Étude de l'augmentation de données pour la robustesse des réseaux de neurones profonds

Jean Michel Amath Sarr

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

UNIVERSITÉ CHEIKH ANTA DIOP

Sorbonne Université

Université Cheikh Anta Diop

EDITE 130 – UMMISCO

ÉTUDE DE L'AUGMENTATION DE DONNÉES POUR LA ROBUSTESSE DES RÉSEAUX DE NEURONES PROFONDS

Par Jean Michel Amath Sarr

Thèse de doctorat d'Informatique

Dirigée par Christophe Cambier et Alassane Bah

Présentée et soutenue publiquement le 13 janvier 2023

Devant un jury composé de :

PRÉSIDENT DU JURY :	DATHE Hamidou	Professeur Titulaire	Université Cheikh Anta DIOP - Sénégal
RAPPORTEURS :	HENRIET Julien	Maître de Conférences HDR	Université de Franche-Comté - France
	BOUSSO Mamadou	Maître de Conférences CAMES	Université de Thies - Sénégal
EXAMINATEURS :	ZUCKER Jean-Daniel	Directeur de Recherche	Sorbonne Université, Institut de Recherche et Développement - France
	EZIN Eugène C.	Professeur Titulaire	Université d'Abomey-Calavi - Bénin
DIRECTEURS DE THÈSE :	BAH Alassane	Professeur Titulaire	Université Cheikh Anta DIOP - Sénégal
	CAMBIER Christophe	Maître de Conférences HDR	Sorbonne Université, Institut de Recherche et Développement - France

Table des matières

Remerciements

Je tiens à saisir cette occasion pour remercier mes deux directeurs de thèse Pr. Alassane Bah et Pr. Christophe Cambier qui ont accepté de superviser mon travail. Alassane Bah m'a aidé à focaliser mon attention sur la revue de littérature et la question scientifique. Mais il m'a aussi aidé à concentrer mes efforts de manière générale. Christophe Cambier m'a permis de prendre du recul sur mon travail et à adopter une attitude plus durable dans le processus de recherche. Leurs remarques se sont révélées précieuses pour persévérer dans la réalisation de ce manuscrit. Je tiens à remercier tous les membres de mon jury : Pr. Hamidou Dathe, Pr. Julien Henriet, Pr. Mamadou Bousso, Dr. Jean-Daniel Zucker, Pr. Eugène C. Ezin.

Je remercie Dr. Thimothée Brochier, pour m'avoir donné l'opportunité d'appliquer les réseaux de neurones sur la reconnaissance du fond de l'océan. Je remercie aussi Pr. Georgios Yannakakis avec qui j'ai collaboré pour appliquer les réseaux de neurones dans le contexte de la ludification.

Je remercie mes bailleurs : l'État du Sénégal, l'IRD à travers le Programme Doctoral International Modélisation des Systèmes Complexes (PDI-MSC) qui a permis ce travail en cotutelle entre l'Université Cheikh Anta Diop et Sorbonne Université, et enfin le Google PhD Fellowship.

Je remercie les organisations qui m'ont ouvert sur le monde : *Deep Learning Indaba* qui m'a permis d'assister aux *Deep Learning Indaba 2018* à Stellenbosch University en Afrique du Sud, et *Deep Learning Indaba 2019* à Kenyatta University au Kenya. Je remercie aussi ProbAI qui m'a permis d'assister à *Nordic Probabilistic AI School 2019* à NTNU, Trondheim en Norvège. Enfin, je remercie Black in AI qui m'a permis d'assister à *NeurIPS 2019* à Vancouver au Canada. Les rencontres que j'ai faites m'ont permis de faire évoluer mes intérêts scientifiques et de mieux positionner mon sujet.

Enfin, un grand merci à mes parents, qui m'ont soutenu depuis mon plus jeune âge, et grâce auxquels j'ai pu me concentrer sur la recherche scientifique.

Abstract

In this thesis, we considered the problem of the robustness of neural networks. We have considered the case where the training and deployment datasets are not independently and identically distributed from the same source, this condition being referred to as dataset shift . Our main research axis has been data augmentation. Indeed, an extensive literature review and preliminary experiments showed us the regularization potential of data augmentation. Thus, as a first step, we sought to use data augmentation to make neural networks more robust to various synthetic and natural dataset shifts. However, the results of this approach was mixed. Indeed, we observed that in some cases the augmented data could lead to performance jumps on the deployment set. But this phenomenon did not occur every time. In some cases, the augmented data could even reduce performance on the deployment set. In our conclusion, we offer a granular explanation for this phenomenon. Better use of data augmentation toward neural network robustness is to generate stress tests to observe a model behavior when various shift occurs. Then, to use that information to estimate the error on the deployment set of interest even without labels, we call this deployment error estimation. Furthermore, we show that the use of independent data augmentation can improve deployment error estimation. We believe that this use of data augmentation will allow us to better quantify the reliability of neural networks when deployed on new unknown datasets.

Keywords : dataset shift, robustness, data augmentation, deployment error estimation, deep neural networks

Résumé

Dans cette thèse, nous avons considéré le problème de robustesse des réseaux de neurones. C'est-à-dire que nous avons considéré le cas ou le jeu d'apprentissage et le jeu de déploiement ne sont pas indépendamment et identiquement distribués suivant la même source. On parle alors de glissement de données. Notre principal outil de travail a été l'augmentation de données. En effet, une revue approfondie de la littérature et des expériences préliminaires nous ont montré le potentiel de régularisation de l'augmentation des données. Ainsi, dans un premier temps, nous avons cherché à utiliser l'augmentation de données pour rendre les réseaux de neurones plus robustes à divers glissements de données synthétiques et naturels. Cependant, les résultats de cette approche se sont révélés mitigés. En effet, nous avons observé que dans certains cas l'augmentation de données pouvait donner lieu à des bonds de performance sur le jeu de déploiement. Mais ce phénomène ne se produisait pas à chaque fois. Dans certains cas, augmenter les données pouvait même réduire les performances sur le jeu de déploiement. Nous proposons une explication granulaire à ce phénomène dans nos conclusions. Une meilleure utilisation de l'augmentation des données pour la robustesse des réseaux de neurones consiste à générer des tests de résistance ou "stress test" pour observer le comportement d'un modèle lorsque divers glissements de données surviennent. Ensuite, ces informations sur le comportement du modèle sont utilisées pour estimer l'erreur sur l'ensemble de déploiement même sans étiquettes, nous appelons cela l'estimation de l'erreur de déploiement. Par ailleurs, nous montrons que l'utilisation d'augmentation de données indépendantes peut améliorer l'estimation de l'erreur de déploiement. Nous croyons que cet usage de l'augmentation de données permettra de mieux cerner quantitativement la fiabilité des réseaux de neurones lorsqu'ils seront déployés sur de nouveaux jeux de données inconnus.

Mots-clés : glissement de données, robustesse, augmentation de données, estimation de l'erreur de déploiement, réseaux de neurones profonds

Sigles et Abréviations

Symboles	Signification
ATC	Average Threshold Accuracy
AdaIN	Adaptative Instance Normalization
BNN	Bayesian Neural Network
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutional Neural Network
COCO	Common Object in Context
DA	Data Augmentation
DE	Deployment Error
DenseNet	Densely Connected Convolutional Neural Networks
ECE	Expected Calibration Error
ELBO	Expected Lower Bound
ERM	Empirical Risk Minimization
FGSM	Fast Gradient Sign Method
FMoW	Functional Map of the World
GDE	Generalized Disagreement Equality
GRU	Gated Recurrent Unit
i.i.d	indépendamment et identiquement distribué
ISDA	Implicit Semantic Data Augmentation
KL	Divergence de Kullback-Leibler
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
mCE	Mean Corruption Error
MNIST	Modified National Institute of Standards and Technology
MPIW	Mean Prediction Interval Width
NaN	Not a Number
OOD	Out Of Distribution
OTDD	Optimal Transport Dataset Distance
PICP	Prediction Interval Coverage Probability
RF	Random Forest
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ResNet	Deep Residual Convolutional Neural Networks
SELU	Scale Exponential Linear Unit
SVM	Support Vector Machine
VAE	Variational Autoencoder

Table des figures

Liste des tableaux

Chapitre Un

Introduction Générale

Sommaire

1.1 Contexte : risques d'un déploiement trop hatif des réseaux de neurones

Cette thèse a d'abord commencé par un résultat étrange que nous avons obtenu lors d'une simulation de déploiement d'un modèle d'apprentissage profond à des données nouvelles. En effet, notre performance sur le jeu de déploiement était supérieure à notre performance sur le jeu d'apprentissage, nous présentons ce résultat au chapitre 6. Cette incompréhension a dirigé notre travail de recherche. En essayant de comprendre le résultat, nous avons pris conscience que la fiabilité des modèles d'apprentissage machine repose entièrement sur l'hypothèse que les données d'apprentissage et de déploiement doivent être identiquement et indépendamment distribuées. Nous nommerons cette hypothèse : l'hypothèse $i.i.d$ dans la suite. En pratique, cette condition est difficile à garantir. Ce qui limite le déploiement des réseaux de neurones aux applications ayant une portée critique pour la société. En particulier pour des applications où une différence de prédiction peut signifier un risque pour :

— la stabilité économique (trading à haute fréquence),
— la médecine (imagerie dermatologique et médecine personnalisée),
— la justice sociale (reconnaissance faciale),
— le transport (voitures autonomes).

En effet, aujourd'hui, ces différentes applications peuvent entraîner un risque si on ne peut pas garantir que les données de déploiement sont identiques aux données d'apprentissages. Nous illustrons chacun de ces points dans la suite :

— Le 6 mai 2010, plusieurs indices boursiers américains dont le S&P 500, le Dow Jones et le Nasdaq se sont effondrés emportant avec eux mille milliards de dollars américains avant de retrouver leur cour ordinaire. Cet épisode à été appelé le "Flash

Crash" par les économistes. Récemment, des chercheurs se sont penchés sur la question [Kirilenko et al., 2017] et ont estimé que les algorithmes de trading à haute fréquence ont vraisemblablement exacerbé un déséquilibre du marché suite à des ventes exceptionnelles effectuées par des opérateurs traditionnels. Les algorithmes de trading à haute fréquence peuvent incorporer des modules d'apprentissage profond [Arévalo et al., 2016], et nous pensons que si de tels algorithmes étaient déployés à grande échelle, les risques de crash peuvent être non négligeables si les données réelles sont trop différentes des données d'apprentissage.

— En imagerie dermatologique [D'Amour et al., 2020] : une représentativité plus faible de certains types de peau (mates, et foncées) dans les données d'apprentissage peut conduire à des prédictions plus variables lorsqu'un modèle est déployé. Un exemple typique de glissement de sous-population [Koh et al., 2021]. Ce type de variabilité pourrait être une différence de traitement octroyé aux minorités ethniques.

— En médecine personnalisée, les scores de risque polygénique sont basés sur une moyenne pondérée de la présence de certaines variations dans les nucléotides de l'ADN (On parle aussi de Single Nucleotide Polymorphism ou SNP en anglais.). Ces variations de nucléotides sont associées au risque d'apparition de certaines maladies. Ainsi, les scores de risques polygéniques permettent de mesurer le risque de certaines maladies rien qu'à partir du code génétique d'un patient. L'intérêt de cette technologie est de permettre une médecine personnalisée directement à partir du code génétique du patient. Cependant la plupart des nucléotides qui ont été trouvés avec les études d'association pangénomique actuelles ("Genome-wide association study" cf GWAS) sont d'ascendance européenne, ce qui peut conduire à des résultats imprévisibles sur des populations différentes. En effet, une même maladie peut être liée à différents ensembles de nucléotides pour des populations d'origines différentes. Et les nucléotides actuellement identifiés sont plus en mesure de prédire des risques chez des populations européennes que chez des populations asiatiques ou africaines par exemple. Ce qui représente un manque à gagner pour ces populations en termes d'accès à la médecine personnalisée.

— Pour la reconnaissance faciale, la sous-représentation de certains groupes dans le jeu d'apprentissage peut réduire les performances du modèle après le déploiement. Par exemple, la classification erronée du sexe est généralement un problème pour les femmes à la peau plus foncée [Buolamwini and Gebru, 2018, Wilson et al., 2019]. Ainsi, on peut craindre que certaines personnes passent plus de temps à la police pour justifier de leur identité dans les aéroports si les systèmes de reconnaissance faciale étaient déployés globalement.

— Dans le transport, les voitures autonomes utilisent des données collectées par des appareils photo ou des caméras. Si les données d'apprentissage ont été prises en hiver, alors le modèle déployé ne pourra traiter avec succès que des données collectées en hiver. Car des données collectées en été n'auraient pas la même luminosité ni les mêmes les contrastes. Par ailleurs, les modèles de vision par ordinateur seraient tributaires des caractéristiques des capteurs. C'est-à-dire que changer les capteurs constitue une violation de l'hypothèse $i.i.d.$

Dans la suite nous introduisons la problématique et notre plan de thèse.

1.2 Problématique

Aujourd'hui, la généralisation hors distribution, c'est-à-dire la tâche consistant à obtenir des modèles fiables même en contexte de glissement de données est un problème ouvert ayant une description éparse à travers de nombreuses littératures. Lorsque l'hypothèse *i.i.d* ne tient plus on parle de glissement des données ("dataset shift"), de glissement de distribution ("distribution shift"), de covariance décalée ("covariate shift"), et on parle aussi de généralisation hors distribution ("out of distribution generalization") lorsque l'on cherche généraliser hors hypothèse *i.i.d*. Dans ce manuscrit, nous allons nous concentrer plus spécifiquement sur un sous-problème de la généralisation hors distribution, le problème de la robustesse des réseaux de neurones profonds. En quelques mots, un modèle robuste est un modèle fiable, même hors des conditions spécifiées par l'apprentissage. En d'autres termes, un modèle robuste conserve un bon niveau de performance même lorsqu'il est confronté à une situation nouvelle. Mais cela pose de nouvelles questions. Dans quelle mesure un modèle est fiable suite à un glissement de distribution donnée ?

— Peut-on améliorer la fiabilité des réseaux de neurones lorsque nous les déployons sur des données inconnues ?

— Peut-on quantifier la fiabilité d'un réseau de neurones lors du déploiement sur des données inconnues même en l'absence d'étiquettes ?

Ces questions s'articulent directement sur la problématique de robustesse. En effet, premièrement, il n'y a aucune garantie sur la fiabilité d'un modèle dès lors qu'on l'expose à une source de données inconnue. Donc chercher à garantir une performance minimale, est la tâche principale d'un véritable apprentissage hors distribution, cependant nos expériences nous ont montré à quel point cette tâche est difficile. Cela a orienté notre travail vers la tâche d'estimation de la magnitude de la chute de performance sur des données inconnues non étiquetées. Cela est important, car l'étiquetage d'un jeu de données de grande taille est un procédé complexe, fastidieux, et toujours coûteux. Ainsi être en mesure d'estimer la performance d'un modèle sur des données inconnues non étiquetées peut nous indiquer si (1) un modèle doit être complètement réadapté aux nouvelles données (en cas de chute importante de la performance). Cela signifierait : investir dans l'étiquetage du nouveau jeu de données. Ou (2) déployer le modèle (en cas de chute minime de la performance), dans ce cas, on économise des ressources en utilisant le modèle tel qu'il est.

1.3 Plan de thèse

La revue de littérature pourrait être divisée en deux. En effet, les chapitres 2 et 3 seront des simples rappels pour un lecteur déjà familier avec les réseaux de neurones profonds. Par contre les chapitres 4 et 5 entreront en profondeur sur la problématique de robustesse ainsi que les outils les plus prometteurs pour attaquer le problème. Nos contributions s'articulent à travers :

— la mise en évidence du problème de généralisation hors distribution au chapitre 6,

— une investigation empirique de l'augmentation de données pour la généralisation puis sur la généralisation hors distribution au chapitre 7,

— une application de l'augmentation de données à l'estimation de l'erreur de déploiement sur des données non étiquetées au chapitre 8.

Nous détaillons le contenu de chaque chapitre dans la suite.

— Dans le chapitre 2 nous présentons l'apprentissage machine traditionnel.

— Au chapitre 3 nous présentons les réseaux de neurones comme approximateurs universels. Nous décrivons quelques architectures utilisées dans cette thèse. Ainsi, pour des tâches de vision, nous présentons les réseaux de neurones convolutifs. Pour les des tâches comprenant des séquences, nous décrivons les réseaux de neurones récurrents. Enfin, nous présentons aussi les réseaux de neurones Bayésiens. Nous nous sommes intéressés à ces derniers pour leur capacité à encoder de l'information à priori et pour leur capacité à être incertain.

