \,,$$

for all $1 \leq k \leq K$.

### 5.5.2 Limitations

Whenever we experiment our proposed continuous diffusion on a high dimensional latent space, we notice a scaling issue. As the corrupting process adds noise independently and identically on all dimensions, the final sample of the noising chain $\mathsf{z}_e^T$ may be located quite far from the codebooks. Even in the context of our 5-letter word toy example, our diffusion leads to geometric problems, as shown in Figure 5.11. There, we plotted three denoising trajectories sampled after training a simple diffusion bridge.

Because the training procedure will direct the denoising process towards the codebooks, this limitation may be mitigated after reaching a good estimate of the models parameters. However, this implies long and complex

trainings, and could impact the final performance of the diffusion bridge.

We now investigate a more sophisticated corrupting process, which aims at directing the noising towards the codebooks in the latent space. As we present in the next section, such an improvement can be incorporated directly in our mathematical framework.

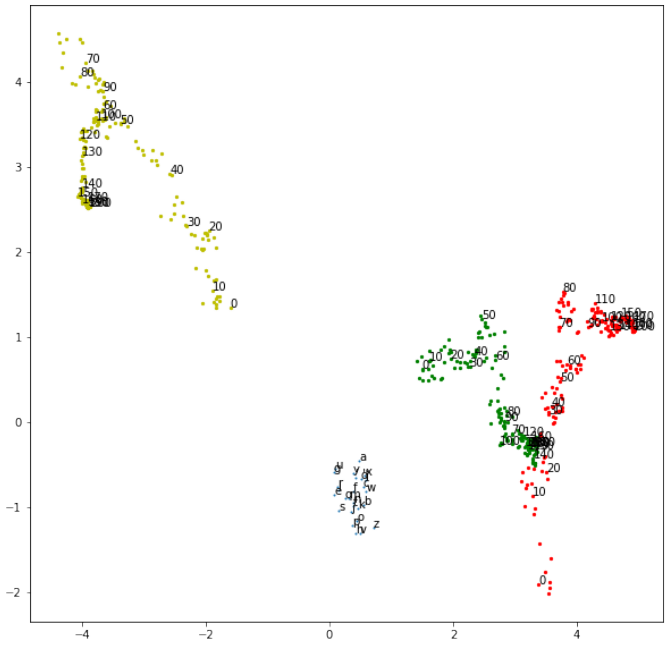


FIGURE 5.11 – PCA projection of embeddings of each letter and denoising diffusion trajectories. Because of a scaling problem, the starting point of the denoising diffusion $\mathsf{z}_e^T$ is very far from the codebooks $(e_k)_{k=1}^K$, and the denoising process stay very far from the embeddings, resulting in exploration problems (not all embeddings are reachable, therefore the model will have trouble predicting some of them).

### 5.5.3 Embedding-guided denoising

Let $\mu$ be the empirical mean vector of the codebooks, and $M$ their covariance matrix :

$$\mu = \frac{1}{K}\sum_{j=1}^{K} \mathrm{e}_k\,, \quad M = \frac{1}{K}\sum_{j=1}^{K} (\mathrm{e}_k - \mu)\,(\mathrm{e}_k - \mu)^\top\,.$$

Consider that the noising process boils down to sampling solution to the following Stochastic Differential Equation (SDE) in $\mathbb{R}^d$ :

$$\mathrm{d}Z_t = \vartheta(Z_t - \mu)\mathrm{d}t + \eta M^{1/2}\mathrm{d}W_t\,, \tag{5.7}$$

We can define the variational density by integrating this SDE along small step-sizes. In this setting, $q^{\mathsf{z}_e}_{\phi,t|t-1}(\mathsf{z}_e^t|\mathsf{z}_e^{t-1})$ is a Gaussian probability density function with mean vector $\mu + (\mathsf{z}_e^{t-1} - \mu)\mathrm{e}^{-\vartheta\delta_t}$ and covariance matrix $(2\vartheta)^{-1}\eta^2(1 - \mathrm{e}^{-2\vartheta\delta_t})M$. The denoising process amounts then to sampling from the law of $\mathsf{z}_e^{t-1}$ given $\mathsf{z}_e^0$ and $\mathsf{z}_e^t$, a Gaussian probability density function with mean :

$$\tilde{\mu}_{\phi,t-1|0,t}(\mathsf{z}_e^0, \mathsf{z}_e^t) = \frac{\beta_t}{1-\bar{\alpha}_t}\{\sqrt{\bar{\alpha}_{t-1}}\mathsf{z}_e^0 + (1 - \sqrt{\bar{\alpha}_{t-1}})\mu\} + \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\sqrt{\alpha_t}\{\mathsf{z}_e^t - (1 - \sqrt{\alpha_t})\mu\}$$

and covariance matrix

$$\tilde{\sigma}^2_{\phi,t-1|0,t} = \frac{1}{2\vartheta}\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\, M\,,$$

with the same notation $\alpha$, $\bar{\alpha}$ and $\beta$ as introduced in Section 5.3.2. Note that under $q_\phi$, $\mathsf{z}_e^t$ has the same distribution as $\mathsf{h}_e^t(\mathsf{z}_e^0, \varepsilon_t) = \sqrt{\bar{\alpha}_t}\mathsf{z}_e^0 + (1-\sqrt{\bar{\alpha}_t})\mu + \sqrt{\frac{1}{2\vartheta}(1-\bar{\alpha}_t)}M^{1/2}\varepsilon_t$, where $\varepsilon_t \sim \mathcal{N}(0, \mathbf{I})$. Then, $\tilde{\mu}_{\phi,t-1|0,t}$ can be reparameterised as follows :

$$\tilde{\mu}_{\phi,t-1|0,t}(\mathsf{z}_e^0, \varepsilon_t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathsf{h}_e^t(\mathsf{z}_e^0, \varepsilon_t) - \sqrt{\frac{1}{2\vartheta(1-\bar{\alpha}_t)}}\beta_t M^{1/2}\varepsilon_t\right).$$

We therefore propose to use

$$\mu_{\theta,t-1|t}(\mathsf{z}_e^t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathsf{z}_e^t - \sqrt{\frac{1}{2\vartheta(1-\bar{\alpha}_t)}}\beta_t M^{1/2}\varepsilon_\theta(\mathsf{z}_e^t, t)\right).$$

We demonstrate the impact of this extension in Figure 5.12, and detail its performance on the Relative Humidity dataset in the next chapter.

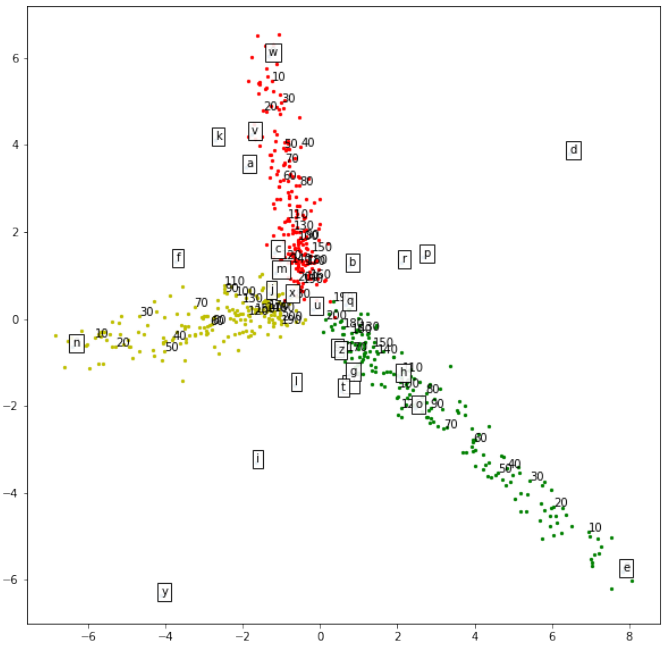


FIGURE 5.12 – PCA projection of embeddings of each letter and guided denoising diffusion trajectories.