— Ensuite au chapitre 4, nous discutons des approches pour faire face à un apprentissage hors contexte *i.i.d.* Nous mettons dans un cadre commun l'apprentissage par transfert, l'adaptation de domaine, et la généralisation hors distribution. Ensuite, nous définissons un sous-problème de la généralisation hors distribution qui est appelé le problème d'apprentissage robuste dans la littérature. C'est sur ce problème que nous concentrons nos travaux de thèse.

— Au chapitre 5, nous faisons l'état de l'art des méthodes appliquées au problème d'apprentissage robuste. Notre revue de littérature nous a poussé à choisir l'augmentation de données comme piste principale pour améliorer les performances en contexte de glissement de données. Par ailleurs, nous avons aussi noté que mesurer la performance des modèles en contexte hors distribution était un problème à part entière. Or, la mesure de la performance est un point clé lors de l'évaluation des interventions pour rendre les modèles robustes. Nous avons donc aussi présenté l'état de l'art de la mesure de la robustesse.

— Notre première contribution est venue d'une application pratique au chapitre 6. Il s'agissait d'apprendre sur des données acoustiques collectées en 2011 sur le large des côtes Ouest Africaines, et d'utiliser le modèle sur des données collectées en 2015. Ce cas pratique introduisit un résultat étrange, à savoir que la performance s'améliorait lorsque l'on passait du jeu d'apprentissage (2011) au jeu de déploiement (2015). Ici, nous avons utilisé un apprentissage inter-domaine pour améliorer davantage nos performances sur le jeu de déploiement 2015. Il s'agit de la méthode la plus naïve pour améliorer les performances sur un nouveau jeu de données. Ce travail nous a motivés à mieux comprendre le phénomène de généralisation hors distribution.

— Nous avons conduit une série de sept expériences au chapitre 7 traitant de deux thèmes : (1) augmentation de données et généralisation, (2) augmentation de données et généralisation hors distribution.

1. Le premier thème englobe une série de trois expériences où nous parvenons à mieux comprendre empiriquement l'effet de l'augmentation de données pour améliorer la généralisation. La plupart de ces résultats étaient déjà connus, mais nous ont permis de mieux cerner les effets de l'augmentation de données sur la généralisation. Le point le plus important que nous avons noté dans ce premier thème est l'amélioration significative de l'efficacité d'apprentissage. Un apprentissage est efficace lorsqu'un modèle arrive à un niveau acceptable de perfor-

mance avec très peu de données. Nous sommes parvenus à apprendre un modèle capable de générer des lettres manuscrites avec 100 exemples. Il s'agissait ici d'une application pratique de l'augmentation de données pour un apprentissage efficace.

2. Le second thème englobe une série de quatre expériences, où nous avons essayé d'appliquer l'augmentation de données de diverses manières pour améliorer la capacité de généralisation des modèles pour différents contextes de glissements de données (synthétiques et naturels). Ici, nous utilisons le terme essayé, car les résultats sont nuancés. Déjà, il est très difficile de créer des augmentations de données en quantité arbitraire. Par ailleurs, parfois, un schéma d'augmentation de données améliorera la généralisation hors distribution et parfois la réduira. Nous revenons sur une explication de ce phénomène en conclusion de la thèse. En donnant un axe pour trouver des augmentations de données utiles pour mieux généraliser sur les données de déploiement, même en l'absence d'étiquettes.

— Enfin, le chapitre 8 traite du thème de l'augmentation de données pour l'estimation de l'erreur de déploiement. Ce chapitre englobe trois expériences, et nous tirons profit des résultats que nous avons trouvé au cours du chapitre 7. En effet, nous abandonnons l'idée d'utiliser l'augmentation de données pour améliorer la généralisation hors distribution. En revanche nous l'utilisons pour créer des environnements de stress test aux modèles. Ainsi, nous utilisons l'augmentation de données pour estimer l'erreur de généralisation hors distribution pour un nouveau domaine. Là, nous innovons en apportant une manière plus fine de quantifier la variance autour de l'erreur de généralisation hors distribution (que nous appelons aussi erreur de déploiement dans le manuscrit). Ainsi, il était possible d'estimer l'erreur de déploiement, mais l'incertitude autour de cette erreur était quantifiée de manière simpliste, sans aucun contrôle des hypothèses sous-jacentes. Et dans plusieurs cas, le niveau d'incertitude n'était pas non plus révélé. Le danger avec une mauvaise quantification de l'incertitude ou aucune quantification de l'incertitude, est qu'un modélisateur peut déployer un modèle s'il juge que l'erreur de déploiement estimée est faible. Cependant, que se passerait-il si le modèle en question admet une grande variance de performance en cas de glissement de données. Alors, on peut s'attendre à des comportements inattendus post déploiement.

Chapitre Deux

Apprentissage machine

Sommaire

2.1 Introduction

Les ordinateurs nous surpassent quand il s'agit de calculer, en effet, il est facile à un ordinateur de calculer la racine d'un nombre réel ou de multiplier deux matrices entre elles. En d'autres termes toute opération pouvant s'interpréter formellement ou de façon logique peut être traitée à grande vitesse par les processeurs modernes.

Cependant, nous effectuons tous les jours des tâches simples ; mais extraordinairement difficiles à programmer directement, par exemple :

— reconnaître un animal sur une image.

— comprendre les paroles d'une chanson.

— jouer à un jeu de stratégie comme les échecs, les dames, le go.

— conduire une voiture.

— traduire un texte d'une langue vers une autre.

— rédiger des documents.

— effectuer un diagnostic médical.

L'apprentissage machine ("machine learning" en anglais) est le domaine d'étude qui permet aux ordinateurs d'apprendre sans être programmé explicitement. Selon Mitchel [Mitchell, 1997] :

Définition 2.1. Un algorithme apprend d'une expérience E par rapport à une tâche T mesurée par un indicateur P si la performance mesurée par P sur la tâche T s'améliore avec l'expérience E.

L'expérience fait presque toujours référence aux données. Ces données peuvent provenir de domaines très distincts comme l'image, le son, le texte. La tâche fait référence au problème que nous souhaitons voir l'ordinateur résoudre. Enfin, l'indicateur de performance est le moyen à notre disposition pour mesurer la qualité de la résolution faite par l'ordinateur. Voici un exemple simple :

Exemple 2.1. *Classification des mangues*
E : Ajouter une nouvelle mangue au panier.
T : Décider si une mangue est bonne ou mauvaise suivant ses caractéristiques.
P : Goûter une mangue et connaître son vrai goût.

Exemple 2.2. *Prédiction du prix de l'immobilier*
E : Ajouter un nouvel appartement à la base de données.
T : Prédire le prix d'un appartement en fonction de ses caractéristiques (placement, superficie, etc.).
P : Demander au acheteurs le prix auquel ils sont prêt à payer pour un appartement.

Exemple 2.3. *Généreration d'image*
E : Ajouter une photo à la base de données
T : Générer une nouvelle image
P : Estimer si une image est réaliste

Cette définition de Mitchell est utile pour comprendre comment modéliser un problème d'apprentissage supervisé. En effet, si la performance P s'améliore avec l'expérience E sur la tâche T, cela signifie que l'expérience E apporte aussi l'information des étiquettes.

Dans la section suivante, nous introduisons les termes permettant de concevoir un algorithme d'apprentissage pour résoudre la tâche de la classification discutée dans l'exemple ci-dessus. Bien que cette tâche soit très simple, comprendre comment la modéliser permet d'introduire de nombreux concepts pertinents pour le reste du manuscrit.

2.2 Modélisation d'un problème d'apprentissage machine

Imaginons que nous voulons apprendre à une machine à distinguer si une mangue est bonne ou pas. Pour cela, il faut pouvoir digitaliser les informations de la mangue pour qu'un ordinateur puisse la percevoir. Cela demande donc de structurer le problème d'une certaine manière. Pour cela, on va utiliser l'approche traditionnelle de l'apprentissage machine [Shalev-Shwartz and Ben-David, 2014].

2.2.1 Modéliser le domaine

Un domaine est un couple $(\mathcal{X}, \mathbb{P}_X)$ qui caractérise les données où $\mathcal{X} \subset \mathbb{R}^d$. Ici, $\mathcal{X} \subset \mathbb{R}^d$ suppose qu'il y a d caractéristiques que nous pouvons mesurer comme la texture, la couleur, le poids, l'odeur, etc. La loi marginale $\mathbb{P}_X$ est une loi de probabilité à support dans $\mathcal{X}$ permettant de modéliser les données. Une loi de probabilité est très pratique pour encoder les variations dans les caractéristiques des données. Par exemple une information comme :

"les mangues guinéennes sont en moyenne 50 % plus rouge que les mangues sénégalaises". Dans ce cas, les mangues guinéenne $\mathbb{P}_X^g$ et les mangues sénégalaises $\mathbb{P}_X^s$ aurait une densité différente pour la variable correspondant à la couleur. En principe, on n'a jamais accès à la loi explicite des données $\mathbb{P}_X$, à la place, on échantillone un jeu d'apprentissage $S = \{(x^{(1)}), \cdots, (x^{(m)})\}$ identiquement et indépendamment distribué (i.i.d) suivant la loi $\mathbb{P}_X$. En pratique, c'est un panier de mangues.

2.2.2 Modéliser la tâche

Une tâche est un couple $\{\mathcal{Y}, \mathbb{P}_{Y|X}\}$ où $\mathcal{Y}$ est l'espace des étiquettes. Si on souhaite classifier les mangues en bonnes ou mauvaises, alors $\mathcal{Y} = \{0, 1\}$. $\mathbb{P}_{Y|X}$ est la loi de probabilité conditionnelle prédisant l'étiquette en fonction des caractéristiques. En d'autres termes, c'est une fonction prédisant le goût. Dans le cas d'une classification binaire, chaque mangue à un goût : bon ou mauvais, donc, le goût de chaque mangue peut être modélisé à l'aide d'une loi de Bernoulli. Remarquez cependant, que ce sont les humains qui définissent cette fonction de goût. En effet, chaque personne pourrait avoir sa propre loi conditionnelle, et plusieurs personnes pourraient discuter si oui ou non une mangue donnée est bonne. Dans notre contexte, on veut apprendre à la machine à distinguer les manges bonnes des mauvaises. On va supposer que la loi conditionnelle encodant goût est fixe correspondant à un consensus. Ce qui se passe ici, c'est qu'on veut résumer une mangue x, donc une information en dimension d à une étiquette. Ainsi, une bonne mangue sera une paire $(x, 1)$, tandis qu'une mauvaise mangue paire sera une paire $(x, 0)$. Ainsi, maintenant, qu'on a une notion de goût, on peut enrichir notre échantillon S qui devient $S = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(n)})\}$ un jeu de données étiqueté. La question qui suit, est comment transférer cette connaissance à la machine ? Pour cela, on suppose :

1. que la loi $\mathbb{P}_{Y|X}$ est paramétrique, c'est à dire qu'il existe une classe de fonctions paramétriques $\mathcal{H} \coloneqq \{p_w : x \in \mathcal{X} \mapsto \sigma(w^T x) \in [0, 1],\ w \in \mathbb{R}^d\}$ pouvant modéliser la loi conditionnelle, avec σ une fonction de classification, $w^T x$ est le produit scalaire usuel dans $\mathbb{R}^d$ où w^T est la transposée du vecteur w. Dans cette modélisation chaque paramètre w correspond à une loi conditionnelle différente.
2. qu'il existe un paramètre inconnu $\hat{w}$ tel que $p_{\hat{w}} = \mathbb{P}_{Y|X}$. C'est à dire qu'il existe un paramètre $\hat{w}$ représentant la distribution des mangues bonnes ou mauvaises.

Suivant la manière dont on modélise le problème, on peut choisir deux fonctions de classification : la fonction sigmoïde ou la fonction softmax.

Classification binaire avec la fonction sigmoïde

La fonction sigmoïde permet de modéliser une loi de Bernoulli pour chaque mangue. En effet pour toute mangue x, $p_w(x)$ sera une valeur entre 0 et 1 qui désignera la probabilité que la mangue soit bonne. Et par défaut, si la mangue n'est pas bonne, c'est qu'elle est mauvaise. Ainsi, p_w est souvent appelée une hypothèse, un prédicteur ou une règle de prédiction. Car c'est une fonction associant à chaque valeur de x une estimation probabiliste $p_w(x)$ entre 0 et 1, et enfin donnant lieu à une prédiction $\hat{y}$ que nous définissons plus bas.

$$p_w(x) = \sigma(w^T x) = p(y = 1|x, w)$$

Définition 2.2. On définit la fonction sigmoïde :

$$\begin{aligned}\sigma : \mathbb{R} &\to \mathbb{R}_+ \\ : x &\mapsto \frac{1}{1+e^{-x}}\end{aligned}$$

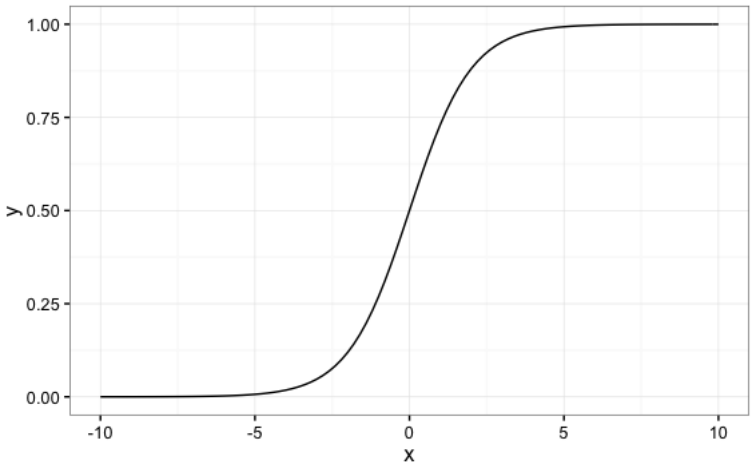


FIGURE 2.1: Fonction sigmoïde. On peut observer que $x \geq 0 \Rightarrow \sigma(x) \geq \frac{1}{2}$ et $x < 0 \Rightarrow \sigma(x) < \frac{1}{2}$. Cette fonction est souvent utilisée dans le contexte de la classification binaire.

Définition 2.3. Dans le cadre de la classification binaire on appelle ligne de décision la courbe permettant de séparer les données. Elle correspond à l'hypersurface suivante :

$$\{x \in \mathcal{X},\, p_w(x) = \frac{1}{2}\}$$

La ligne de décision découpe l'espace des données en régions de décision associées aux classes. Par exemple, $\{x \in \mathcal{X},\, p_w(x) < \frac{1}{2}\}$ représente la région de décision associée à la classe 0, et $\{x \in \mathcal{X},\, p_w(x) \geq \frac{1}{2}\}$ représente la région de décision associée à la classe 1. Ainsi, la prédiction associée à p_w est :

$$\hat{y} = \begin{cases} 1 \ si \ p_w(x) \geq \frac{1}{2} \\ 0 \text{ sinon} \end{cases}$$

La ligne de décision $\{x \in \mathcal{X}, p_w(x) = \frac{1}{2}\}$ découpe l'espace des données en 2 avec une hypersurface, ainsi toute mangue x telle que $p_w(x) \geq 0.5$ sera considérée comme bonne, sinon elle sera considérée comme mauvaise .

Classification binaire avec la fonction softmax

La fonction sigmoïde permettait de modéliser la loi conditionnelle $p(y = 1|x, w)$. Typiquement $p_w(x) = 0.4$ signifiait que $p(y = 1|x, w) = 0.4$ et donc que $p(y = 0|x, w) = 0.6$.

Une façon plus générale de procéder est de considérer les prédictions comme un vecteur associé à une loi de probabilité discrète. En d'autres termes prédire $p_w(x) = [0.6, 0.4]$, pour ce faire on utilise une généralisation de la fonction sigmoid, la fonction softmax :

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$$

Le mérite de cette approche, c'est qu'on ne considère pas différemment la classification binaire et la classification multiclasse. Ainsi dans un contexte k-multiclasse une prédiction p est un vecteur $p = (p_1, \cdots p_k)$ avec $p_i \in [0, 1]$ et $\sum_{j=1}^{k} p_j = 1$.

Nous avons obtenu un modèle permettant à une machine d'exprimer une hypothèse sur le goût des mangues. Cependant, il faut encore enseigner à la machine notre perspective humaine du goût discuté plus haut. Pour cela, il faut choisir le meilleur paramètre pour cette tâche, c'est-à-dire, celui collant le plus avec notre notion du goût $\mathbb{P}_{Y|X}$. C'est là que la notion d'apprentissage entre en jeu. Il faut essayer un grand nombre de paramètres, et sélectionner le paramètre correspondant à notre notion du goût. Cette sélection ne peut se faire que si l'on sait mesurer la performance d'une hypothèse p_w. C'est-à-dire si l'on sait corriger l'hypothèse de la machine sur le goût des mangues.

2.2.3 Mesurer la performance

Fonction coût

Étant donné un prédicteur p_w, on a besoin de mesurer le coût de l'erreur commise par la machine sur le goût d'une mangue. Cela se fait avec une fonction perte associée au problème considéré. En toute généralité une fonction perte est de la forme :

$$l : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}^+$$

En pratique, pour notre modèle de classification binaire, on utilise la fonction convexe suivante :

$$l(x, y, p_w) = -(y \cdot \log(p_w(x)) + (1 - y) \cdot \log(1 - p_w(x)))$$

Remarquez que :

— si x a pour label 1 alors le deuxième terme sera nul , et si $p_w(x)$ est proche de 1, alors le coût sera faible car $\log(1) = 0$

— si x a pour label 1 alors le deuxième terme sera nul, et si $p_w(x)$ est proche de 0 alors le coût sera important car $\lim_{x \to 0^+} \log(x) = \infty$

Un raisonnement symétrique peut être fait sur le label 0. Cela nous montre que cette fonction coût pénalise les erreurs exponentiellement et encourage le prédicteur à ne pas faire d'erreur trop grossière. Il s'agit de la fonction coût issue du maximum de vraisemblance. Nous reviendrons sur comment trouver cette fonction au chapitre 4 dans notre discussion de la robustesse statistiques.

Importance du choix de la fonction coût

En apprentissage machine, il est souvent supposé que la fonction coût est connue. Cependant, cette dernière a une influence sur la nature de l'apprentissage. En effet, elle reflète les croyances du modélisateur de ce qu'est un bon modèle. Et donc influe sur sa sélection.

En effet, reprenons notre exemple de classification de mangue vu en introduction. Pour rappel, une mangue est représentée par un vecteur $x \in \mathcal{X}$ et sa catégorie est bonne $y = 1$ ou mauvaise $y = 0$. On va imaginer un couple $(x, 1)$ correspondant à une bonne mangue. Imaginons maintenant que nous ayons deux modèles m_1 et m_2 ayant respectivement les prédictions probabilistes $m_1(y|x) = [0.1, 0.9]$ et $m_2(y|x) = [0.4, 0.6]$. Les deux modèles ont tous deux une prédiction catégorielle correcte $\hat{y} = 1$ car ils ont su deviner la classe de x.

Ainsi la fonction coût $l_{0-1}(x, y, h) = \mathbb{1}[\hat{y} - y]$ donnerai 0 pour m_1 et m_2. Car cette fonction pénalise un modèle lorsqu'il se trompe sans pénaliser les niveaux de certitudes. Donc une procédure d'apprentissage dotée de cette fonction coût, accorderait la même valeur aux modèles m_1 et m_2. En d'autres termes cette fonction coût cherche à obtenir des prédictions de classe correcte, mais n'incite pas à fournir de bons niveaux de certitudes.

La question qu'on peut se poser maintenant est comment concevoir une fonction coût sélectionnant les modèles avec les "bons" niveaux de certitudes. La question n'est pas triviale, en effet, ordonner des nombres réels se fait naturellement, c'est pour ça qu'on utilise des fonctions coûts, ces dernières ordonnent les modèles potentiels en fonction de leur coût. Mais en faisant cette transformation d'une prédiction sous forme de vecteur à un coût scalaire, ne perd-on pas de l'information ? Comment être sûr que l'ordonnancement final soit cohérent pour le modélisateur ? La littérature statistique [Gneiting and Raftery, 2007] propose la propriété de scores corrects et de scores strictement corrects pour discriminer en faveur de l'honnêteté dans les prédictions du modélisateur. Nous reviendrons là-dessus au chapitre 4.

Performance sur un échantillon et ses limitations

Maintenant que nous savons calculer une erreur numérique sur chaque mangue, on peut calculer l'erreur commise sur tout l'échantillon S. Il s'agit de l'erreur moyenne du prédicteur p_w (aussi appelée risque empirique) calculée sur l'échantillon S :

$$\mathcal{R}_S(p_w) \coloneqq \frac{1}{m} \cdot \sum_{i=1}^{m} l((x^{(i)}, y^{(i)}), p_w)$$

Ce coût, calculé sur notre panier de mangues, n'est malheureusement pas indicatif des performances sur d'autres mangues. Pour cela, il faut passer à limite. En effet, comme le domaine et la tâche, sont bien définis, on peut définir une loi jointe associée au problème d'apprentissage $\mathbb{P} = \mathbb{P}_X \times \mathbb{P}_{Y|X}$.

Généralisation

Pour calculer le risque réel effectué par le prédicteur p_w sur les mangues venant du Sénégal $\mathbb{P}$ il faut calculer l'espérance de la variable aléatoire $l(\cdot, p_w)$:

$$\mathcal{R}_{\mathbb{P}}(p_w) \coloneqq \mathbb{E}_{(x,y)\sim\mathbb{P}}[l((x, y), p_w)]$$

Calculer cette espérance n'est pas facile. En effet, dans la réalité, on ne connaît jamais vraiment l'expression de la loi génératrice des données $\mathbb{P}$. Donc on ne peut pas se baser sur une formule toute faite. En présence d'une quantité illimitée de données, le risque empirique converge vers le risque réel par la loi des grands nombres. C'est la bénédiction

des mégadonnées (big data). Mais dans la plupart des cas, on a seulement accès à un jeu de données limité. C'est là qu'intervient la généralisation, qui est en réalité une estimation du risque réel.

En pratique on utilise un jeu d'essai T *i.i.d* suivant $\mathbb{P}$ différent de S et on mesure le risque empirique $\mathcal{R}_T(p_w)$ cette approche se justifie en montrant que $\mathcal{R}_T(p_w) \approx \mathcal{R}_{\mathbb{P}}(p_w)$, i.e le risque empirique commis sur un jeu d'essai est approximativement égale au risque réel commis par p_w. Cela se montre par l'inégalité d'Hoeffding.

Maintenant que nous avons modélisé le problème d'apprentissage pour une machine, on peut définir un algorithme d'apprentissage.

2.3 Algorithme d'apprentissage

Le but de l'apprentissage est de trouver dans la classe d'hypothèse $\mathcal{H}$ un "bon" prédicteur, c'est-à-dire un prédicteur généralisant bien.

$$\begin{aligned} A : \mathbb{P} &\to \mathcal{H} \\ : S &\mapsto p_w^S \end{aligned}$$

En d'autres termes, c'est une procédure qui pour tout jeu d'apprentissage S tiré aléatoirement d'une source de données va trouver une hypothèse paramétrique p_w^S. On reconnaît un parallèle entre les algorithmes d'apprentissage et l'estimation en statistiques. Un estimateur est un algorithme d'apprentissage. Pour trouver un prédicteur pertinent, il est nécessaire d'explorer la classe d'hypothèse $\mathcal{H}$. En pratique, cela se modélise par la formulation d'un problème d'optimisation.

Trouver la meilleure approximation p_w revient ici à minimiser le risque empirique sur $\mathcal{R}_S$ sur l'espace des paramètres $\mathbb{R}^d$:

$$p_w^S = \arg\min_{w \in \mathbb{R}^d} \mathcal{R}_S(p_w) = \min_{w \in \mathbb{R}^d} \frac{1}{m} \cdot \sum_{i=1}^{m} l(x^{(i)}, y^{(i)}, p_w) \tag{2.1}$$

Ici, on distingue bien un prédicteur quelconque p_w d'un prédicteur issu d'un apprentissage sur un jeu de données S que nous notons p_w^S. De nombreuses méthodes existent pour résoudre ce problème, la plus connue étant la descente du gradient [Robbins and Monro, 1951], mais on retrouve aussi des approches plus moderne comme Adam [Kingma and Ba, 2014].

2.4 Régularisation

La machine est donc en capacité de trouver un prédicteur p_w dans une classe de fonction $\mathcal{H}$ en fonction d'un échantillon $S \sim \mathbb{P}$. Mais que faire si le prédicteur a une faible performance ? Il y a plusieurs manières de procéder, mais un bon départ serait de diagnostiquer la performance du prédicteur en décomposant le risque en biais et complexité.

2.4.1 Décomposition du risque : sous-apprentissage et sur-apprentissage

On peut décomposer le risque réel en deux composantes : le risque d'approximation et le risque d'estimation que nous notons respectivement ϵ_{app} et ϵ_{est}.

$$\mathcal{R}_{\mathbb{P}}(p_w^S) = \epsilon_{app} + \epsilon_{est}$$

avec

$$\epsilon_{app} = \min_{w \in \mathbb{R}^d} \mathcal{R}_{\mathbb{P}}(p_w),\ \epsilon_{est} = \mathcal{R}_{\mathbb{P}}(p_w^S) - \epsilon_{app}$$

Le choix de la classe d'hypothèses paramétriques $\mathcal{H}$ influence les deux erreurs, alors que le choix de l'échantillon S n'affecte que le risque d'estimation.

— Élargir la class $\mathcal{H}$ (c'est-à-dire augmenter le nombre de paramètres de la classe d'hypothèse) sans toucher au jeu d'apprentissage, va diminuer le risque d'approximation et augmenter le risque d'estimation, dans ce cas il y a sur-ajustement ou sur-apprentissage.

— Réduire la classe $\mathcal{H}$ sans toucher au jeu d'apprentissage, va à augmenter le risque d'approximation et diminuer le risque d'estimation. Dans ce cas, il y a sous-ajustement ou sous-apprentissage.

— Augmenter la taille de l'échantillon S va permettre au prédicteur p_w^S d'être plus proche du meilleur prédicteur sur la distribution $\mathbb{P}$ donc réduit le risque d'estimation.

On peut se représenter l'effet du choix de la classe $\mathcal{H}$ sur ces erreurs comme un levier sur des vases communiquant en observant la figure 2.2. En effet :

$$\text{Élargir } \mathcal{H} \Rightarrow \searrow \epsilon_{app}\ et\ \nearrow \epsilon_{est}$$

$$\text{Réduire } \mathcal{H} \Rightarrow \nearrow \epsilon_{app}\ et\ \searrow \epsilon_{est}$$

$$\text{Augmenter } S\ \Rightarrow \searrow \epsilon_{est}$$

Cette approche de décomposition du risque est un bon axe théorique, mais en pratique, il est très difficile d'estimer ε_{app}, car on n'a pas accès à la distribution générant les données $\mathbb{P}$. Donc, on diagnostique le sous-apprentissage et le sur-apprentissage avec une heuristique sur l'erreur d'apprentissage et l'erreur de test. En effet, $\mathcal{R}_{\mathbb{P}}(p_w^S) \approx \mathcal{R}_T(p_w^S)$ avec T identiquement et indépendamment distribué suivant $\mathbb{P}$. Donc on utilise le schéma suivant : $\mathcal{R}_T(p_w^S) = (\mathcal{R}_T(p_w^S) - \mathcal{R}_S(p_w^S)) + \mathcal{R}_S(p_w^S)$. Intuitivement, lorsque le terme $(\mathcal{R}_T(p_w^S) - \mathcal{R}_S(p_w^S))$ est grand on a sur-ajustement, et lorsque le terme $\mathcal{R}_S(p_w^S)$ est grand, il y a sous-ajustement.

2.4.2 Taxanomie des méthodes de régularisation

Selon Kukacka et al. [Kukačka et al., 2017] :

Définition 2.4. La régularisation est toute technique qui vise à ce que le modèle généralise mieux, c'est-à-dire qu'il produise de meilleurs résultats sur l'ensemble de test.

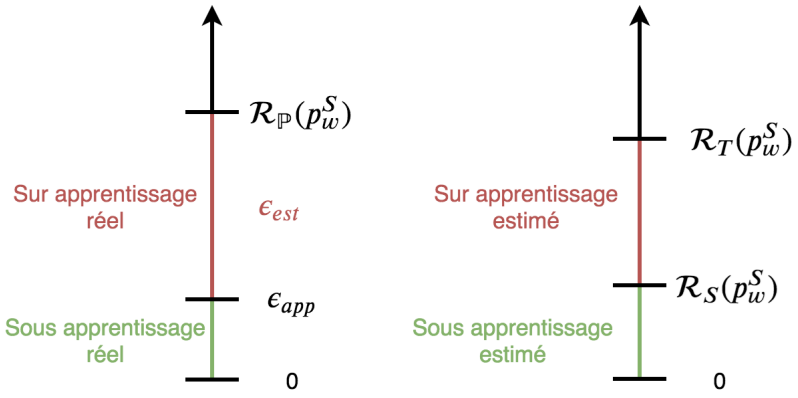


FIGURE 2.2: Décomposition respectivement théorique (gauche) et heuristique (droite) du risque d'un prédicteur p_w^S entraîné sur un jeu S.

Un des objectifs de Kukacka et al. est d'offrir une taxonomie pour classifier les techniques de régularisation, en effet, ces dernières sont nombreuses et variées. Ainsi, pour rappel, nous cherchons à obtenir une fonction paramétrique p_w réalisant de bonnes performances sur le jeu de test avec la procédure suivante :

$$\arg\min_w \frac{1}{|S|} \sum_{x,y \in S} l(x, y, p_w) + R(\cdot)$$

Dans cette équation, il y a

- un terme de minimisation arg min représentant la procédure d'optimisation
- p_w un ensemble de modèles,
- l une fonction coût,
- un terme de régularisation R
- les données d'apprentissages S

Ainsi, on peut :

- Changer la méthode d'optimisation.
- Élargir ou réduire la classe $\mathcal{H}$.
- Changer la fonction coût.
- Ajouter un terme de régularisation.
- Obtenir un jeu d'apprentissage plus important.

Exemple 2.4. *Dégradation des pondérations*
Dans cet exemple, nous illustrons la méthode dégradation des podération sur la tâche de régression logistique définie précédemment.

- *Régression logistique :* $h_S = \min_{w \in \mathbb{R}^d} \frac{1}{m} \cdot \sum_{i=1}^{m} l(x^{(i)}, y^{(i)}, h_w) + \lambda ||w||_2$

Avec λ un hyper-paramètre.

Cela revient à imposer à l'algorithme d'expliquer les données avec des poids de normes minimales. Cela incite à rechercher les hypothèses performantes les plus simples. Philosophiquement, la recherche de simplicité est le principe heuristique du "rasoir d'Ockham" autrement appelé principe de parcimonie que l'on peut énoncer ainsi : *"Les hypothèses suffisantes les plus simples sont les plus vraisemblables"*. En d'autres termes, la recherche de simplicité s'encode par la contrainte dans la recherche des paramètres.

2.5 Conclusion

Nous venons de voir les briques principales de l'apprentissage machine. Ces dernières s'articulent autour de trois axes :

— la digitalisation d'un domaine pour qu'une machine puisse percevoir,

— la modélisation d'une tâche que nous souhaitons voir la machine faire,

— la mesure de la performance pour qu'une machine puisse s'améliorer.

Un algorithme d'apprentissage est une procédure permettant à la machine de "s'essayer" à la tâche et de sélectionner le programme le plus performant. Cependant, comme on le verra au chapitre 4, la fiabilité des modèles chute dès lors que les données d'apprentissage et de test ne viennent pas de la même distribution. Comme la régularisation a pour but d'obtenir de bonnes performances sur le jeu de test, nous allons prendre la taxonomie de Kukacka comme point de départ pour résoudre notre problème. Plus particulièrement, nous discuterons des raisons d'élargir la classe $\mathcal{H}$ avec l'apprentissage profond au chapitre 3. Au chapitre 4, nous discuterons des statistiques robustes comme moyen de concevoir des fonction coûts entraînant une meilleure généralisation. Enfin, au chapitre 5, nous discuterons de l'augmentation de données comme moyen d'augmenter le jeu d'apprentissage, mais aussi comme moyen d'ajouter un terme de régularisation à la fonction coût.

Chapitre Trois

Apprentissage profond

Sommaire

3.1 Introduction : le problème des représentations

Considérons le jeu de données en Figure 3.1. Supposons que notre objectif soit de séparer les données avec l'ensemble des lignes droites de type $\mathcal{H} = \{p_w : (x, y) \in \mathbb{R}^2 \mapsto w_1 x + w_2 y + w_3, w \in \mathbb{R}^3\}$. La performance sera alors faible, car nous commettrons des erreurs au moins sur la moitié des données. Car aucune ligne droite ne peut parfaitement séparer les données. On dit que les données ne sont pas linéairement séparables.