### 5.5.4 Results on the Relative Humidity dataset

In order to benchmark this extension, we run the same experiments as presented in Section 5.4.3, where we train a diffusion bridge prior model on the relative humidity forecasting task. Although we could not obtain any noticeable improvement compared to the initially proposed diffusion bridge, regarding the RMSE criteria, our model now requires much less training epochs in order to converge (5 epochs for the embedded guided diffusion against over 50 for the previous diffusion). As an illustration, we plotted the evolution of the ELBO, as well as the RMSE criteria, over the training epochs for the simple diffusion (as presented in the previous Section) compared to the Extended guided diffusion, see Figure 5.13. These results comfort our intuition that the scaling issue, arising from adding noise independently and identically in all dimensions of the latent space, makes the training procedure

much more complex. Although this limitation may be mitigated during the training procedure, the embedded guided diffusion offer an efficient alternative to the corrupting and reconstruction process, without suffering any additional computational cost or implementation complexity.

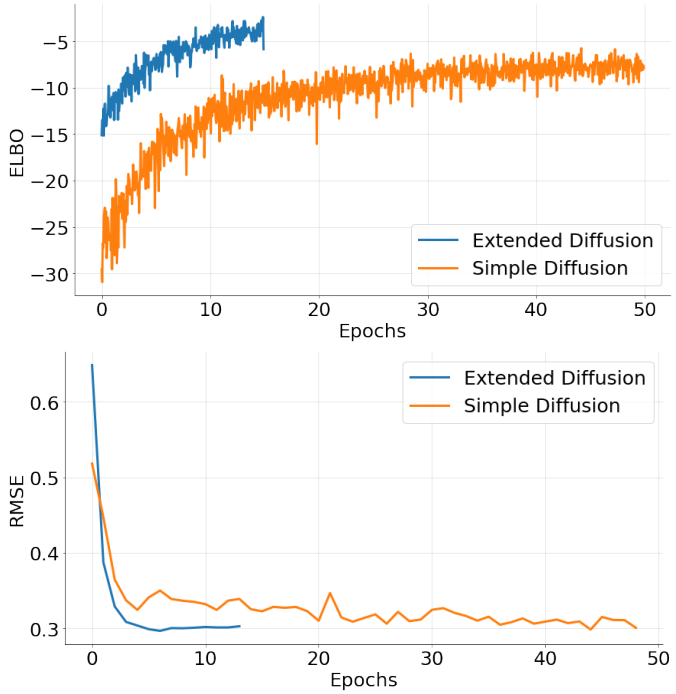



FIGURE 5.13 – Comparison of the evolution of the ELBO, as well as the RMSE criteria, over the training epochs of the simple diffusion presented in the previous Section, compared to the Extended guided diffusion. The latter version is able to converge much more quickly, although the resulting RMSE in not significantly higher.

## 5.6 Conclusion

In the previous chapter, we showed that discrete latent models offer performance on par with their continuous counterparts, while allowing simpler training procedures. In this chapter, we aimed at exploring more diverse and complex architectures.

Recent discrete latent models found in the literature are often benchmarked on computer vision tasks, such as image synthesis or inpainting. In our exploration, we found that these models rely on very complex, hard to train, autoregressive priors, presenting two main limitations : the autoregressive nature of the prior models do not always match the latent space structure, and estimating their parameters require a subsequent training procedure. In order to address these shortcomings, and to compare recent models on a theoretically grounded benchmark, we proposed a new mathematical framework for discrete latent models. By proposing a novel method for modelling the discrete latent space, we allow for more generic prior models, and bypass the need for a two step training.

Modelling the discrete distribution of the latent space is a complex task, for which no satisfactory method exists yet. We proposed instead to map discrete vectors to a continuous space, through a set of learned embeddings. This allows us to leverage known modelling tools, such as such as diffusion bridges, and scales much better to high dimension spaces than purely discrete alternatives. Our framework allows for any law between the continuous and discrete spaces. We hope that it will serve as a sound and stable foundation to derive future generative models.

We then demonstrated the performance of our framework on an image synthesis, and an inpainting task, where our proposed model is able to generate complex, coherent samples, competitive with state of the art methods. We

also successfully applied our proposed methodology on the Relative Humidity dataset. We believe that these first numerical experiments open up many research avenues, such as scaling to larger models, optimal scaling of the hyperparameters, alternative diffusion methods, or studying the influence of the regularization loss for end-to-end training.

In a last section, we extended our initial diffusion bridge to improve the noising and denoising processes in the context of embedding vectors, such as the codebooks. This allowed to improve training performance, by greatly reducing the number of epochs before convergence of the model. It also brings forth a new approach to apply diffusion bridges to inherent discrete tasks, such as Natural Language Processing.

# Bibliographie


Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*.

Beskos, A., Roberts, G., Stuart, A., and Voss, J. (2008). Mcmc methods for diffusion bridges. *Stochastics and Dynamics*, 8(03) :319–350.

Bladt, M., Finch, S., and Sørensen, M. (2016). Simulation of multivariate diffusion bridges. *Journal of the Royal Statistical Society : Series B : Statistical Methodology*, pages 343–369.

Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. (2018). Pixelsnail : An improved autoregressive generative model. In *International Conference on Machine Learning*, pages 864–872. PMLR.

De Bortoli, V., Doucet, A., Heng, J., and Thornton, J. (2021). Simulating diffusion bridges with score matching. *arXiv preprint arXiv :2111.07243*.

Esser, P., Rombach, R., and Ommer, B. (2021). Taming transformers for high-resolution image synthesis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883.

Gu, S., Chen, D., Bao, J., Wen, F., Zhang, B., Chen, D., Yuan, L., and Guo, B. (2021). Vector quantized diffusion model for text-to-image synthesis. *arXiv preprint*.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS 2021)*, 34.

Hoogeboom, E., Nielsen, D., Jaini, P., Forr é, P., and Welling, M. (2021). Argmax flows and multinomial diffusion : Learning categorical distributions. *Advances in Neural Information Processing Systems (NeurIPS 2021)*, 34.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. (2021). Normalizing flows : An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11) :3964–3979.

Lin, M., Chen, R., and Mykland, P. (2010). On generating monte carlo samples of continuous diffusion bridges. *Journal of the American Statistical Association*, 105(490) :820–838.

Mittal, G., Engel, J., Hawthorne, C., and Simon, I. (2021). Symbolic music generation with diffusion models. *arXiv preprint arXiv :2103.16091*.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet : A generative model for raw audio. *arXiv preprint arXiv :1609.03499*.

Oord, A. v. d., Vinyals, O., and Kavukcuoglu, K. (2017). Neural discrete representation learning. *Advances in neural information processing systems (NeurIPS 2017)*.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. 139 :8821–8831.

Razavi, A., van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *Advances in neural information processing systems (NeurIPS 2019)*, pages 14866–14876.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). Pixelcnn++ : Improving the pixelcnn with discretized logistic mixture likelihood and other modifications.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. 37 :2256–2265.

Song, J., Meng, C., and Ermon, S. (2021). Denoising diffusion implicit models.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. 32.

Vahdat, A., Kreis, K., and Kautz, J. (2021). Score-based generative modeling in latent space.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016). Conditional image generation with pixelcnn decoders.

Van Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR.

Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., and Wierstra, D. (2016). Matching networks for one shot learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Willetts, M., Miscouridou, X., Roberts, S., and Holmes, C. (2021). Relaxed-responsibility hierarchical discrete VAEs. *ArXiv :2007.07307*.