Une solution pour améliorer la performance serait de changer la représentation des données de sorte à ce qu'elles soient facilement séparables. Cela est possible en passant des coordonnées cartésiennes aux coordonnées polaires. C'est-à-dire en utilisant une fonction $R : (x, y) \mapsto (r(x, y), \theta(x, y))$ transformant les données de sorte à les rendre séparables. Il s'agit d'une fonction r définissant le rayon, et d'une fonction θ définissant l'angle d'un point du plan.

$$r(x,y) = \sqrt{x^2 + y^2},\ \theta(x,y) = \begin{cases} \arctan(\frac{y}{x}), \text{ si } x > 0, y > 0 \\ \arctan(\frac{y}{x}) + 2\pi, \text{ si } x > 0, y < 0 \\ \arctan(\frac{y}{x}) + \pi, \text{ si } x < 0 \\ \frac{\pi}{2}, \text{ si } x = 0, y > 0 \\ \frac{3\pi}{2}, \text{ si } x = 0, y < 0 \end{cases}$$

Ainsi, pour le même modèle, il est beaucoup plus facile d'utiliser notre classe de lignes droites pour séparer les données.

$$\mathcal{H} = \{p_w : (r, \theta) \in \mathbb{R} \times]-\pi, \pi] \mapsto w_1 r + w_2 \theta + w_3, w \in \mathbb{R}^3\}$$

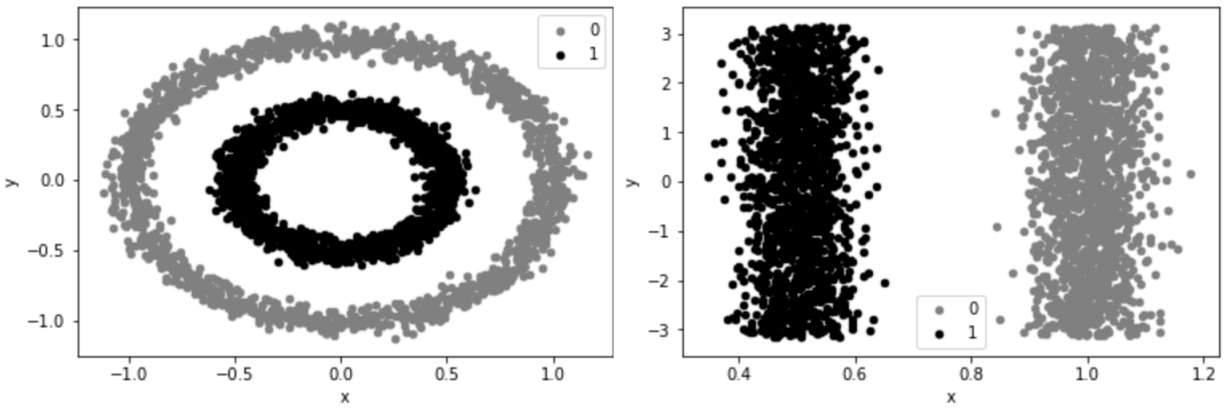


FIGURE 3.1: Changement de coordonnées. À gauche, les données sont exprimées en coordonnées cartésiennes (x, y). À droite, les données sont exprimées en coordonnées polaires (r, θ).

Dans cet exemple, nous avions affaire avec un jeu de données bidimensionnel. Nous pouvions choisir visuellement une transformation des données pertinente pour séparer facilement les données. Mais dans la réalité, les données réelles (images, enregistrements audios, données issues de capteurs) sont souvent multidimensionnelles et la recherche de représentations pertinentes est souvent longue, fastidieuse et non intuitive.

Et si la recherche de représentations R pouvait être faite automatiquement ? C'est exactement la promesse de l'apprentissage profond. Dans la suite, nous discutons des réseaux de neurones standards et de leurs propriétés générales. Puis nous décrivons en détail les réseaux de neurones convolutifs et les réseaux de neurones récurrents particulièrement adaptés pour des problèmes de traitement d'image ou d'apprentissage séquentiel. Enfin, nous présentons les réseaux de neurones Bayésiens pour leurs propriétés théoriques et pour leur capacité à induire de l'invariance, une des approches les plus utilisées pour le problème de robustesse.

3.2 Apprentissage profond

La performance des algorithmes d'apprentissage dépend beaucoup des représentations que l'on se fait des données. Une bonne représentation des données est une perspective permettant de mieux synthétiser les données. Par exemple, il est beaucoup plus facile d'effectuer des opérations arithmétiques (additions, soustractions, multiplications, divisions) avec des chiffres arabes, qu'avec des chiffres romains. Un exemple en haute dimension serait une tâche de reconnaissance de véhicule sur une photo par exemple. En effet, une photo pour une machine, est un ensemble de pixels donc une suite de chiffres sans cohérence réelle. Notre cerveau a appris à reconnaître des caractéristiques de haut niveau : roues, carrosserie, marques, allure générale. Mais comment encoder ces caractéristiques de haut niveau par une représentation $\mathcal{R}$ compréhensible pour une machine ? Une option serait d'apprendre directement les représentations, et c'est là que les réseaux de neurones artificiels se sont révélés d'une étonnante efficacité.

3.2.1 Les réseaux de neurones : approximateurs universels

Les réseaux de neurones sont des couches de neurones transformant successivement les données. Un neurone standard est la composée d'une application non-linéaire f et d'une application linéaire $w^T x$. Ici, un neurone est une fonction $x \mapsto f(w^T x)$. Cette approche permet de composer plusieurs neurones et d'augmenter la complexité de la représentation des données $\mathcal{R}$:

$$f_\Theta(x) = f_{d-1}(\cdots \theta_2 f_1(\theta_1^T x)\cdots) \tag{3.1}$$
$$= f_{\theta_1:\theta_d}(x) \tag{3.2}$$
$$\tag{3.3}$$

Avec

— $\theta_1, \cdots, \theta_d$ des matrices telles que le produit matriciel $\theta_d \cdot \theta_{d-1} \cdots \theta_1$ soit possible.

— Θ le produit cartésien des espaces dans lesquels les θ_i sont définis. C'est l'espace de tous les paramètres.

— $f_1, \cdots, f_{d-1}$ des fonctions différentiables.

Ainsi, si on reprend notre exemple de classification des mangues, on utilisera une fonction $h_\Theta : x \mapsto \sigma(f_\Theta(x))$. En augmentant le nombre de paramètres, leur dimension et leur interconnexion, on augmente significativement la complexité de la classe de prédicteurs $\mathcal{H}$, jusqu'à obtenir une classe d'approximateurs universels. En effet, plusieurs auteurs nottament Cybenco [Cybenko, 1989] ont montré que les réseaux de neurones sont des approximateurs universels. En effet, ils sont construits avec une classe de fonction dense dans l'espace des fonctions continues sur un compact $K \subset \mathbb{R}^d$. En d'autres termes, toute fonction continue définie sur un compact $K \subset \mathbb{R}^d$ à valeur réelle peut être approchée arbitrairement par un réseau de neurones. Mais en pratique, cette puissance de représentation est déterminée par le choix d'une architecture spécialisée sur le domaine des données à traiter. Cette propriété donne aux réseaux de neurones une grande modularité. Car hormis le choix de l'architecture, le modélisateur peut améliorer exponentiellement la complexité des fonctions apprises [Bengio et al., 2013a] en ajoutant des couches, ou de la profondeur, d'où le terme d'apprentissage profond . Cela étant possible par un grand nombre de composition de fonctions non-linéaires [LeCun et al., 2015].

3.2.2 Succès de l'apprentissage profond

Certainement, l'une des dates charnière concernant le changement de paradigme dans les méthodes d'apprentissage a été le résultat record en 2012 en classification d'image [Krizhevsky et al., 2012]. En effet un réseau de neurones convolutifs profond nommé AlexNet a été entraîné sur la base de données ImageNet constituée de 15 millions d'images en haute résolution réparties en 22 000 catégories. Le gain de performance d'AlexNet réduisait l'erreur faite par les autres classifieurs de moitié. La difficulté de ce challenge de classification d'image a motivé de nombreux chercheurs dans d'autres disciplines à utiliser les réseaux de neurones.

En traitement du langage naturel, les réseaux de neurones s'appliquent à de très nombreuses tâches. Par exemple, en reconnaissance de la parole, les réseaux de neurones pro-

fonds ont montré des performances supérieures aux méthodes d'apprentissage traditionnelles comme les modèles de Markov caché couplé aux modèles à mixture de Gaussiennes [Hinton et al., 2012]. Mais les réseaux de neurones ont établi l'état de l'art pour les tâches de questions-réponses, de traduction [Sutskever et al., 2014] et la génération de texte plausible [Brown et al., 2020].

Une combinaison de réseaux de neurones profonds et de recherche arborescente a aussi permis de battre le champion du monde de Go, à l'époque reconnu comme la référence en termes de jeu abstrait, dû à la taille de l'espace des possibilités comparé aux échecs ou aux dames [Silver et al., 2016].

De même en physique expérimentale, les outils des physiciens de haute énergie sont les accélérateurs de particules faisant entrer en collision des protons et/ou des anti-protons afin de découvrir des particules exotiques. L'observation de telles particules requiert l'utilisation de méthodes statistiques avancées et les méthodes d'apprentissage profond montrent une grande performance comparés aux méthodes d'apprentissage classique. Ces performances s'expliquent notamment par la capacité des réseaux de neurones à apprendre des fonctions de complexité arbitraire [Baldi et al., 2014].

Enfin, en bio-informatique, l'apprentissage profond a montré de nombreux succès en tirant profit de la grande quantité de données générée par les processus biologiques. Notons par exemple la prédiction du comportement des enzymes, la prédiction des propriétés associées à l'expression des gènes, la prédiction des fonctions de l'ADN, la classification d'image biomédicale [Li et al., 2019]. Plus récemment, les réseaux de neurones profonds ont permis la prédiction de la structure en trois dimensions de l'ADN en partant de la séquence d'acides aminés [Jumper et al., 2021].

3.3 Réseau de neurones convolutifs

3.3.1 Limites des réseaux de neurons standards

Les algorithmes traditionnels de reconnaissance de formes pour traiter les images ou la parole ont été construits à l'aide d'un extracteur de caractéristiques, puis d'un classificateur. Est-il possible d'utiliser les réseaux de neurones pour classer la parole et les images ? Eh bien, oui, en effet, on a déjà vu que théoriquement les réseaux de neurones sont capables d'apprendre des fonctions arbitrairement complexes [Cybenko, 1989]. En pratique aussi, les réseaux de neurones sont performants sur un grand nombre de tâches et une grande variété de domaines comme on l'a vu précédemment avec les succès de l'apprentissage profond. Cependant, les architectures traditionnelles rencontrent quelques problèmes pour travailler avec des données issues de divers domaines telles que les images, la parole ou les séries temporelles. Dans la suite, nous voyons quelques spécificités des images, et justifions les limites des réseaux de neurones standards sur ce type de données.

— Les images ont généralement des centaines de variables (une pour chaque pixel), et les architectures entièrement connectées auront en effet beaucoup de paramètres à apprendre ce qui est coûteux (en termes de mémoire et de calcul)

— Aussi les réseaux de neurones standards ignorent simplement la topologie des données, par exemple l'organisation des pixels dans une image n'est pas prise en compte. Ils

peuvent être présentés dans n'importe quel ordre sans influencer la sortie. Alors que dans le cas des images, les variables sont fortement corrélées.

— Les réseaux de neurones standards manquent également d'invariance intégrée par rapport aux distorsions locales. Cela signifie qu'un petit changement dans l'entrée pourrait affecter la sortie.

Ces considérations sont à l'origine des réseaux de neurones convolutifs. En effet, ces derniers reposent sur trois idées architecturales : (1) la rareté des connexions (2) le partage des paramètres et (3) la représentation équivariante ([LeCun and Bengio, 1995]). Nous présenterons succinctement l'un des premiers de ces réseaux en figure 3.2 : LeNet-5 qui a été utilisé dans la reconnaissance de chiffres écrits à la main [LeCun and Bengio, 1995].

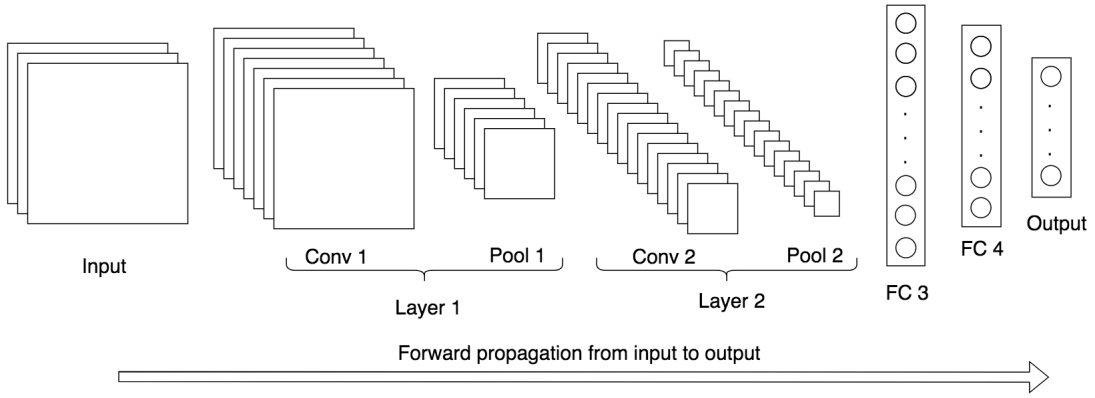


FIGURE 3.2: Architecture LeNet-5, les deux premières couches sont des couches de convolution, elles intègrent la convolution et le pooling, leur sortie est ensuite envoyée vers deux couches de neurones entièrement connectés, pour enfin faire une prédiction. La flèche illustre le flux de données à travers le réseau.

De nos jours, de multiples architectures et modèles ont vu le jour avec un énorme succès en vision par ordinateur, voici une sélection récente de références dépassant les 10 000 citations :

— AlexNet [Krizhevsky et al., 2012]
— VGGNet [Simonyan and Zisserman, 2014]
— Inception Networks [Szegedy et al., 2015]
— ResNets [He et al., 2015]
— WideResNets [Zagoruyko and Komodakis, 2016]
— Dense Networks [Huang et al., 2017]

Dans cette thèse, nous utilisons beaucoup les convolutions, à savoir dans notre expérience de déploiement en situation réelle avec des données acoustique 6. Puis lors de nos différentes expériences au chapitre 7, nous utilisons LetNet-5, et ResNet.

Cependant, bien que les convolutions puissent aussi traiter d'autres structures de données que les images comme la voix par exemple, une autre catégorie d'architecture s'est particulièrement distingué dans le traitement des données sous forme de séquence comme la voix ou le texte : les réseaux de neurones récurrents.

3.4 Réseaux de neurones récurrents

3.4.1 Apprentissage séquentiel

Les réseaux de neurones récurrents ont été utilisés avec succès dans l'étiquetage de séquences, c'est une tâche se référant à l'apprentissage de relation entre une séquence de données et une séquence d'étiquettes. Quelques exemples peuvent inclure :

— La reconnaissance de la parole ; e.g : passer d'un discours sonore à une phrase sous forme de texte.

— La traduction automatique ; e.g : passer d'un texte en anglais à texte en français.

— L'analyse des sentiments ; e.g : passer des commentaires d'un film à un nombre d'étoiles (1-5).

Pourquoi ne pas utiliser un réseau de neurones standards pour effectuer l'étiquetage de séquence ? Eh bien, comme précédemment ce type de réseau ne prend pas en compte la structure des données, chaque point est considéré comme indépendant des autres, donc des informations contextuelles peuvent être perdues. Par exemple, dans la reconnaissance vocale, le même discours peut être prononcé avec un accent différent à différentes vitesses. Prendre arbitrairement deux séquences à un pas de temps différent d'un enregistrement peut ne pas fournir beaucoup d'informations sur l'ensemble du discours. Pour résoudre ces problèmes, les réseaux de neurones récurrents que nous décrivons ci-dessous utilisent le partage de paramètres. Notons cependant que les architectures obtenant l'état de l'art pour les modèles de séquences sont les modèles à attention [Vaswani et al., 2017]. Dans la suite, nous développons les la littérature sur réseaux de neurones récurrents, car ce sont ceux que nous avons utilisés dans cette thèse.

3.4.2 Défi de l'apprentissage séquentiel

Bien que les réseaux de neurones récurrents soient très efficaces pour l'étiquetage de séquences, ils ont certains défauts :

— Premièrement, il est difficile de détecter les dépendances à long terme lorsque nous avons des séquences très longues. Par exemple, prenons la phrase suivante : "le narrateur ressent une étrange impression en apercevant trois arbres, et sourit". Dans cette phrase le sujet du verbe sourire est le narrateur, cependant, comme le sujet est "loin" du verbe sur lequel il agit, la sensibilité de cet élément contextuel s'affaiblit à mesure que le verbe est éloigné de son sujet dans un réseau de neurones récurrent. Dans la littérature, ce problème est appelé dissipation des gradients [Bengio et al., 1994].

— Deuxièmement, dans leur représentation réelle, les réseaux de neurones récurrents ne prennent en compte que les informations du passé, par essence les unités récurrentes ne sont reliées que dans un sens.

Pour faire face à ces défis, certains éléments ont été ajoutés. Le modèle de mémoire à court et long terme (LSTM) [Hochreiter and Schmidhuber, 1997] et l'unité récurrente fermée (GRU) [Cho et al., 2014] ont été capables de stocker et d'accéder à des informations sur de très longues périodes de temps (longue séquence), en redessinant la couche récurrente standard autour d'unités de mémoire. Ces deux architectures montrent des performances

comparables comme le montre Chung et ses collègues [Chung et al., 2014]. Ainsi, nous utiliserons les GRU dans nos applications des prochains chapitres, car plus modernes et plus succincts que les LSTM. Notons aussi que pour accéder à toutes les informations contextuelles (passées et futures), des réseaux de neurones récurrents bidirectionnels ont été proposés [Schuster and Paliwal, 1997]. La principale différence est que les unités récurrentes sont connectées dans les deux sens.

Dans notre première expérience du chapitre 7, nous utilisons Sketch-RNN [Ha and Eck, 2018] qui est un autoencoder variationnel composé d'un réseau récurrent bidirectionnel avec GRU comme encodeur, et un réseau récurrent avec GRU comme décodeur. Cette architecture était particulièrement bien adaptée à l'apprentissage génératif de séquences de tracés pour apprendre une distribution de lettres à dessiner.

3.5 Réseaux de neurones Bayésiens

Les réseaux de neurones Bayésiens peuvent modéliser l'incertitude en apprenant une loi a posteriori pour modéliser les paramètres du réseau. Cette modélisation de l'incertitude est très utile pour qu'un modèle puisse "savoir quand il ne sait pas". Car dans ce cas, le recours à un expert permettrait de résoudre le problème. Un autre avantage des réseaux de neurones Bayésiens est leur capacité à exploiter de petits ensembles de données en incorporant des connaissances a priori sur la problématique à modéliser. Dans la suite, nous décrivons les réseaux de neurones Bayésiens et voyons les méthodes modernes pour les entraîner. Finalement, nous voyons la méthode de Nalisnick pour incorporer de l'invariance dans un a priori.

3.5.1 Formule de Bayes et réseaux de neurones

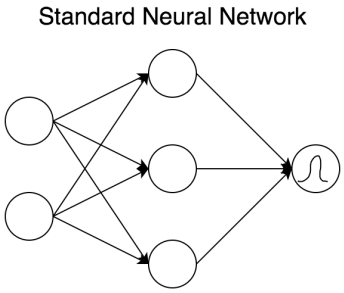
Une façon d'imaginer le fonctionnement des réseaux de neurones Bayésiens (Bayesian Neural Networks BNN) est de les comparer aux réseaux de neurones standards. La première différence est que les BNN impliquent deux réseaux de neurones avec la même architecture, un pour l'a priori et un pour l'a posteriori. Ils sont liés par la formule de Bayes :

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)}$$

Où X est un ensemble de données, $p(X)$ est l'évidence, θ sont les paramètres du réseau, $p(X|\theta)$ est la vraisemblance, enfin $p(\theta)$ et $p(\theta|X)$ sont respectivement l'a priori et l'a posteriori. La deuxième différence est que chaque poids est représenté par une loi de probabilité au lieu d'un scalaire. Il existe de nombreuses façons de sélectionner une loi pour un problème d'apprentissage. Mais, souvent, l'a priori est choisi avec une heuristique et n'implique pas d'apprentissage, et seul l'apprentissage de l'a postériori est effectué.

3.5.2 Approximation de l'a posteriori : inférence variationnelle

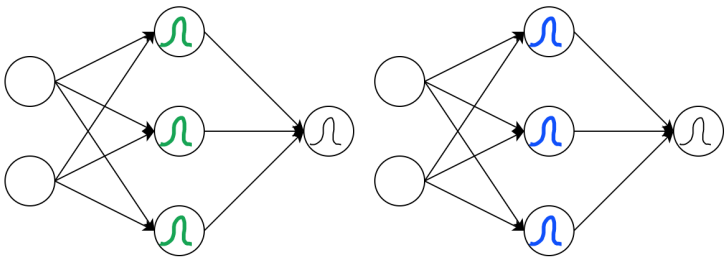
Le calcul de la vraie loi a postériori est souvent incalculable, à cause, de la difficulté à calculer $p(X)$. Ainsi, de nombreux développements techniques se sont développés pour calculer l'a posteriori. Les techniques Markov Chain Monte-Carlo (MCMC) offrent



$$\theta = \{w_1, w_2, w_3\}$$

$$p(y|x, \theta) = f(\sum_{i=1}^{3} w_i x)$$

(a)



$$\theta = \{(\mu_1, \sigma_1^2), (\mu_2, \sigma_2^2), (\mu_3, \sigma_3^2)\}$$

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

$$p(y|x) = \mathbb{E}_{\theta \sim p(\theta|\mathcal{D})} p(y|x, \theta)$$

(b)

FIGURE 3.3: (a) Réseau de neurones standard avec une couche cachée de trois neurones, les paramètres sont des scalaires. La sortie applique une fonction d'activation à une combinaison linéaire des paramètres de la dernière couche. (b) Réseau de neurones Bayésien, chaque poids est modélisé par une loi de probabilité. Comme l'a posteriori est le fruit d'un ratio souvent incalculable, car le dénominateur est impossible à calculer on utiliser des méthodes comme l'inférence variationnelle ou chaque neurone est modélisé par une loi paramétrique connue. Par exemple, ici, on a choisi de modéliser chaque neurone avec une loi Gaussienne paramétrée par une moyenne μ et une variance σ. Enfin la prédiction finale est calculée en moyennant les prédictions obtenues pour des paramètres tirées de la loi a posteriori.

les meilleures garanties théoriques pour trouver la vraie loi a posteriori, mais sont souvent peu pratiques avec des données volumineuses car elles ont besoin de l'ensemble de données complet pour apprendre [Chen et al., 2014, Balan et al., 2015]. L'inférence variationnelle est une autre méthode d'inférence bayésienne, l'idée centrale est d'utiliser une approximation variationnelle de l'a posteriori avec une loi de probabilité paramétrique. Puis l'a posteriori est mis à jour avec l'ELBO (Expected Lower Bound) : un terme composé de la vraisemblance et de la divergence de Kullback-Leibler entre l'a posteriori et l'approximation variationnelle de l'a posteriori. Cette technique a attiré beaucoup d'attention récemment, et l'approximation variationnelle a été utilisée dans le contexte des réseaux de neurones Bayésiens. Les premières approches utilisaient des Gaussienne entièrement factorisées [Graves, 2011, Blundell et al., 2015]. Cependant, des lois d'approximations plus sophistiquées ont été développées pour l'apprentissage de l'a postériori variationnel par exemple Louizos et Welling [Louizos and Welling, 2016] utilisaient une matrice de covariance complète, Gal et Ghahramani [Gal and Ghahramani, 2016] ont montré que Dropout correspondait à une approximation Bayésienne. Plus tard, Louizos et Welling [Louizos and Welling, 2017] ont introduit les flux normaux pour permettre une approximation de l'a posteriori plus flexible. Récemment, d'autres auteurs [Mescheder et al., 2017] ont fait valoir que les méthodes implicites pourraient offrir une approximation plus flexible que les méthodes inversibles telles que les flux normaux. Par exemple, Pawlowski et al. [Pawlowski et al., 2017] ont utilisé un hyperréseau pour modéliser l'a posteriori, de même, Shi et al.[Shi et al., 2017] ont introduit un algorithme plus efficace appelé Matrix Multiplication Neural Network (MMNN) pour échantillonner une grande matrice afin de modéliser l'a posteriori. En résumé, il existe de nombreuses loi d'approximation pour modéliser la loi a posteriori.

3.5.3 Modélisation de l'a priori

Les connaissances a priori peuvent exprimer des d'hypothèses sur la fonction de génération de données. Ainsi, plusieurs hypothèses ont trouvé un encodage approprié directement dans les poids des réseaux de neurones, notamment la compression des représentations : avec une induction de parcimonie sur les paramètres du réseau [Molchanov et al., 2017, Neklyudov et al., 2017] , la compression peut aussi être faite en tronquant les poids, par exemple cela consiste à tronquer des poids flottant de 32 chiffres à des poids flottants de 16 ou 8 chiffres [Ullrich et al., 2017]. Enfin, certaines approches permettent de mélanger l'hypothèse de parcimonie avec la troncation des paramètres [Louizos et al., 2017, Federici et al., 2017]. De même, Nalisnick et Smyth [Nalisnick and Smyth, 2018] ont proposé une méthodologie pour incorporer directement une hypothèse d'invariance dans l'a priori, nous en discuterons davantage en section 7.

Comme l'a priori induit un terme de régularisation dans l'expression de l'ELBO, on voit un lien entre l'apprentissage de l'a priori et le pré-apprentissage non supervisé [Erhan et al., 2010] très utilisé en apprentissage semi-supervisé en contexte non Bayésien. Les deux méthodes sont appliquées avant l'entraînement et induisent une régularisation. Le pré-apprentissage non supervisé régularise en sélectionnant les paramètres d'initialisation du réseau de neurones en tenant compte de la distribution des données avant la tâche de classification. En d'autres termes, le pré-apprentissage non supervisé initialise des architectures de réseaux de neurones dans des régions à partir desquelles de meilleurs

bassins d'attraction peuvent être atteints, en termes d'améliorations de généralisation. Le pré-apprentissage non supervisé favorise les paramètres qui expliquent mieux les facteurs informatifs de variation dans les données X, et ces facteurs de variation sont utiles dans la tâche de prédiction de Y.

3.5.4 A priori approximativement invariant

Nalisnick a introduit une méthode élégante pour insérer de l'information dans l'a priori [Nalisnick and Smyth, 2018]. Cela est peu conventionnel, car en général l'a priori est choisi par défaut (par exemple, on suppose que les paramètres suivent une loi normale centrée réduite.), et le travail consiste à apprendre l'a posteriori. Rappelons qu'un modèle Bayésien est composé de deux réseaux de neurones (donc deux modèles), et que chacun à des neurones paramétrés par une loi de probabilité. On peut noter un modèle $p(y|x,\theta)$ ou $\theta \sim p_\lambda$, dans notre cas, la loi paramétrant chaque neurone est une Gaussienne, on a $\lambda = \{\mu, \sigma^2\}$. Ainsi, un modèle peut effectuer autant de prédictions qu'il n'a de neurones échantillonés suivant sa loi p_λ. Mais la prédiction finale du modèle sera la moyenne des prédictions obtenues suivant tous les paramètres possibles : $p(y|x) = \mathbb{E}_{\theta\sim p_\lambda}[p(y|x,\theta)]$. Dans la suite, nous discutons de la méthodologie proposée par Nalisnick pour apprendre le modèle a priori. Cela commence avec la borne supérieure suivante :

$$KL(p(y|x)||\mathbb{E}_{\hat{x}\sim q_\eta}[p(y|\hat{x})]) \leq \mathbb{E}_{\hat{x}\sim q_\eta}\mathbb{E}_{\theta\sim p_\lambda} KL(p(y|x,\theta)||p(y|\hat{x},\theta)) \tag{3.4}$$

- On considère que les données x sont augmentées $\hat{x}$ avec $\hat{x} \sim q(\hat{x}; x, \eta)$ ou q_η est une corruption paramétrée par η.
- On considère qu'un modèle est invariant si $p(y|x) = p(y|\hat{x})$ quelque soit la donnée augmentée $\hat{x}$. On suppose qu'un modèle est approximativement invariant lorsque seulement $p(y|x) = \mathbb{E}_{\hat{x}\sim q_\eta}[p(y|\hat{x})]$), c'est à dire lorsque qu'en moyenne la prédiction sur les données augmentées est égale à la prédiction sur la donnée d'origine.
- Pour mesurer la dissimilarité entre deux lois de probabilités on utilise la divergence de Kullback-Leibler $KL(p||q) = \int_\chi p(x)\log(\frac{p(x)}{q(x)})dx$.
- Le terme de gauche mesure donc la dissimilarité entre la prédiction effectuée par le modèle sur x, et la moyenne des prédictions effectuées sur les données augmentés lorsque l'on fait varier le paramètre de contrôle de l'augmentation.
- Le terme de droite est une double moyenne sur les paramètres de l'a priori et sur le paramètre des corruptions appliquées aux données.

Cette borne supérieure nous incline à minimiser le terme de droite dans l'équation 3.4. Mais en pratique, cette optimisation ferait chuter la variance σ^2 de chaque neurone, et reviendrait à transformer l'a priori en un réseau déterministe ou chaque neurone serait paramétré uniquement par sa moyenne μ. Donc Nalisnick a proposé la fonction objectif suivante à maximiser :

$$\mathcal{J}(\lambda, x) = \mathbb{H}_\lambda[\theta] - \mathbb{E}_{\hat{x}\sim q_\eta}\mathbb{E}_{\theta\sim p_\lambda} KL(p(y|x,\theta)||p(y|\hat{x},\theta)) \tag{3.5}$$

Minimiser la borne supérieure revient à maximiser son opposé, ce qui explique le second terme de l'équation, le premier terme $\mathbb{H}_\lambda[\theta] = -\int_\theta p_\lambda(\theta)\log(p_\lambda(\theta))d\theta$ est l'entropie de

l'a priori. En pratique maximiser l'entropie revient à augmenter la variance. Ainsi, cette fonction objectif empêche à la variance des neurones de tendre vers 0. Nous allons dans le chapitre 7 investiguer l'utilité de cette variance pour la généralisation et la généralisation hors distribution.

3.6 Conclusion

Dans ce chapitre, nous avons vu les réseaux de neurones artificiels standards. Puis nous avons vu en détail deux types d'architectures spécialisés pour les problèmes traitant avec les images et les séquences : les réseaux de neurones convolutifs et les réseaux de neurones récurrents. Nous nous sommes attardés sur ces détails, car nous avons utilisé ces architectures dans la plupart des expériences que nous avons conduites au chapitre 7. Nous avons aussi introduit les réseaux de neurones Bayésiens pour leurs propriétés théoriques intéressantes. En particulier, l'encodage d'invariance directement dans le réseau de neurones. Cette approche est complémentaire à l'augmentation de données que nous verront au chapitre 5. Dans le chapitre 4, nous définissons en détail le problème de robustesse, et le replaçons dans le contexte de la littérature plus vaste de la généralisation hors distribution.

Chapitre Quatre

Le problème de robustesse en apprentissage machine

Sommaire

4.1 Introduction : illustration du problème

Aujourd'hui, les réseaux de neurones profonds apprennent directement à représenter des données complexes dans des espaces de dimensions réduites. Cependant, ils sont fragiles aux petites variations sur les objets qu'ils doivent prédire. En effet, il suffit d'ajouter quelques pixels de bruit bien sélectionnés à une image de panda pour qu'un réseau de neurones change sa prédiction correcte de panda à une prédiction incorrecte de gibbon. On parle d'exemples adverses. La conception d'exemples spécifiquement dédiés à faire échouer les réseaux de neurones est un pan entier de recherche découvert récemment [Nguyen et al., 2015].

Pour voir comment les réseaux de neurones peuvent être victimes de l'hypothèse *i.i.d*, on peut faire l'expérience de pensée suivante [Arjovsky et al., 2020]. Un classifieur est entraîné à reconnaître des vaches et des chameaux, les photos de vaches étant souvent prises avec un fond d'herbe, et les photos de chameaux prises sur un fond de sable. Eh bien, il y a de fortes chances qu'un réseau de neurones prenne le raccourci d'associer les vaches à l'herbe et les chameaux au sable. Un tel raccourci ferait échouer ce classifieur pour une vache sur un fond de plage (voir figure 4.1). Cela est tout à fait cohérent, car l'apprentissage cherche le chemin de moindre résistance pour réussir la tâche sans se faire sanctionner par la fonction coût. Ainsi, réussir en exploitant des corrélations entre un objet et son environnement, a la même valeur qu'apprendre à distinguer les caractéristiques internes de l'objet. Cela pointe du doigt l'importance de la fonction coût pour l'apprentissage. Nous entrerons plus dans les détails des fonctions coûts dans notre section sur les statistiques robustes.

(a)

(b)

FIGURE 4.1: Illustration d'un exemple *i.i.d* et d'un exemple hors distribution : (a) une vache sur fond d'herbe [Samuels, 2016], (b) une vache sur fond de plage [belgianchocolate-FlickR, 2017]. Un classifieur entraîné à reconnaître des vaches sur un fond d'herbe et des chameaux sur un fond de sable associera un objet à son contexte. Ainsi, une vache sur fond de sable sera confondue à un chameau.

Dans la suite de ce chapitre, nous situons le problème de robustesse dans la littérature en suivant une approche en entonnoir. Nous commençons d'abord par présenter l'état de l'art lorsque l'on fait face au contexte $i.i.d$: l'apprentissage par transfert, l'adaptation de domaine et la généralisation hors domaine. Nous spécifions les hypothèses que font ces approches, et mettons en lumière les différences philosophiques qu'elles suggèrent pour un modélisateur. Enfin, nous voyons deux littératures centrées sur la notion de robustesse utile à notre problème : la robustesse algorithmique et la robustesse statistiques.