# Chapitre 6

# Conclusion

## 6.1 Thesis summary

In this thesis, we develop deep learning architectures for modelling building energy consumption and air quality. Using historic data, we propose to optimize energy demand, while improving indoor comfort and well being. Because modelling the behavior of a building is a complex task, where many of the relevant factors are unknown, we quantify the uncertainty associated with each prediction of our models.

When simulating buildings behaviors, we show in Chapter 3 that we can replace numerical simulators such as TRNSYS with statistical surrogate models. Once trained, they offer much faster computation times, which allows us to run costly optimization tasks in a reasonable time frame. We present an application on two real buildings handled by Oze-Energies. The unknown parameters of the building, such as physical properties or HVAC usages, are estimated by comparing the output of the model with the historic data gathered. Then, energy loads are optimized by evaluating simulated scenarios from the metamodel, and selecting a set of HVAC settings leading to a reduced consumption for an equivalent thermic comfort. This experiment demonstrates that results are coherent between the metamodel and the original physical simulator TRNSYS, throughout the calibration and optimization tasks. Additionally, the speed of the metamodel, and its ability to parallelize dozen of simulations, allows us to calibrate buildings that would have taken days using TRNSYS.

After demonstrating the performance of statistical models to approximate complex functions, we explore methods for quantifying their uncertainty. In a context where input variables may be noisy or unavailable, predicting a single point estimate cannot reflect the potential uncertainty ; instead, we model the distribution of the observations, from which we can quantify the uncertainty. Well known statistical models, such as Hidden Markov Model, fit this description, however their training procedure usually do not scale well to high number of parameters. In Chapter 4, we propose two approaches for combining the approximation capabilities of neural networks, with uncertainty modelling techniques.

We start by modelling noise on the recurrent layers of neural networks, whose parameters are estimated using Sequential Monte Carlo (SMC) methods. As it is sufficient to constraint this modelling to the last layer of the model only, we develop a decoupled deep learning architecture adapted to time series. The parameters of the model are first estimated in an efficient, deterministic gradient descent. Then, the weights of the last layer are finetuned by minimizing the log likelihood associated with a set of weighted particles. On a relative humidity forecasting task, our model produces accurate predictions along with confidence intervals.

In a second approach, we propose a quantized latent model. Recent advances in generative modelling have pushed towards these models, as they can lead to a more meaningful representation of the data, without degrading performance. In our case, we are able to simplify the training procedure, as we no longer rely on the complex

approximation of SMC methods. Furthermore, we show that interpreting the discrete latent states paves the way for new applications, such as unsupervised segmentation for instance. The parameters of the model are estimated by Variational Inference, by maximizing a lower bound on the likelihood. Performance wise, this new discrete based model is on par with its continuous counterpart on the relative humidity forecasting task.

The last part of this thesis focuses on modelling discrete distributions, in the same context of quantized latent states. In Chapter 5, we experiment with several choices of prior models, as they are at the center of the discrete latent generative models literature. State of the art architectures rely on complex autoregressive priors, which require several implementation tricks for their training procedure to converge. We propose a theoretically grounded methodology for modelling the discrete latent state distribution, regardless of the nature of the observations (time series, images, etc.), by using diffusion bridges. The key idea is to iteratively corrupt a complex distribution into a non informative one, by adding noise, then learning the inverse reconstruction process, bridging the gap between both distributions. Our experiments show the relevance of this new methodology on computer vision and time series tasks.

Finally, we propose an improvement of the noising process of the diffusion bridge. Because of the high dimension of the latent vectors space, adding independent noise in all dimension quickly causes scaling and geometric problems. By computing instead a covariance matrix as a function of relevant latent vectors, we are able to better direct the noising process towards a non informative distribution. Our new model is much faster to converge during training, reducing the number of epochs by a factor of 10.

## 6.2 Perspectives

**Metamodelling and energy load optimization.** The end-to-end metamodelling methodology proposed in Chapter 3 was satisfactory on two well chosen buildings, and introduced new improvement perspectives for future works on more complex behavior modelling.

For instance, during the training of the metamodel, we defined the loss as a function of a hyper parameter $\beta$, which we were not able to tune. Regardless, the results of the metamodel were satisfactory, yet exploring how to balance the consumption error criteria with the indoor temperature one may be required to obtain more precise results, especially during calibration. We believe this parameter can only be tuned by discussing with energy managers, as they are the most aware of the importance in the accuracy of each variable. Similarly, when optimizing energy loads, we only considered the total cumulated consumption of a building, without pondering any particular behavior. While implementing more complex objectives is not a limitation, defining which behavior to reward (low instant power consumption, heating during off-peak hours) will require significant reflection with the energy managers.

The metamodel is not only much faster than TRNSYS, it is also more flexible in its usage and manipulation. For instance, we could modify our model to provide uncertainty estimation, using the methods developed in Chapter 4 and Chapter 5. Although it would bring a valuable insight regarding the chosen optimization scenario, it is still unclear how to conjugate uncertainty estimation with the calibration and optimization tasks. We leave this question open for future works. Additionally, the metamodel allows for different input parameters at each time step of the simulation. Instead of using the NSGA-II algorithms to solve our optimization task, we could explore reinforcement learning methods, and assign a reward associated with each hour to hour policy. This would be computationally intensive but reinforcement learning approaches allow to define new user-specific and non differentiable rewards which could benefit from insights from energy managers.

**Sequential Monte Carlo methods.** In order to demonstrate the use cases for SMC methods in our decoupled architecture, we utilised simple smoothing algorithms, known to quickly degenerate when dealing with longer time

series. Many alternative to the Path-space smoother have already been proposed in the literature, that allow to mitigate particle degeneracy with limited additional cost. Implementing such algorithms seems like the natural next step in the process of improving our methodology.

In addition, SMC methods shine in that parameters can be updated online, a use case particularly adapted to Oze-Energies. Modelling the changes in a building during an entire is extremely complex, which is why we usually limit ourselves to short time periods, about a few months. Instead of performing independent trainings for different part of the year, we could update the learnt parameters all along the year, in order to better match local weather and occupation conditions.

**Variational discrete latent models.** In this thesis, we presented simple Variational Inference training procedures, in order to focus on the models and their applications. In the future, it would be interesting to develop parameter estimations with more complex methods from the literature, such as the Importance Weighted VAE which allows to tighten the lower bound on the likelihood, or the $\beta$-VAE which introduced an new hyper parameter for balancing the prior loss term with reconstruction accuracy.

Additionally, one could explore more diverse sampling procedures and prior models. In the presented experiments, we constrained ourselves to quantizing vectors by sampling one of the neighboring codebook. Yet our framework is designed for any discrete law, which opens new perspectives for discrete latent models. Similarly, comparing the performance of more diverse prior models could bring new insights toward their impact on the overall performance, as well as on the modelled distribution of the latent states.

Finally, we believe these architectures could be relevant in semi supervised learning, for instance by first training the generative model unsupervised, then using the available annotations to finetune the prior model. This way, we would be able to leverage important amounts of data, with limited annotation cost.