4.2 Apprendre hors contexte $i.i.d$:

Plusieurs types d'apprentissages ont émergé pour palier la contrainte de l'hypothèse $i.i.d$. Nous pouvons citer par exemple l'apprentissage multi-tâche [Caruana, 1997], le meta apprentissage [Vilalta and Drissi, 2002], ou l'apprentissage k-shot [Wang et al., 2019b]. Cependant par soucis de cohérence du formalisme décrit, nous nous concentrerons sur l'apprentissage par transfert [Pan and Yang, 2010], l'adaptation de domaine [Wang and Deng, 2018], et la généralisation hors domaine [Wang et al., 2021]. En effet, il est difficile de maintenir un formalisme cohérent pour un grand nombre de formes d'apprentissage ayant des hypothèses de départ et des objectifs différents. Néanmoins, nous encourageons le lecteur intéressé à se tourner vers la littérature citée pour plus d'information. La littérature citée est souvent une revue récente du domaine en question, et permet de mieux englober les avancées actuelles, et de les mettre en perspective avec les travaux fondateurs.

Pour bien comprendre la différence et les nuances entre ces formes d'apprentissages, on va appeler domaine un couple $\mathcal{D} = \{\mathcal{X}, \mathbb{P}_X\}$ ou $\mathcal{X}$ est l'espace des caractéristiques des données, et $\mathbb{P}_X$ est la loi de probabilité associée à cet espace. Ensuite, on définit une tâche qui sera notée par un couple $\mathcal{T} = \{\mathcal{Y}, \mathbb{P}_{Y|X}\}$, ou $\mathcal{Y}$ est l'espace des étiquettes, et $\mathbb{P}_{Y|X}$ la loi de probabilité conditionnelle prédisant la classe en fonction des caractéristiques.

Comme on accepte que les données de déploiement soient différentes des données d'apprentissage, on va avoir une source de données sur laquelle on fait nos apprentissages et des données cibles pour le déploiement. Ainsi, un couple domaine, tâche $(\mathcal{D}, \mathcal{T})$ permet de modéliser énormément de cas de figure. Ainsi, au départ, on peut avoir un domaine initial et une tâche initiale (aussi appelé source en anglais) $(\mathcal{D}_S, \mathcal{T}_S)$, mais la condition de déploiement peut différer avec un domaine cible et une tâche cible (aussi appelée "target" en anglais) $(\mathcal{D}_T, \mathcal{T}_T)$. À l'évidence, il y a plusieurs scénarios dans lesquels $(\mathcal{D}_S, \mathcal{T}_S) \neq (\mathcal{D}_T, \mathcal{T}_T)$, et ce sont ces différents scénarios qui vont donner lieu à des types d'apprentissages différents, chacun ayant sa propre raison d'être. Ainsi, le domaine source se décompose en espace des caractéristiques source et en distribution source $\mathcal{D}_S = \{\mathcal{X}_S, \mathbb{P}_X^S\}$. De même la tâche source se décompose en espace d'étiquettes source et en loi conditionnelle source $\mathcal{T}_S = \{\mathcal{Y}_S, \mathbb{P}_{Y|X}^S\}$.

Remarque 4.1. Notez que lorsque $\mathcal{X}_S \neq \mathcal{X}_T$ ou $\mathcal{Y}_S \neq \mathcal{Y}_T$, on suppose que les dimensions des espaces sont différentes. On suppose aussi qu'il n'y a pas de dimension vide. Alors nécessairement $\mathbb{P}_X^S \neq \mathbb{P}_X^T$, de même $\mathbb{P}_{Y|X}^S \neq \mathbb{P}_{Y|X}^T$. En d'autres termes changer l'espace des caractéristiques respectivement l'espace des étiquettes change nécessairement la distribution des caractéristiques respectivement la distribution conditionnelle des étiquettes. Nous avons représenté ça sur la figure 4.2 avec deux situations où le domaine varie (ce sont les colonnes.), et deux situations où les tâches varient (ce sont les lignes.).

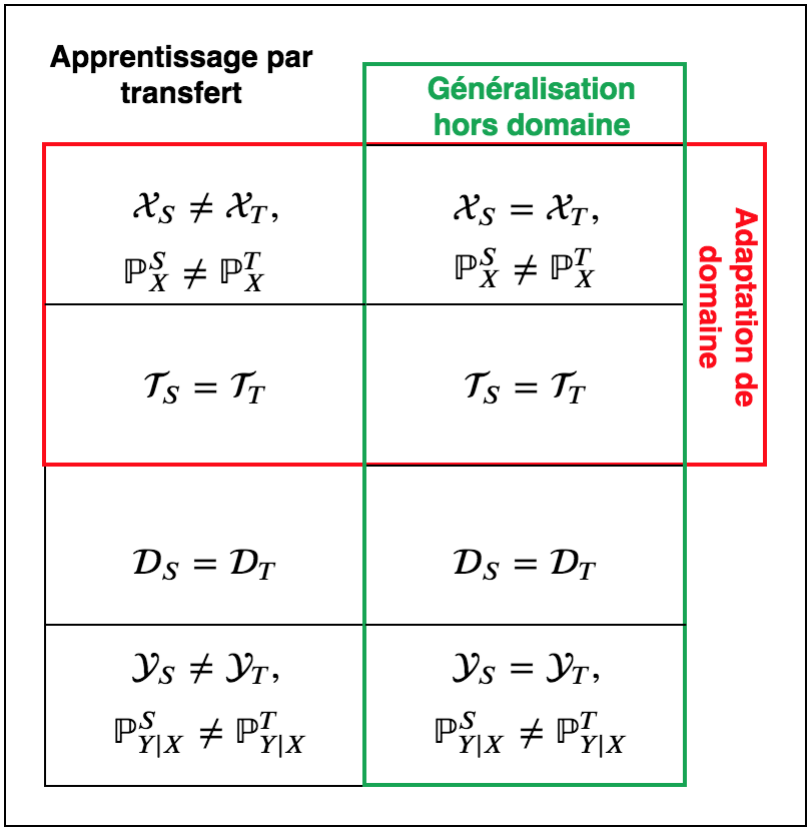


FIGURE 4.2: Tableau récapitulatif des différentes formes d'apprentissage hors contexte *i.i.d.* L'apprentissage par transfert se retrouve dans tous les quadrants de ce tableau. L'adaptation de domaine et la généralisation hors distribution sont des cas particuliers de l'apprentissage par transfert. Et ces deux dernières formes d'apprentissage partagent certaines conditions (Quadrant en haut à droite).

4.2.1 Apprentissage par transfert

Ainsi l'apprentissage par transfert [Pan and Yang, 2010] regroupe des situations avec des domaines différents $\mathcal{D}_S \neq \mathcal{D}_T$ ou des situations avec des tâches différentes $\mathcal{T}_S \neq \mathcal{T}_T$. Des domaines différents signifient que $\mathcal{X}_S \neq \mathcal{X}_T$ ou $\mathcal{X}_S = \mathcal{X}_T$ et $\mathbb{P}_X^S \neq \mathbb{P}_X^T$. Similairement, des tâches différentes signifient que $\mathcal{Y}_S \neq \mathcal{Y}_T$ ou $\mathcal{Y}_S = \mathcal{Y}_T$ et $\mathbb{P}_{Y|X}^S \neq \mathbb{P}_{Y|X}^T$.

Exemple 4.1. *Si $\mathcal{D}_S \neq \mathcal{D}_T$ avec $\mathcal{X}_S \neq \mathcal{X}_T$, et $\mathcal{T}_S \neq \mathcal{T}_T$ avec $\mathcal{Y}_S \neq \mathcal{Y}_T$, alors on est dans un cas où on a par exemple beaucoup de données initiales. Ainsi avec ImageNet [Deng et al., 2009] on a plus d'un million de photos de dimensions (256,256) et avec au total 1000 classes. Mais notre application d'intérêt concerne un petit jeu de données de 5000 images de radiographies du poumon prises au rayon X en haute dimension (1024,1024), et on veut détecter la présence de tumeur. Ici, la tâche de déploiement est une tâche de classification binaire. L'objectif du modélisateur ici est de tirer profit de la grande masse de données initiales pour apprendre des représentations généralistes, puis d'utiliser ces représentations généralistes (connaissance) pour apprendre plus facilement sur le petit jeu de données de radiographies du poumon.*

4.2.2 Adaptation de domaine

L'adaptation de domaine [Wang and Deng, 2018] est un sous-ensemble de l'apprentissage par transfert. Il concerne seulement les situations où les domaines sont différents à savoir $\mathcal{D}_S \neq \mathcal{D}_T$. C'est à dire les situations ou $\mathcal{X}_S \neq \mathcal{X}_T$ ou $\mathcal{X}_S = \mathcal{X}_T$ et $\mathbb{P}_{\mathcal{X}_S} \neq \mathbb{P}_{\mathcal{X}_T}$

Exemple 4.2. *Si $\mathcal{D}_S \neq \mathcal{D}_T$ avec $\mathcal{X}_S \neq \mathcal{X}_T$, mais $\mathcal{T}_S = \mathcal{T}_T$, les espaces de caractéristiques des données n'ont pas les mêmes dimensions, mais la tâche est la même. En vison par ordinateur, cela correspondrait à un cas où vous voulez avoir un classifieur binaire pour reconnaître les chiens et les chats. Vous l'entraînez avec des données standardisées en basse résolution collectées sur Internet. Mais vous voulez vous en servir sur votre téléphone ayant un appareil photo capturant des images en haute résolution.*

4.2.3 Généralisation hors distribution

L'une des premières références discutant de contexte en dehors de l'hypothèse *i.i.d* est Shimodaira [Shimodaira, 2000]) avec le terme de covariance décalée ("covariate shift"). D'autres auteurs on utilisé le terme de glissement de données [Quinonero-Candela et al., 2008, Moreno-Torres et al., 2012] ("dataset shift"). Aujourd'hui, on parle de généralisation hors distribution [Arjovsky, 2020] ("out of distribution generalization") ou ou de généralisation hors domaine ("domain generalization") [Zhou et al., 2021, Wang et al., 2021]. De manière générale ce sont les situations ou les données initiales et les données cibles sont générées par des lois de probabilité différentes i.e $\mathbb{P}^S_{XY} \neq \mathbb{P}^T_{XY}$.

Remarque 4.2. Ici, on préfère le terme de généralisation hors distribution, car cela permet de distinguer la distribution du domaine tel que discuté en apprentissage par transfert et en adaptation de domaine. Par ailleurs, on utilisera les termes de glissement de données ou de glissement de distribution de manière interchangeable. On utilise aussi de manière interchangeable les termes de loi de probabilité et de distribution de probabilité ("probability distribution"), le second terme étant très utilisé dans la littérature anglophone.

Nous pouvons toujours factoriser une distribution de données conjointe comme un produit d'une distribution conditionnelle et d'une distribution marginale $\mathbb{P}_{XY} = \mathbb{P}_X \mathbb{P}_{Y|X}$. Cela permet de définir des cas particuliers de glissement de données :

— glissement covarié : $\mathbb{P}^S_{Y|X} = \mathbb{P}^T_{Y|X}$ mais $\mathbb{P}^S_X \neq \mathbb{P}^T_X$. Ce cas se trouve dans le quadrant en haut à droite de la figure 4.2. C'est aussi un problème d'adaptation de domaine.

— glissement de concept : $\mathbb{P}^S_{Y|X} \neq \mathbb{P}^T_{Y|X}$ mais $\mathbb{P}^S_X = \mathbb{P}^T_X$. Ce cas se trouve dans le quadrant en bas à droite de la figure 4.2.

— glissement général de l'ensemble de données : $\mathbb{P}^S_{XY} \neq \mathbb{P}^T_{XY}$, mais hors glissement covarié et glissement de concept.

Selon les auteurs, la dernière forme de décalage est rare et difficile à résoudre.

Dans cette thèse, nous allons nous concentrer principalement sur le glissement covarié, car il inclut de nombreux problèmes non résolus en apprentissage machine. Mais nous allons l'affiner un peu. En effet, selon Moreno-Torres (2012), le glissement covarié peut être causé principalement par deux facteurs : le biais de sélection des données et les domaines non-stationnaires. D'une part, le biais de sélection des données est lié à un ensemble

d'apprentissage échantillonné de manière non-uniforme à partir de la source de données. D'autre part, les domaines non-stationnaires induisent des glissements liés à des conditions spatiales ou temporelles différentes ou à des adversaires.

Nous avons travaillé avec ces deux cas de figure au chapitre 6. Usuellement le bias de sélection des données cause un déséquilibre de classe, comme cela est mentionné dans la littérature [Moreno-Torres et al., 2012]. En effet le bias d'échantillonnage se produit lorsque le procédé de collecte des données est variable pour deux jeux de données, c'est ce que nous avons observé dans le cadre des données de campagnes de 2011 et 2015. En effet, les deux itinéraires n'étaient pas exactement les mêmes, ce qui a modifié la loi conditionnelle.

Par ailleurs, au chapitre 6 le changement de domaine s'est produit avec un changement de paramétrage de l'appareil de mesure, dans notre cas : l'echosondeur. Cela a causé un changement dans la structure du bruit dans les données. De manière générale, la non-stationnarité du domaine se produit en vision par ordinateur lorsque des objets sont photographiés à des saisons différentes. Cela s'explique par des changements dans les reflets de luminosité sur les objets avec le changement de saison.

4.2.4 Différences philosophiques

On a vu que ces trois formes d'apprentissage sont entremêlées. Et le lecteur peut trouver les différences que nous avons tracées en figure 4.2 arbitraires. Dans ce paragraphe, nous allons davantage discuter des différences entre l'apprentissage par transfert l'adaptation de domaine et la généralisation hors distribution.

En quelques mots, on pourrait encore les séparer en deux groupes, avec d'une part l'apprentissage par transfert et l'adaptation de domaine et d'autre part la généralisation hors distribution.

Adaptation de domaine et apprentissage par transfert

La principale différence étant que dans le premier groupe, on suppose que le modélisateur a déjà collecté un jeu de données pour simuler la tâche de déploiement. Nous voyons par ailleurs trois cas de figure dans cette situation :

— adaptation de domaine ou apprentissage par transfert supervisé : le modélisateur a collecté un petit nombre d'exemples provenant de la distribution de déploiement et les a étiquetés.

— Adaptation de domaine ou apprentissage par transfert semi-supervisé : le modélisateur a collecté un vaste nombre d'exemples provenant de la distribution de déploiement, et en a étiqueté une petite quantité.

— Adaptation de domaine ou apprentissage par transfert non supervisé : le modélisateur a collecté un vaste nombres d'exemples provenant de la distribution de déploiement, mais n'en a étiqueté aucun.

Dans le cas supervisé, on suppose que la collecte d'exemples est difficile ou coûteuse, et l'étiquetage est marginalement coûteux. Dans le cas semi-supervisé, la collecte d'exemples est facile, l'étiquetage est coûteux et le modélisateur a les moyens d'étiqueter un échantillon des données collectées. Dans le cas non supervisé la collecte est facile, et l'étiquetage est hors de portée.

Généralisation hors distribution et robustesse

Dans un problème de généralisation hors distribution, le modélisateur n'a pas collecté de données pour la tâche de déploiement. Il suppose que les données de déploiement ne sont que marginalement différentes des données d'apprentissage et cherche un modèle conservant de bonnes performances même en cas de glissement des données. La tâche de généralisation hors distribution peut être développée en contexte mono-source ou en contexte multi-source. En contexte, multi-source le modélisateur a accès à n sources de données étiquetées : $L^1, \cdots, L^n$ et tente de généraliser à une source de données non étiquetées U^{n+1} inconnue. La plupart des méthodes tentent d'utiliser les informations dans les sources de données pour trouver des invariances pouvant être utiles sur la distribution inconnue, voir [Zhou et al., 2021, Wang et al., 2021] pour une revue de littérature complète sur la théorie, les algorithmes et les applications. Dans cette thèse, nous nous intéressons plus particulièrement au sous-problème le plus difficile de la généralisation hors distribution, lorsque la source d'apprentissage est unique. Dans ce cas, on a accès à un jeu de données étiqueté L_1 et on souhaite généraliser à un autre jeu de données non étiqueté U_2 proche du jeu d'origine. Suivant la littérature, les auteurs parlent de généralisation hors distribution mono-source [Wang et al., 2021] ou d'apprentissage robuste [Xu and Mannor, 2012] en opposition à l'apprentissage hors distribution multi-source . Dans la suite, nous approfondissons la notion d'apprentissage robuste.