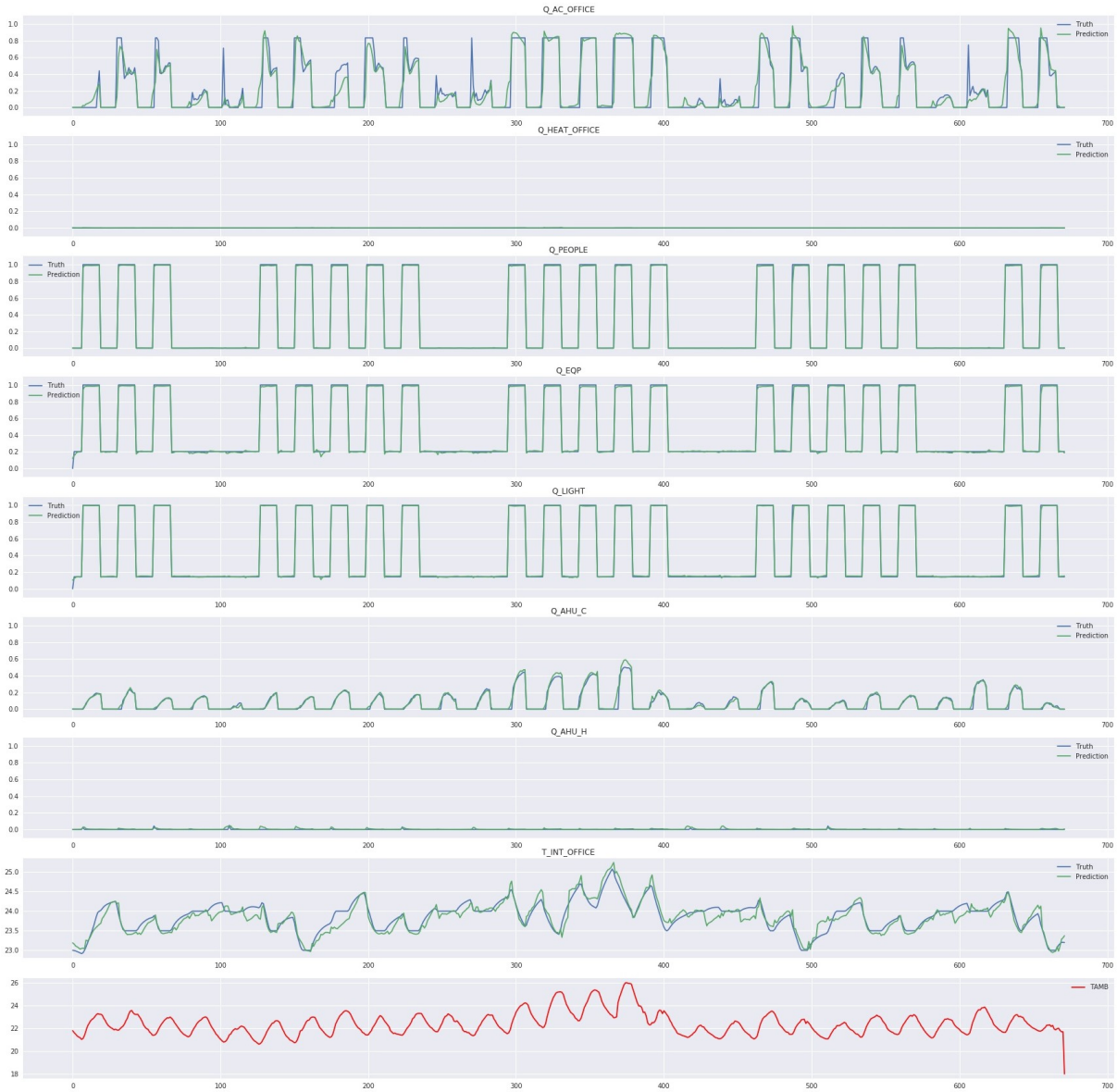
# Annexe A

# Appendix

## A.1 Building management

### A.1.1 Additional illustrations of the metamodel

FIGURE A.1 – Inference from the metamodel compared to the ground truth (TRNSYS simulation), on a sample of the validation dataset.



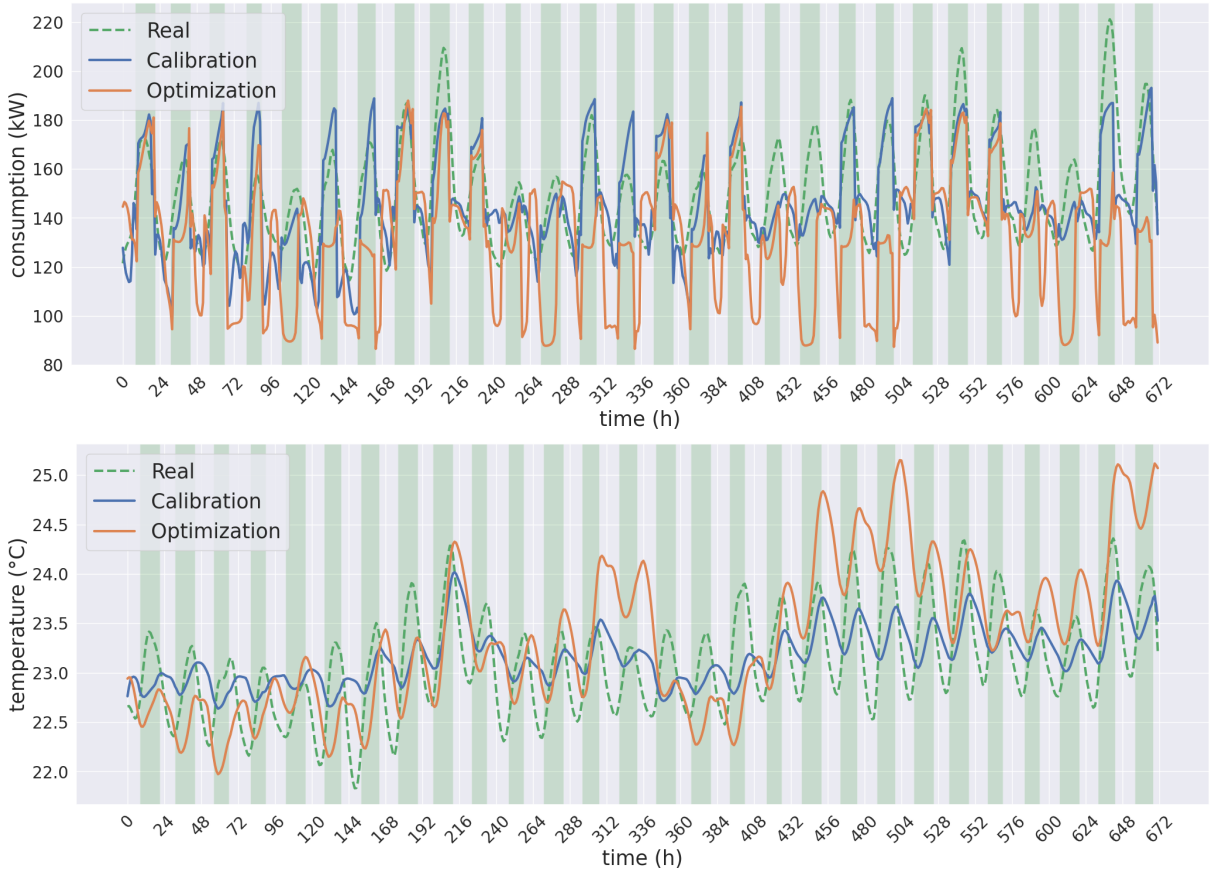



FIGURE A.2 – Indoor temperature and consumption for Real data, Calibration and Optimization for Livingstone.

### A.1.2 Ranges used to train the metamodel

| Variable | Minimum | Maximum | Step |
|---|---|---|---|
| airchange_infiltration_vol_per_h ($\mathrm{m^3 h^{-1}}$) | 0.1 | 0.5 | 0.1 |
| capacitance_kJ_perdegreK_perm3 ($\mathrm{kJK^{-1}m^{-3}}$) | 50 | 300 | 10 |
| power_VCV_kW_heat ($\mathrm{kW}$) | 0 | 1000 | 100 |
| power_VCV_kW_clim ($\mathrm{kW}$) | 0 | 1000 | 100 |
| nb_occupants | 1000 | 2000 | 200 |
| nb_PCs | 1000 | 2000 | 200 |
| percent_light_night | 0 | 70 | 10 |
| percent_PCs_night | 0 | 70 | 10 |
| facade_1_thickness_2 ($\mathrm{m}$) | 0.05 | 0.15 | 0.05 |
| facade_2_thickness_2 ($\mathrm{m}$) | 0.05 | 0.15 | 0.05 |
| facade_3_thickness_2 ($\mathrm{m}$) | 0.05 | 0.15 | 0.05 |
| facade_4_thickness_2 ($\mathrm{m}$) | 0.05 | 0.15 | 0.05 |
| roof_1_thickness_3 ($\mathrm{m}$) | 0.05 | 0.15 | 0.05 |
| facade_1_window_area_percent | 40 | 50 | 5 |
| facade_2_window_area_percent | 40 | 50 | 5 |
| facade_3_window_area_percent | 40 | 50 | 5 |
| facade_4_window_area_percent | 40 | 50 | 5 |
| start_occupation_monday (h) | 7 | 9 | 1 |
| start_occupation_tuesday (h) | 7 | 9 | 1 |
| start_occupation_wednesday (h) | 7 | 9 | 1 |
| start_occupation_thursday (h) | 7 | 9 | 1 |
| start_occupation_friday (h) | 7 | 9 | 1 |
| end_occupation_monday (h) | 17 | 20 | 1 |
| end_occupation_tuesday (h) | 17 | 20 | 1 |
| end_occupation_wednesday (h) | 17 | 20 | 1 |
| end_occupation_thursday (h) | 17 | 20 | 1 |
| end_occupation_friday (h) | 17 | 20 | 1 |

TABLE A.1 – List of parameters contained in $\lambda$ and $\delta$, along with sampling and calibration ranges. During training of the metamodel, occupation values are converted in a one dimensional time series, with value $0$ or $1$ based on the occupation state of the building.

### A.1.3 Additional air quality forecasting samples

## A.2 Prior models

### A.2.1 Details on the loss function

*Proof of Lemma 5.3.1.* By definition,

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\varphi}\left[\log \frac{p_\theta(\mathsf{z}^{0:T}, \mathsf{z}_e^{0:T}, x)}{q_\varphi(\mathsf{z}^{0:T}, \mathsf{z}_e^{0:T}|x)}\right],$$

which yields

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\varphi}\left[\log p_\theta^x(x|\mathsf{z}^0)\right] + \mathbb{E}_{q_\varphi}\left[\log \frac{p_\theta^{\mathsf{z}}(\mathsf{z}^{0:T}|\mathsf{z}_e^{0:T})}{q_\varphi^{\mathsf{z}}(\mathsf{z}^{0:T}|\mathsf{z}_e^{0:T})}\right] + \mathbb{E}_{q_\varphi}\left[\log \frac{p_\theta^{\mathsf{z}_e}(\mathsf{z}_e^{0:T})}{q_\varphi^{\mathsf{z}_e}(\mathsf{z}_e^{0:T}|x)}\right].$$

| Variable | Minimum | Maximum | Step |
|---|---|---|---|
| start_clim_day (h) | 7 | 9 | 1 |
| end_clim_day (h) | 18 | 20 | 1 |
| t_clim_red_day (°C) | 24 | 30 | 0.5 |
| t_clim_conf_day (°C) | 20 | 24 | 0.5 |
| start_heat_day (h) | 6 | 8 | 1 |
| end_heat_day (h) | 17 | 19 | 1 |
| t_heat_red_day (°C) | 17 | 22 | 0.5 |
| t_heat_conf_day (°C) | 22 | 24 | 0.5 |
| start_ventilation_day (h) | 7 | 9 | 1 |
| end_ventilation_day (h) | 18 | 20 | 1 |
| t_ventilation_day (°C) | 18 | 26 | 0.5 |
| vol_ventilation_day | 0.7 | 1.7 | 0.3 |

TABLE A.2 – List of variables contained in $\psi_k$, along with their ranges. Each parameter can hold a different value for each day of the week. For ease of reading, we replaced them by a single line, as the ranges are the same for every day.

| Variable | Description |
|---|---|
| DNI | Direct Normal Irradiance |
| IBEAM_H | Direct Horizontal Irradiance |
| IBEAM_N | Direct Normal Irradiance |
| IDIFF_H | Diffuse Horizontal Irradiance |
| IGLOB_H | Global Horizontal Irradiance |
| RHUM | Outdoor Relative Humidity |
| TAMB | Outdoor temperature |

TABLE A.3 – Weather data as contained in $\varphi_k$.

The last term may be decomposed as

$$\mathbb{E}_{q_\phi}\left[\log \frac{p_\theta^{\mathsf{z}_e}(\mathsf{z}_e^{0:T})}{q_\varphi^{\mathsf{z}_e}(\mathsf{z}_e^{0:T}|x)}\right] = \mathbb{E}_{q_\varphi}\left[\log p_{\theta,T}^{\mathsf{z}_e}(\mathsf{z}_e^T)\right] + \sum_{t=1}^{T} \mathbb{E}_{q_\varphi}\left[\log \frac{p_{\theta,t-1|t}^{\mathsf{z}_e}(\mathsf{z}_e^{t-1}|\mathsf{z}_e^t)}{q_{\varphi,t|t-1}^{\mathsf{z}_e}(\mathsf{z}_e^t|\mathsf{z}_e^{t-1})}\right]$$

and

$$\mathbb{E}_{q_\phi}\left[\log \frac{p_\theta^{\mathsf{z}_e}(\mathsf{z}_e^{0:T})}{q_\varphi^{\mathsf{z}_e}(\mathsf{z}_e^{0:T}|x)}\right] = \mathbb{E}_{q_\varphi}\left[\log p_{\theta,T}^{\mathsf{z}_e}(\mathsf{z}_e^T)\right] + \mathbb{E}_{q_\varphi}\left[\log \frac{p_{\theta,0|1}^{\mathsf{z}_e}(\mathsf{z}_e^0|\mathsf{z}_e^1)}{q_{\varphi,1|0}^{\mathsf{z}_e}(\mathsf{z}_e^1|\mathsf{z}_e^0)}\right] + \sum_{t=2}^{T} \mathbb{E}_{q_\varphi}\left[\log \frac{p_{\theta,t-1|t}^{\mathsf{z}_e}(\mathsf{z}_e^{t-1}|\mathsf{z}_e^t)}{q_{\varphi,t|t-1}^{\mathsf{z}_e}(\mathsf{z}_e^t|\mathsf{z}_e^{t-1})}\right].$$