4.3 Robustesse

Le mot robustesse est utilisé dans de nombreux contextes, en apprentissage, Xu et Mannor introduisent la notion de robustesse algorithmique qui est la plus sous-entendue en apprentissage machine. Mais nous avons aussi voulu savoir ce que les statisticiens entendaient avec la notion de robustesse. Nous voulons noter ici, que le mot robustesse a été utilisé dans de nombreuses formes d'apprentissage que nous ne discuterons pas ici, mais nous pouvons renvoyer le lecteur intéressé vers plus de littérature. Par exemple, notons l'optimisation robuste [Ben-Tal et al., 2009, Gao et al., 2017] qui va modéliser le problème d'apprentissage avec une approche mini-max. Dans ce paradigme, on va minimiser l'erreur moyenne sur la distribution perturbant au maximum la source des données en restant à proximité de la distribution source. En pratique, la recherche de la pire distribution se fait dans une boule restreignant la divergence entre la distribution source et la pire distribution candidate. Par exemple, Gao et al. [Gao et al., 2017] utilisent une divergence de Wasserstein pour spécifier la boule autour de la distribution source. Au final, cette approche est équivalente à une méthode de régularisation. Dans la suite, cependant, nous nous concentrons sur la robustesse algorithmique et les statistiques robustes. La différence principale est que la robustesse algorithmique qualifie une propriété d'un algorithme, que ses prédictions soient lisses au voisinage des éléments de l'échantillon d'apprentissage. Là où les statistiques robustes cherchent un estimateur peu sensible aux valeurs aberrantes par le choix de fonctions coûts appropriés. Avec le recul, on peut voir que ces deux approches ne font pas les mêmes hypothèses avant et après l'apprentissage. En effet, dans le cas des statistiques robustes, le modélisateur suppose qu'il a accès à des données avec du bruit sur les étiquettes, et le challenge est de réussir à ce que ces données ne perturbent

pas l'apprentissage. Alors que dans le cas de la robustesse algorithmique, le modélisateur suppose qu'il a accès à des données sans bruit sur les étiquettes durant l'apprentissage, mais s'attend à ce qu'il y ait du bruit sur l'espace des données. Donc son challenge est de trouver un modèle insensible à du bruit sur l'espace des données.

4.4 Robustesse algorithmique

4.4.1 Robustesse et pseudo robustesse algorithmique

La robustesse algorithmique se produit lorsqu'un modèle ne change pas beaucoup sa prédiction suite à un léger changement dans les données fournies [Xu and Mannor, 2012]. En d'autres termes, si un exemple de test est proche d'un exemple d'entraînement, leurs pertes doivent être proches pour un algorithme robuste. Dans la suite, on considère que les jeux de données sont fixes de taille n.

Un algorithme A est une fonction définie d'un espace d'échantillons $\mathcal{Z}^n$ à valeur dans un espace d'hypothèses $\mathcal{H}$ en d'autres termes $A : \mathcal{Z}^n \to \mathcal{H}$. Donc si $S \in Z^n$ est un ensemble de données, A_S est une fonction d'hypothèse h. Et donc si $z \in S$, $A_S(z)$ est une prédiction. On note $l(A_S, z)$ le coût du point z pour l'hypothèse A_S. Nous supposons que la fonction coût considérée est positive et borné par M.

Définition 4.1. Un algorithme A est $(K, \varepsilon(\cdot))$ robuste, si on peut partitionner l'espace échantillon $\mathcal{Z}$ en K sous-ensembles disjoints $\{C_i\}_{i=1}^K$, et $\forall S \in \mathcal{Z}$, $s \in S$, $z \in \mathcal{Z}$:

$$z, s \in C_i \Rightarrow |l(A_S(z)) - l(A_S(s))| \leq \varepsilon(S)$$

Avec $\varepsilon : \mathcal{Z}^n \to \mathbb{R}$ la fonction bornant le coût spécifique au jeu de données.

Cette définition est une propriété de l'algorithme A, du jeu de donnée S et de la fonction coût. En effet, elle qualifie des petits changements de coûts associés à l'hypothèse A_S pour des points proches (dans le sens qu'ils appartiennent au même sous ensemble local). En d'autres termes, si l'espace d'échantillonnage peut être partitionné de manière à ce que pour n'importe quel ensemble de données, l'algorithme ait un coût très similaire pour des exemples proches les uns des autres, alors il est robuste. Cette propriété est spécifique au jeu de données, car les bornes varieront d'un jeu de données à un autre. En effet, la K-partition restera la même, mais des échantillons S et T pourraient avoir des exemples plus ou moins proches des lignes de partitionnement et entraîner des bornes différentes $\varepsilon(S)$ et $\varepsilon(T)$. Cette propriété est aussi associée à la fonction coût, car des fonctions coûts différentes entraîneront des bornes différentes, cela va de soit.

Cette propriété de robustesse ressemble beaucoup à la notion de continuité des fonctions à valeurs réelles. En effet, pour rappel une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est continue en $x \in \mathbb{R}^n$ si $\forall \epsilon \in \mathbb{R}$ $\exists \delta > 0$, $y \in B(x, \delta) \Rightarrow f(y) \in B(f(x), \epsilon)$. Et elle est continue sur $\mathbb{R}^n$ si elle est continue en tout point $x \in \mathbb{R}^n$.

La similarité, c'est qu'ici la continuité en question concerne la fonction coût, et l'algorithme trouvant une hypothèse pour tout jeu de données. En effet, un algorithme robuste apprendra quel que soit le jeu de données une hypothèse ayant un coût continue sur les points de l'espace.

Telle que décrite, cette propriété n'est pas dépendante d'une distribution générant des données, mais s'applique à tout l'espace $\mathcal{Z}$. Pour l'opérationaliser, on considérera que $\mathcal{Z}$ est un compact de sorte à ce qu'il puisse être borné. Par ailleurs en l'état, cette propriété de robustesse doit être vérifiée sur tous les points s du jeux de données S. C'est une contrainte beaucoup trop forte. En effet, prenons le contexte de la classification par exemple. Un algorithme pourrait être robuste sur toutes ses prédictions correctes. Mais il ne serait pas robuste sur les exemples incorrectement classifiés. Car le coût serait élevé pour un point mal classifié, et serait donc nécessairement très différent des points proches, mais bien classifiés. Ainsi Xu et al. proposèrent l'idée de pseudo-robustesse qui est plus proche de la réalité.

Définition 4.2. Un algorithme A est $(K, \varepsilon(\cdot), \hat{n}(\cdot))$ pseudo robuste, si on peut partitionner l'espace échantillon $\mathcal{Z}$ en K sous-ensembles disjoints $\{C_i\}_{i=1}^{K}$, et $\forall S \in \mathcal{Z}, \exists \hat{S} \subset S$, tel que $\forall s \in \hat{S}$, $z \in \mathcal{Z}$:

$$z, s \in C_i \Rightarrow |l(A_S(z)) - l(A_S(s))| \leq \varepsilon(S)$$

Avec $\varepsilon : \mathcal{Z}^n \to \mathbb{R}$ la fonction bornant le coût spécifique au jeu de données, et $\hat{n} : S \mapsto |\hat{S}|$ le nombre d'exemples dans $\hat{S}$.

Remarquez que si un algorithme est (K, ε, n) pseudo-robuste quel que soit le jeu de données S, alors il est (K, ε) robuste. Cette définition garantit donc qu'un algorithme A sera au moins $K, \varepsilon(\cdot)$ robuste sur un sous-échantillon quel que soit le jeu de données S considéré.

4.4.2 Erreur de généralisation d'un algorithme robuste et pseudo robuste

Théorème 4.1. *Si S est un échantillon de n points i.i.d suivant une loi inconnue $\mathbb{P}$, et A est $(K, \varepsilon(\cdot))$ robuste, alors pour tout $\delta > 0$ avec une probabilité d'au moins $1 - \delta$*

$$|\mathcal{R}_S(A_S) - \mathcal{R}_{\mathbb{P}}(A_S)| \leq \varepsilon(S) + M\sqrt{\frac{2K\ln(2) + 2\ln(\frac{1}{\delta})}{n}}$$

Cela indique que l'erreur de généralisation dépend de M la borne de la fonction coût, de $\varepsilon(S)$ la borne des points qui sont sur la même partition. Par ailleurs, cette quantité diminue avec n qui augmente. Et cette quantité reste sous ces bornes avec une probabilité arbitraire de $1 - \delta$. Ceci est une garantie sur le comportement d'un algorithme robuste. En effet, la performance de l'algorithme robuste lors du déploiement $\mathcal{R}_{\mathbb{P}}(A_S)$ sera borné.

Voyons maintenant l'erreur de généralisation pour un algorithme pseudo-robuste :

Théorème 4.2. *Si S est un échantillon de n points i.i.d suivant une loi inconnue $\mathbb{P}$, et A est $(K, \varepsilon(\cdot), \hat{n}(\cdot))$ pseudo-robuste, alors pour tout $\delta > 0$ avec une probabilité d'au moins $1 - \delta$*

$$|\mathcal{R}_S(A_S) - \mathcal{R}_{\mathbb{P}}(A_S)| \leq \frac{\hat{n}(S)}{n}\varepsilon(S) + M\left(\frac{n - \hat{n}(S)}{n} + \sqrt{\frac{2K\ln(2) + 2\ln(\frac{1}{\delta})}{n}}\right)$$

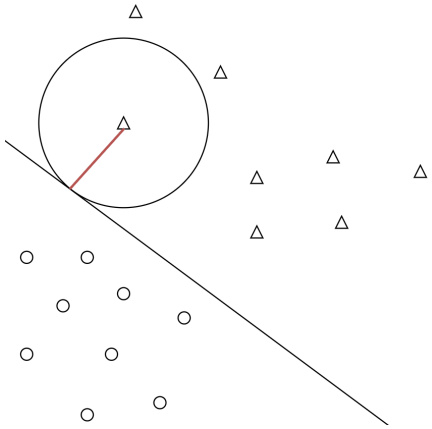


FIGURE 4.3: En rouge la distance correspondant à la marge du triangle (au centre de la boule) pour le classifieur classifiant les triangles et les ronds

Nous avons introduit ce formalisme pour pouvoir apprécier l'effet de l'invariance sur la généralisation d'algorithmes de classification pseudo-robustes. En effet, dans la pratique, un algorithme de classification robuste aura toujours une généralisation parfaite. Mais cela n'existe pas, donc la notion de robustesse telle que décrite plus haut ne nous donnera pas de compréhension plus profonde sur ce que nous cherchons en réalité, un algorithme robuste la plupart du temps. Et dans une moindre mesure, un algorithme robuste face aux changements de distributions.

Mais pour saisir les effets de l'invariance sur la généralisation, nous aurons besoin de définir la marge d'un classifieur et d'un algorithme.

4.4.3 Erreur de généralisation d'un algorithme pseudo robuste à large marge

Définition 4.3. Soit $s^{(i)} = (x^{(i)}, y^{(i)})$ un point de $\mathcal{X} \times \mathcal{Y}$, $g : \mathcal{X} \to \mathcal{Y}$ un classifieur et d une distance définie sur $\mathcal{X}$. On appelle marge de $s^{(i)}$ par rapport à g la quantité suivante que nous visualisons en figure 4.3 :

$$\gamma_d(s^{(i)}, g) = \inf_{a \in \mathbb{R}} \{d(x^{(i)}, x) \leq a, \text{avec } g(x) \neq y^{(i)}\}$$

La marge d'un exemple par rapport à un classifieur est la plus petite distance à la ligne de décision de g.

Un classifeur à large marge est un classifieur qui trace des lignes de décisions de sorte que les exemples issus du jeu de données soient concentrés au sein des partitions correspondant aux régions de décisions. Un algorithme robuste à large marge classe nécessairement bien tous les exemples considérés. Mais comme cela n'arrive jamais en pratique, on va considérer les algorithmes pseudo-robustes à large marge, ces algorithmes sont très communs, en effet, un classifeur peut correctement classifieur un sous-ensemble de son jeu d'apprentissage, et il y aura forcément une marge γ. Donc, en considérant ces algorithmes, on peut obtenir une borne de généralisation s'appliquant à un très grand nombre de cas.

Considérons un algorithme A, une marge arbitraire $\gamma > 0$. Et considérons la fonction

suivante

$$\hat{n} : S \mapsto \sum_{i=1}^{n} \mathbb{1}[\gamma_d(s^{(i)}, A_S) > \gamma]$$

Cette fonction va associer à chaque échantillon S le nombre d'exemples bien classifiés ayant une distance plus grande que γ à la ligne de décision apprise par A sur S. Ainsi cet algorithme A est $(N_{\mathcal{Y}} \cdot \mathcal{N}(\mathcal{X}, d, \frac{\gamma}{2}), 0, \hat{n})$ pseudo robuste pour la fonction coût l_{0-1} qui est borné par 1. Ici $N_{\mathcal{Y}}$ est le nombre de classes et $\mathcal{N}(\mathcal{X}, d, \frac{\gamma}{2})$ est le nombre couvrant de l'espace des caractéristiques $\mathcal{X}$ [Xu and Mannor, 2012]. Ainsi, l'erreur de généralisation d'un tel algorithme est donnée par le résultat suivant :

Théorème 4.3. *Si S est un échantillon de n points i.i.d suivant une loi inconnue $\mathbb{P}$, et A un algorithme $(N_{\mathcal{Y}} \cdot \mathcal{N}(\mathcal{X}, d, \frac{\gamma}{2}), 0, \hat{n})$ pseudo-robuste, alors pour tout $\delta > 0$ avec une probabilité d'au moins $1 - \delta$*

$$|\mathcal{R}_S(A_S) - \mathcal{R}_{\mathbb{P}}(A_S)| \leq \frac{n - \hat{n}(S)}{n} + \sqrt{\frac{2\ln(2) \cdot N_{\mathcal{Y}} \cdot \mathcal{N}(\mathcal{X}, d, \frac{\gamma}{2}) + 2\ln(\frac{1}{\delta})}{n}}$$

4.4.4 Erreur de généralisation d'un algorithme pseudo-robuste à large marge et invariant

Sokolik [Sokolic et al., 2017b] avait étudié les réseaux de neurones profonds à large marge en s'inspirant fortement des travaux de Xu [Xu and Mannor, 2012], puis a étendu sa théorie aux classifieurs invariants robustes [Sokolic et al., 2017a]. Cependant, une des limitations de Sokolik et ses collègues est de travailler avec des algorithmes robustes. Donc les bornes qu'ils développent [Sokolic et al., 2017b, Sokolic et al., 2017a] sont vrais pour des algorithmes classifiant correctement tous les exemples de tout jeu de données. Or, dans la pratique, cela n'arrive jamais. Cependant, son approche est très intéressante, et nous allons l'utiliser pour décliner des bornes pour un algorithme pseudo-robuste à large marge et invariant.