By (5.1),

$$\mathbb{E}_{q_\phi}\left[\log \frac{p_\theta^{\mathsf{z}_e}(\mathsf{z}_e^{0:T})}{q_\varphi^{\mathsf{z}_e}(\mathsf{z}_e^{0:T}|x)}\right] = \mathbb{E}_{q_\varphi}\left[\log \frac{p_{\theta,T}^{\mathsf{z}_e}(\mathsf{z}_e^T)}{q_{\varphi,T|0}^{\mathsf{z}_e}(\mathsf{z}_e^T|\mathsf{z}_e^0)}\right] + \sum_{t=2}^{T} \mathbb{E}_{q_\varphi}\left[\log \frac{p_{\theta,t-1|t}^{\mathsf{z}_e}(\mathsf{z}_e^{t-1}|\mathsf{z}_e^t)}{q_{\varphi,t-1|0,t}^{\mathsf{z}_e}(\mathsf{z}_e^{t-1}|\mathsf{z}_e^0,\mathsf{z}_e^t)}\right] + \mathbb{E}_{q_\varphi}\left[\log p_{\theta,0|1}^{\mathsf{z}_e}(\mathsf{z}_e^0|\mathsf{z}_e^1)\right],$$

which concludes the proof. □

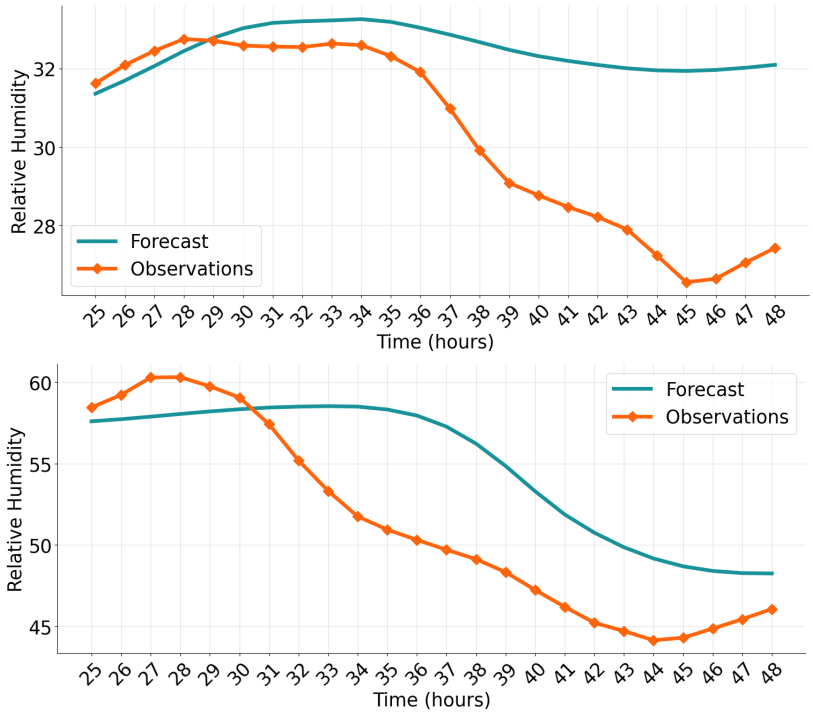
### A.2.2 Inpainting diffusion sampling

We consider the case in which we know a sub-part of the picture $\overline{X}$, and want to predict the complementary pixels $\underline{X}$. Knowing the corresponding $n$ latent vectors $\overline{\mathsf{z}_e}^0$ which result from $\underline{X}$ through the encoder, we sample $N-n$ $\underline{\mathsf{z}_e}^T$

| Variable | Description |
|---|---|
| Q_AC_OFFICE | AC consumption |
| Q_HEAT_OFFICE | Heat consumption |
| Q_PEOPLE | Heating power due to human activities in the building |
| Q_EQP | Consumption of equipment, such as computers, elevators, fridges |
| Q_LIGHT | Consumption of lights |
| Q_AHU_C | Consumption of AHU when cooling outside air |
| Q_AHU_H | Consumption of AHU when heating outside air |
| T_INT_OFFICE | Indoor temperature |

TABLE A.4 – Output variables of the equivalent model designed by the energy managers.

FIGURE A.3 – Prediction of Relative Humidity given observations on two additional 24 hour samples, by the LSTM model. The neural network is able to model general trends, but fails at grasping hour to hour behaviors.




from the uninformative distribution $\underline{z_e}^T \sim \mathcal{N}(0, (2\vartheta)^{-1}\eta^2 \mathbf{I}_{d\times(N-n)})$. In order to produce the chain of samples $z_e^{t-1}$ from $z_e^t$ we then follow the following procedure.

— $\underline{z_e}^{t-1}$ is predicted from $z_e^t$ using the neural network predictor, similar to the unconditioned case.

— Sample $\overline{z_e}^{t-1}$ using the forward bridge noising process.

### A.2.3 Additional regularisation considerations

We consider here details about the parameterisation of $p^z_\theta(z^t|z_e^t)$ and $q^z_\phi(z^t|z_e^t)$ in order to compute $\mathcal{L}_t^{reg}(\theta, \varphi)$. Using the Gumbel-Softmax formulation provides an efficient and differentiable parameterisation.

$$p^z_{\theta,t}(z^t = \cdot|z_e^t) = \text{Softmax}\{(-\|z_e - e_k\|_2^2 + G_k)/\tau_t\}_{1\leqslant k\leqslant K}\,,$$
$$q_{\phi,t}(z^t = \cdot|z_e^t) = \text{Softmax}\{(-\|z_e - e_k\|_2^2 + \tilde{G}_k)/\tau\}_{1\leqslant k\leqslant K}\,,$$

where $\{(G_k, \tilde{G}_k)\}_{1 \leqslant k \leqslant K}$ are i.i.d. with distribution $\mathrm{Gumbel}(0,1)$, $\tau > 0$, and $\{\tau_t\}_{0 \leqslant t \leqslant T}$ are positive time-dependent scaling parameters. Then, up to the additive normalizing terms,

$$\mathcal{L}_t^{reg}(\theta, \phi) = \mathbb{E}_{q_\varphi}\left[\log \frac{p^{\mathsf{z}}_{\theta,t}(\mathsf{z}^t|\mathsf{z}_e^t)}{q^{\mathsf{z}}_{\varphi,t}(\mathsf{z}^t|\mathsf{z}_e^t)}\right] = \left(-\frac{1}{\tau_t} + \frac{1}{\tau}\right)\|\mathsf{z}_e^t - \widehat{\mathsf{z}^t}\|_2^2 - \frac{\tilde{G}_k}{\tau} + \frac{G_k}{\tau_t}\,,$$

where $\widehat{\mathsf{z}^t} \sim q^{\mathsf{z}}_{\phi,t}(\mathsf{z}^t|\mathsf{z}_e^t)$. Considering only the first term which depend on $\mathsf{z}_e^t$ and produce non-zero gradients, we get :

$$\mathcal{L}_t^{reg}(\theta, \phi) = \gamma_t \|\mathsf{z}_e^t - \widehat{\mathsf{z}^t}\|_2^2$$

where $\gamma_t = -1/\tau_t + 1/\tau$ drives the behavior of the regulariser. By choosing is $\gamma_t$ negative for large $t$, the regulariser pushes the codebooks away from $\mathsf{z}_e^t$, which prevents too early specialization, or matching of codebooks with noise, as $\mathsf{z}_e^{t \approx T}$ is close to the uninformative distribution. Finally, for small $t$, choosing $\gamma_t$ positive helps matching codebooks with $\mathsf{z}_e$ when the corruption is small. In practice $\tau = 1$ and a simple schedule from $10$ to $0.1$ for $\tau_t$ was considered in this work.

### A.2.4 Neural Networks

For $\varepsilon_\theta(\mathsf{z}_e^t, t)$, we use a U-net like architecture similar to the one mentioned in **?**. It consists of a deep convolutional neural network with 57M parameters, which is slightly below the PixelCNN architecture (95.8M parameters). The VQ-VAE encoder / decoders are also deep convolutional networks totalling 65M parameters.

### A.2.5 Toy Example Appendix

**Parameterisation** We consider a neural network to model $\varepsilon_\theta(\mathsf{z}_e^t, t)$. The network shown in Figure A.4 consists of a time embedding similar to **?**, as well as a few linear or 1D-convolutional layers, totalling around $5000$ parameters.

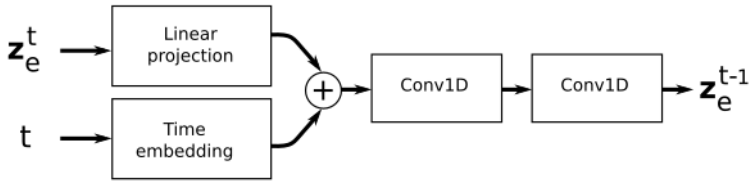



FIGURE A.4 – Graphical representation of the neural network used for the toy dataset.

For the parameterisation of the quantization part, we choose $p^{\mathsf{z}}_{\theta,t}(\mathsf{z}^t = \mathrm{e}_j|\mathsf{z}_e^t) = \mathrm{Softmax}_{1 \leq k \leq K}\{-\|\mathsf{z}_e - \mathrm{e}_k\|_2\}_j$, and the same parameterisation for $q^{\mathsf{z}}_{\phi,t}(\mathsf{z}^t|\mathsf{z}_e^t)$. Therefore our loss simplifies to :

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\varphi}\left[\log p^x_\theta(x|\mathsf{z}^0)\right] + \mathcal{L}_t(\theta, \varphi)\,,$$

where $t$ is sampled uniformly in $\{0, \ldots, T\}$.

**Discrete samples during diffusion process** Discrete sequences corresponding to the denoising diffusion process shown in Figure 5.3 are shown in Table A.5.

**End-to-end training** In order to train the codebooks alongside the diffusion process, we need to backpropagate the gradient of the likelihood of the data $\mathsf{z}_e$ given a $\mathsf{z}_e^0$ reconstructed by the diffusion process (corresponding to

| t | NN sequence |
|---|---|
| 50 | (0, 7, 3, 6, 2) |
| 40 | (6, 5, 5, 5, 3) |
| 30 | (5, 5, 5, 4, 2) |
| 20 | (6, 6, 5, 4, 3) |
| 10 | (5, 6, 5, 4, 3) |
| 0 | (5, 6, 5, 4, 3) |

TABLE A.5 – Discrete samples during diffusion process. The discrete sequence is obtained by computing the nearest neighbour centroid $\mu_j$ for each $X_s^t$. At $t = 0$, $X^0$ is sampled from a centered Gaussian distribution with small covariance matrix $(2\vartheta)^{-1}\eta^2\mathbf{I}_{2\times5}$, resulting in a uniform discrete sequence, as all centroids have a similar unit norm.

$\mathcal{L}^{rec}(\theta, \phi)$). We use the Gumbel-Softmax parameterisation in order to obtain a differentiable process and update the codebooks $\mathrm{e}_j$.

In this toy example, the use of the third part of the loss $\sum_{t=0}^{T} \mathcal{L}_t^{reg}(\theta, \phi)$ is not mandatory as we obtain good results with $\mathcal{L}_t^{reg}(\theta, \varphi) = 0$, which means parametrising $p^{\mathsf{z}}_{\theta,t}(\mathsf{z}^t|\mathsf{z}_e^t) = q^{\mathsf{z}}_{\varphi,t}(\mathsf{z}^t|\mathsf{z}_e^t)$. However we noticed that $\mathcal{L}_t^{reg}(\theta, \varphi)$ is useful to improve the learning of the codebooks. If we choose $\gamma_t$ to be decreasing with time $t$, we have the following. When $t$ is low, the denoising process is almost over, $\mathcal{L}_t^{reg}(\theta, \varphi)$ pushes $\mathsf{z}_e$ and the selected $\mathsf{z}$ to close together : $\|\mathsf{z}_e\| \sim 1$, then $\|\mathsf{z}_e^t\|$ will be likely near a specific $\mathrm{e}_j$ and far from the others ; therefore only a single codebook is selected and receives gradient. When $t$ is high, $\|\mathsf{z}_e^t\| \sim 0$ and the Gumbel-Softmax makes it so that all codebooks are equidistant from $\|\mathsf{z}_e^t\|$ and receive non-zero gradient. This naturally solves training problem associated with dead codebooks in VQ-VAEs. Joint training of the denoising and codebooks yield excellent codebook positionning as shown in Figure A.5.

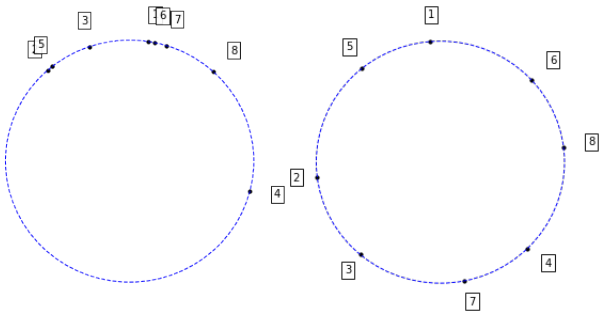


FIGURE A.5 – Left, initial random codebooks positions. Right, after training, position of codebook vectors. Note that the codebook indexes do not match the indexes of the Gaussians, the model learnt to make the associations between neighboring centroids in a different order.

**Toy Diffusion inpainting** We consider a case in which we want to reconstruct an $x$ while we only know one (or a few) dimensions, and sample the others. Consider that $x$ is generated using a sequence $q = (q_1, q_2, q_r, q_4, q_5)$ where the last one if fixed $q_1 = 0, q_5 = 4$. Then, knowing $q_1, q_5$, we sample $q_2, q_3, q_4$, as shown in Figure A.6.

### A.2.6 Additional visuals

**Cifar**

**MiniImageNet**

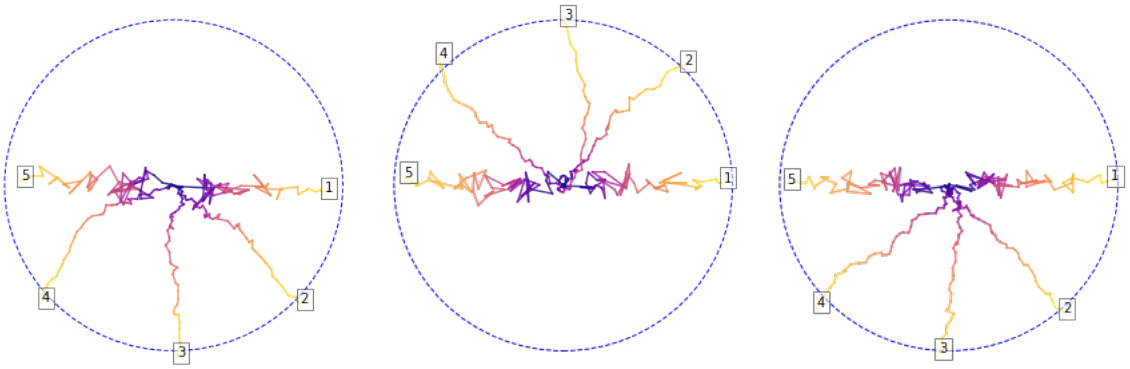



FIGURE A.6 – Three independent sampling of $X$ using a trained diffusion bridge, with fixed $q_1 = 0, q_5 = 4$. The three corresponding sequences are $(0, 7, 6, 5, 4)$, $(0, 1, 2, 3, 4)$, $(0, 7, 6, 5, 4)$ all valid sequences.

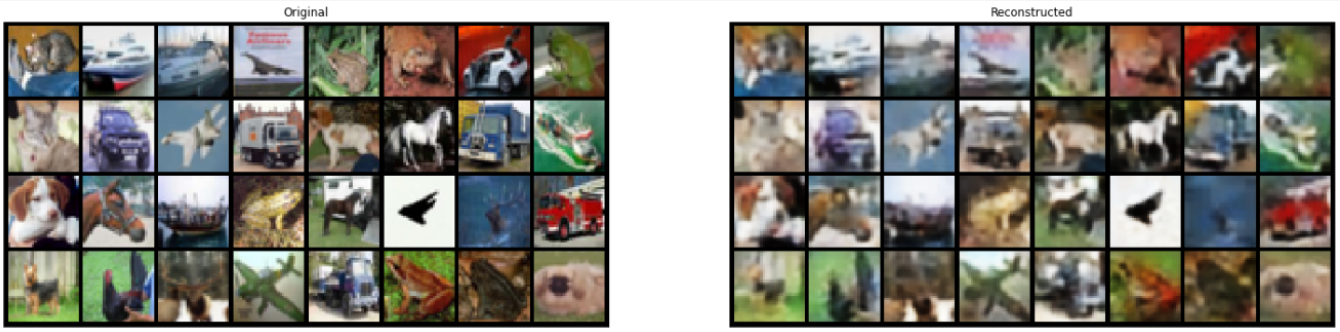



FIGURE A.7 – Reconstruction of the VQVAE model used in the following benchmarks.

FIGURE A.8 – Samples from the PixelCNN prior (left) and from our diffusion prior (right) on CIFAR10.

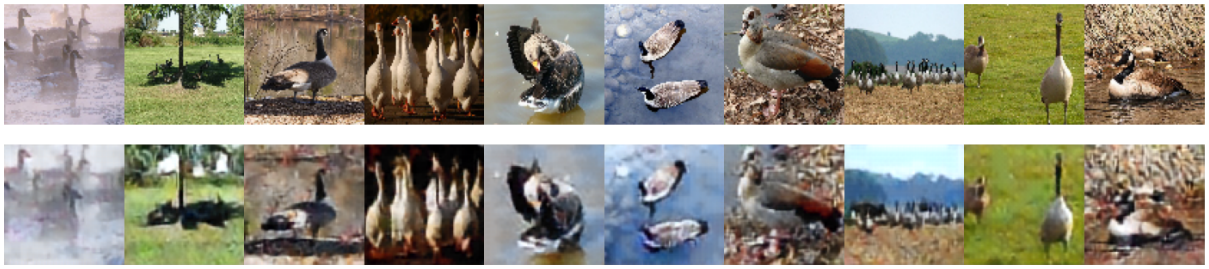


FIGURE A.9 – Reconstruction of the trained VQ-VAE on the *mini*ImageNet dataset. Original images are encoded, discretised, and decoded.


FIGURE A.10 – Samples from our model for the miniimagenet dataset

FIGURE A.11 – Conditional sampling : Top : reconstructions from the vqvae of originals images, Middle : conditional sampling with the left side of the image as condition, for our model. Bottom 1 and 2 : conditional sampling in the same context with the PixelCNN prior.

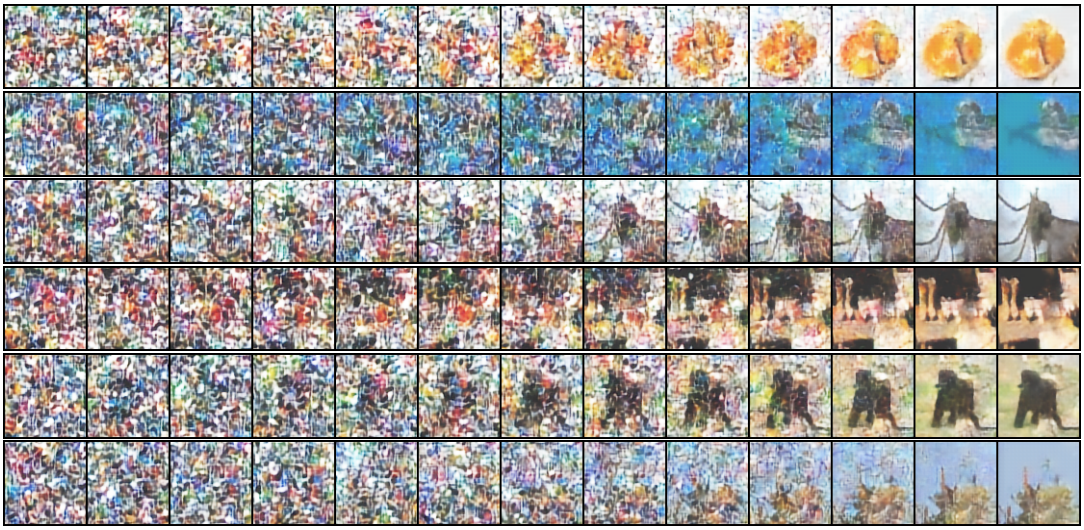


FIGURE A.12 – Sampling denoising chain from up to $t = 0$, shown at regular intervals, conditioned on the left part of the picture. The sampling procedure is described in Appendix A.2.2.

ECOLE
DOCTORALE
DE MATHEMATIQUES
HADAMARD

**Titre :** Métamodèles et approches bayésiennes pour les systèmes dynamiques.

**Mots clés :** Métamodèle ; apprentissage profond ; méthodes de Monte Carlo séquentielles ; inférence variationnelle ; modèles génératifs profonds ; quantification de l'incertitude.

**Résumé :** Dans ce manuscrit, nous développons des architectures d'apprentissage profond pour modéliser la consommation énergétique et la qualité de l'air de bâtiments.

Nous présentons d'abord une méthodologie de bout-en-bout permettant d'optimiser la demande énergétique tout en améliorant le confort, en substituant au traditionnel simulateur physique un modèle numériquement plus efficace. A partir de données historiques, nous vérifions que les simulations de ce métamodèle correspondent aux conditions réelles du bâtiment. Cependant, les performances des prédictions sont dégradées dans certaines situations à cause de différents facteurs alétoires.

Nous proposons alors de quantifier l'incertitude des prédictions en combinant des modèles à espaces d'état à des modèles d'apprentissage profond pour les séries temporelles. Dans une première approche, nous montrons comment les poids d'un modèle peuvent être affinés par des méthodes de Monte Carlo séquentielles, afin de prendre en compte l'incertitude sur la dernière couche. Nous proposons un second modèle génératif à états latents discrets, permettant une procédure d'apprentissage moins coûteuse par Inférence Variationnelle ayant des performances équivalentes sur une tâche de prévision de l'humidité relative.

Enfin, notre dernière contribution étend l'utilisation de ces modèles discrets, en proposant une nouvelle loi a priori basée sur des ponts de diffusion. En apprenant à corrompre puis à reconstruire des échantillons de l'espace latent, notre modèle est capable d'apprendre la distribution a priori, quelle que soit la nature des données.

**Title :** Metamodel and bayesian approaches for dynamic systems

**Keywords :** Metamodel ; deep learning ; sequential monte carlo ; variational inference ; deep generative models ; uncertainty quantification.

**Abstract :** In this thesis, we develop deep learning architectures for modelling building energy consumption and air quality.

We first present an end-to-end methodology for optimizing energy demand while improving indoor comfort, by substituting the traditionally used physical simulators with a much faster surrogate model. Using historic data, we can ensure that simulations from this metamodel match the real conditions of the buildings. Yet some differences remain, due to unavailable and random factors.

We propose to quantify this uncertainty by combining state space models with time series deep learning models. In a first approach, we show how the weights of a model can be finetuned through Sequential Monte Carlo methods, in order to take into account uncertainty on the last layer. We propose a second generative model with discrete latent states, allowing for a simpler training procedure through Variational Inference and equivalent performance on a relative humidity forecasting task.

Finally, our last work extends on these quantized models, by proposing a new prior based on diffusion bridges. By learning to corrupt and reconstruct samples from the latent space, our model is able to learn the complex prior distribution, regardless of the nature of the data.

**Institut Polytechnique de Paris**
91120 Palaiseau, France