On peut imaginer un ensemble des perturbations comme un ensemble de fonctions $\mathcal{T} = \{t_1, \cdots, t_n\}$ pouvant être appliquées à l'espace des caractéristiques. Ces perturbations sont des fonctions de bruits $t : \mathbb{R}^d \to \mathbb{R}^d$, par exemple un changement de luminosité ou de contraste dans le contexte des images. Un classifieur g est invariant à $\mathcal{T}$ si $\forall x \in \mathcal{X}, g(t_i(x)) = g(t_j(x)$. En d'autres termes, g est un classifieur dont les prédictions sont insensibles aux perturbations issus de $\mathcal{T}$. Un algorithme invariant à $\mathcal{T}$ que nous notons $A^{\mathcal{T}}$ est un algorithme qui apprendra un classifieur invariant à toutes les perturbations issues de $\mathcal{T}$ quel que soit le jeu de données $S \in \mathbb{Z}^n$

Nous voulons apprécier l'avantage d'avoir un algorithme invariant sur l'erreur de généralisation. Pour ce faire, admettons qu'il existe une factorisation de l'espace des caractéristiques $\mathcal{X} = \{t(x), t \in \mathcal{T}, x \in \mathcal{X}_0\}$. En d'autres termes $\mathcal{X}_0$ est l'espace des caractéristiques de base dépouillé de toutes les corruptions issues du procédé de collecte des données. Cet espace de base est nécessairement plus simple que l'espace $\mathcal{X}$. Ainsi un algorithme invariant $A^{\mathcal{T}}$ aura une erreur de généralisation indexée sur cet espace de base. C'est ce que nous apprend [Sokolic et al., 2017a]. Ainsi, on obtiendrait les bornes suivantes :

Théorème 4.4. *Si S est un échantillon de n points i.i.d suivant une loi inconnue $\mathbb{P}$, et A un algorithme $(N_y \cdot \mathcal{N}(\mathcal{X}, d, \frac{\gamma}{2}), 0, \hat{n})$ pseudo robuste et invariant à $\mathcal{T}$, alors pour tout $\delta > 0$ avec une probabilité d'au moins $1 - \delta$*

$$|\mathcal{R}_S(A_S) - \mathcal{R}_{\mathbb{P}}(A_S)| \leq \frac{n - \hat{n}(S)}{n} + \sqrt{\frac{2\ln(2) \cdot N_y \cdot \mathcal{N}(\mathcal{X}_0, d, \frac{\gamma}{2}) + 2\ln(\frac{1}{\delta})}{n}}$$

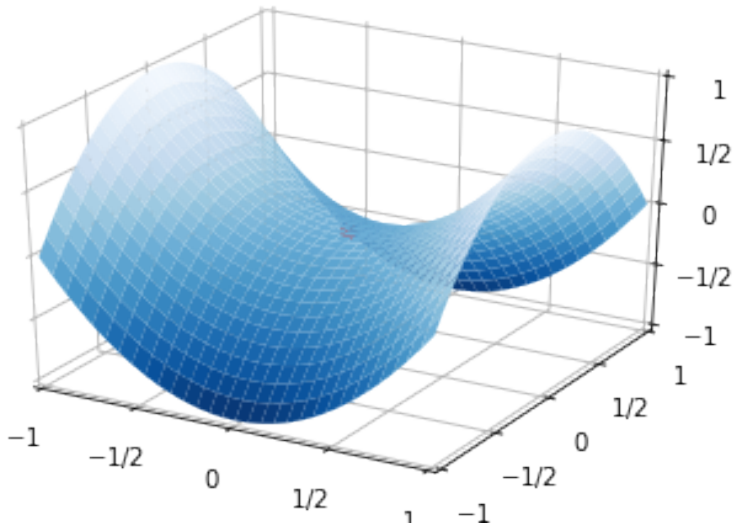
On voit que l'erreur de généralisation d'un algorithme pseudo-robuste à large marge invariant diminue en fonction de la factorisation de l'espace. La variation est donnée par le ratio :

$$R(\mathcal{X}_0, \mathcal{X}, d, \frac{\gamma}{2}) = \sqrt{\frac{\mathcal{N}(\mathcal{X}_0, d, \frac{\gamma}{2})}{\mathcal{N}(\mathcal{X}, d, \frac{\gamma}{2})}}$$

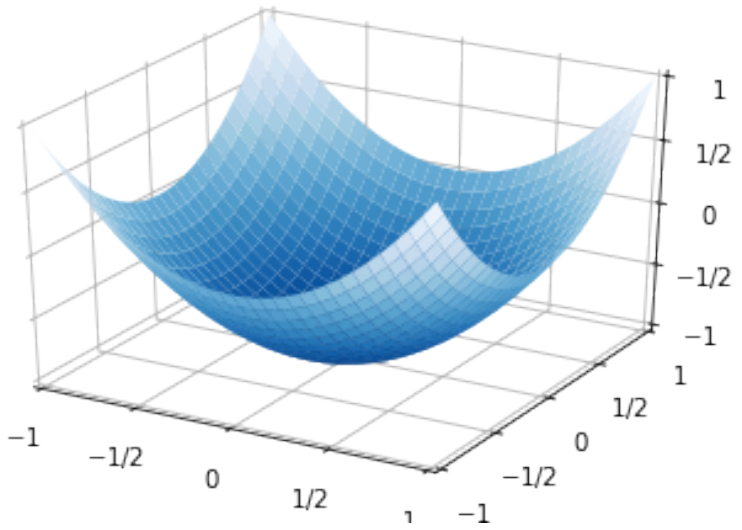
Lorsque ce ratio est beaucoup plus petit que 1, cela indique que la factorisation de l'espace à permis de fortement réduire l'erreur de généralisation. Cependant, une observation intéressante serait de raisonner dans l'autre sens. Que se passerait-il si A était fragile face au groupe des perturbations $\mathcal{T}$, alors à la place d'une factorisation de l'espace, on aurait une multiplication de l'espace et un ratio supérieur à 1. C'est un peu ce qui se passe lors d'un glissement de données. La conclusion de cette étude est que les perturbations auxquelles on veut être invariant doivent appartenir au jeu de données de déploiement. Si on est fragile face à des perturbations inconnues, cela va entraîner un ratio positif et une dégradation de l'erreur de généralisation. Mais quelle est la différence entre un algorithme robuste et un algorithme robuste invariant à $\mathcal{T}$, ou seulement invariant à t_1, mais pas à t_2 ? C'est ce que nous voyons dans la suite.

4.4.5 Taux de variation de la fonction coût, robustesse et invariance

Nous avons vu qu'un algorithme est robuste lorsqu'une petite perturbation des données ne change pas ses prédictions. Mais un algorithme est toujours invariant à des perturbations spécifiques. Ainsi, un algorithme peut être invariant à une transformation t_1, mais pas invariant à une transformation t_2. De même, un algorithme robuste, sera invariant à toute transformation des données à supposer que ces transformations ne changent que légèrement les données. On peut visualiser les idées de robustesse et d'invariance avec des intuitions issues de la physique. En effet, considérons les perturbations $t_1 : x \mapsto x + \epsilon_1$ et $t_2 : x \mapsto x + \epsilon_2$ que nous appliquons à un exemple en haute dimension $x \in \mathcal{X}$. On considère que les transformations sont toutes deux égales en intensité et très légère $||\epsilon_1|| = ||\epsilon_2|| \ll 1$, mais que leurs directions sont différentes $\epsilon_1 \neq \epsilon_2$. On peut interpréter ces transformations comme des vecteurs force qui sont appliqués sur les données. Ces forces, de direction différentes vont pousser l'exemple x respectivement dans les directions ϵ_1 et ϵ_2 dans l'espace des données. Cependant, l'effet de cette perturbation sur le taux de variation de la fonction coût peut être soit faible, soit très important suivant que l'algorithme sous-jacent soit invariant ou non aux perturbations. Considérons que notre algorithme soit invariant à t_1, mais pas invariant à t_2, alors, on peut visualiser l'effet de ces perturbations sur la ligne de décision avec la figure 9.1 a. En effet, cette figure représente un point selle, avec un point au milieu. On peut voir notre exemple x comme le point au milieu, par ailleurs, on peut imaginer que



(a)



(b)

FIGURE 4.4: (a) Taux de variation de la fonction coût d'un modèle invariant à une transformation de données t_1 (direction ascendante) et fragile à une transformation de données t_2 (direction descendante).(b) Taux de variation de la fonction coût d'un modèle robuste. Aucune perturbation locale ne peut déplacer le point hors de sa région de décision.

la transformation t_1 va pousser le point dans la direction ascendante et la transformation t_2 va pousser le point dans la direction descendante. La transformation t_1 va pousser le point vers le haut, puis le point va osciller, et reviendra à sa position initiale. La prédiction n'a pas changé après application de t_1. Voilà une intuition physique de l'invariance à une transformation. Par contre en appliquant la transformation t_2, le point va tomber et sa nouvelle position d'équilibre sera très différente de la position initiale. Par ailleurs, si le point est sorti de la région de décision correspondant à la classe initiale, alors on considérera que l'algorithme est fragile à la transformation t_2. Enfin, si un algorithme est robuste, alors les points sont en situation d'équilibre stable (voir figure 9.1 b). En effet, dans ce cas, si le point x est au milieu de la cuvette, alors peu importe la direction de la force appliquée, à supposer qu'elle soit de faible intensité, alors le point reviendra à la position initiale. Nous reviendrons sur cette intuition en conclusion pour discuter des effets de l'invariance sur la généralisation hors distribution.

4.5 Robustesse statistiques

Dans la suite de ce chapitre, nous discutons des statistiques robustes qui cherchent à apprendre un modèle approximativement correct, à la différence des statistiques paramétriques qui supposent que le modèle est correct. Nous discuterons des scores robustes, qui sont en réalité des fonctions coûts bornés ayant une moindre pénalisation des anomalies que la log-vraisemblance que nous avions déjà étudié au chapitre 2.

Les statistiques robustes sont un vaste corpus théorique centré sur l'estimation en présence d'anomalies. Cela se fait par l'approximation de modèles, c'est-à-dire que les statistiques robustes ne supposent pas que le modèle est bien spécifié. En outre, le but des statistiques robustes est la recherche d'un voisinage de modèles qui approximent les données correctement sans donner trop de poids aux anomalies. Cela diffère de l'approche des statistiques paramétrique qui supposent que le modèle est bien spécifié, c'est-à-dire que la modélisation admet un paramètre capable de générer exactement les données. Pour rester synthétique, le problème de robustesse étudié par les statisticiens a commencé avec Huber [Huber, 1964] et son modèle de contamination $P = (\epsilon)P_0 + (1 - \epsilon)W$ avec P_0 la distribution réelle et W une distribution parasite inconnue. Et le but des statistiques robuste est d'apprendre à ignorer la distribution parasite. Donc à supposer que le modèle n'est qu'une approximation des données, les estimateurs robustes recherche la meilleure approximation en recherchant trois propriétés selon Huber [Huber and Ronchetti, 2009] :

1. L'efficacité : l'estimateur robuste doit trouver un modèle de faible variance.
2. La stabilité : une contamination des données, ne devrait pas trop affecter la variance asymptotique. C'est-à-dire, qu'en contexte de mégadonnées mais avec des anomalies, différents modèles estimés pour la même procédure ne devrait pas avoir des performances trop différentes.
3. Protection contre les ruptures : si les données sont très contaminées, les modèles estimés ne devraient pas avoir de perte de performance catastrophique.

D'après Hampel [Hampel, 2005] les approches principales sont (1) les M-estimateurs qui généralisent les estimateurs du maximum de vraisemblance (2) l'approche par les tests robustes (3) l'approche par les fonctions d'influence ou l'approche infinitésimale. Dans ce

manuscrit, nous avons considéré un sous-ensemble des M-estimateurs, les scores robustes. Mais avant de les décrire, revenons sur l'estimateur du maximum de vraisemblance, pour mieux contraster le log-score des scores robustes.

4.5.1 Estimateur du maximum de vraisemblance

L'approche la plus répandue en apprentissage machine consiste à utiliser le log-score que nous avons vu dans l'introduction. Cette fonction est issue de la procédure de maximisation de la vraisemblance. Dans le cas de classification binaire, il s'agissait de la fonction convexe suivante :

$$l(x, y, h_w) = -(y \cdot \log(h_w(x)) + (1 - y) \cdot \log(1 - h_w(x)))$$

Définition 4.4. Un score est une fonction des probabilités et de l'événement survenu. Ici $\mathcal{P} = \{(p_1, \cdots, p_k) \in [0, 1]^k, \sum_{i=1}^{k} p_i = 1\}$ est le simplexe de probabilité, l'espace de tous les vecteurs de probabilités.

$$S : \mathcal{Y} \times \mathcal{P} \to \mathbb{R}^+ \tag{4.1}$$
$$: (y, p) \mapsto S(y, p) \tag{4.2}$$

Ainsi, la version plus générale du log-score pour une prédiction sous forme de vecteur est la suivante $S(p, y) = -\log(p_y)$ ou p_y est la prédiction à la position de la classe y. Dans le contexte de la classification multiclasse, un score est une fonction coût, on peut utiliser les termes de manière interchangeable. Cependant, les scores forment un domaine d'étude commun à de nombreuses disciplines intéressées par l'évaluation des modèles prédictifs probabilistes comme en météorologie, statistiques ou psychologie [Winkler et al., 1996].

Définition 4.5. La vraisemblance d'un jeu de données $S = \{(x^{(1)}, y^{(1)}), \cdots (x^{(n)}, y^{(n)})\}$ *i.i.d* pour un modèle statistique paramétrique m_w est la fonction suivante :

$$\mathcal{L}(y|x^{(1)}, \cdots, x^{(n)}; w) = \prod_{i=1}^{n} m(y|x^{(i)}), w)$$

La vraisemblance des données est la probabilité de l'échantillon étant donné le modèle statistique de paramètre w. En d'autres termes, c'est le niveau de crédibilité des données en fonction de notre modélisation de la loi conditionnelle par le modèle m_w. Il y a ici deux points essentiels de l'estimation paramétrique venant de statistiques et qui ne sont pas remis en question en apprentissage machine :

— L'hypothèse que les données sont *i.i.d* suivant une distribution inconnue.

— La loi conditionnelle puisse être décrite exactement par le modèle paramétrique m. En d'autres termes qu'il existe un paramètre $\hat{w}$ dans l'espace des paramètres tel que la modélisation de la fonction de vraisemblance puisse générer les données, i.e $\mathcal{L}(y|x_1, \cdots, x_n; \hat{w}) = 1$. Si c'est le cas, on dit que le modèle est bien spécifié.

Si les données sont *i.i.d* et que l'hypothèse de spécification est correcte, alors il nous suffit juste de chercher le paramètre $\hat{w}$ maximisant la fonction de vraisemblance. Ce qui est

équivalent à minimiser la négative log-vraisemblance, (la fonction -log est convexe). Ainsi :

$$\max_{w\in\mathbb{R}^d} \mathcal{L}(y|x^{(1)},\cdots,x^{(n)};w) = -\sum_{i=1}^{n} -\log(\mathcal{L}(y|x^{(i)};w)) \tag{4.3}$$

$$= \min_{w\in\mathbb{R}^d} \sum_{i=1}^{n} -\log(m(y=y^{(i)}|x^{(i)},w)) \tag{4.4}$$

Un estimateur en statistique est un algorithme d'apprentissage en apprentissage machine. C'est-à-dire, une fonction définie sur un espace d'échantillons à valeur dans un espace de fonctions. L'estimateur du maximum de vraisemblance est la principale méthode utilisée en apprentissage machine, car il présente d'excellentes propriétés asymptotiques. En effet, il est asymptotiquement normal, sans biais et efficace. Ce qui signifie que l'estimateur du maximum de vraisemblance est certain de trouver le vrai paramètre $\hat{w}$ à mesure qu'on collecte plus de données, sa variance sera minimale et normalement distribuée. En termes plus pratiques, ces propriétés émergent en situation de mégadonnées.

Cependant, si notre hypothèse que le modèle est bien spécifié est fausse. Alors, cela aura des conséquences importantes sur l'apprentissage. En effet, lorsque le modèle est bien spécifié, on suppose qu'on modélise parfaitement la loi générant les données. On ne suppose pas qu'on fait une approximation. Cependant, dans la plupart des cas, en apprentissage profond, on suppose que le modèle est une excellente approximation des données tout au plus, mais pas une modélisation parfaite des données. Cette distinction, bien que philosophique, a des conséquences concrètes sur l'apprentissage en présence d'anomalies. C'est ce que nous illustrons dans la sous-section suivante, ou nous introduisons les scores robustes et le procédé pour les induire.

4.5.2 Exemple des scores robustes pour estimer une densité

Plaçons-nous dans un cas où notre modèle est mal spécifié. Imaginons qu'en tant que modélisateur nous soyons confronté à un jeu de données issu d'une mixture à densité de probabilité $\mathbb{P}(x) = 0.9\mathcal{N}(0,1) + 0.1\mathcal{N}(5,3)$ (voir figure 4.5). À première vue les données ressemble beaucoup à une simple Gaussienne, mais en réalité, les données sont contaminés. Donc le modélisateur choisis de modéliser une loi Gaussienne $\mathcal{N}(\mu,\sigma)$.

Dans cet exemple, on va modéliser l'apprentissage suivant la méthodologie Bayésienne avec un réseau à un neurone. En voyant les données, on va choisir l'a priori classique que la moyenne $\mu \sim \mathcal{N}(0,10)$ et que la variance est fixe : $\sigma = 1$. Puis, on va apprendre l'a posteriori avec 4 scores différents, dont trois robustes :

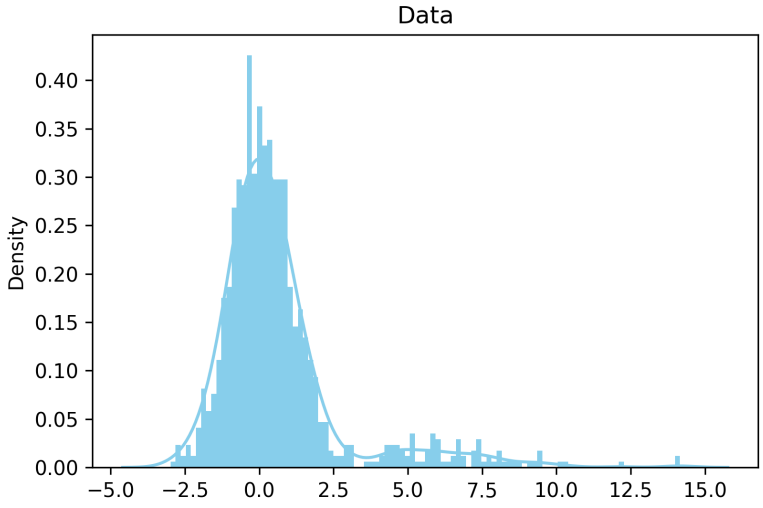
— la log loss issue de la divergence de Kullback-Leibler,

— le coût issue de la divergence de variation totale (TV),

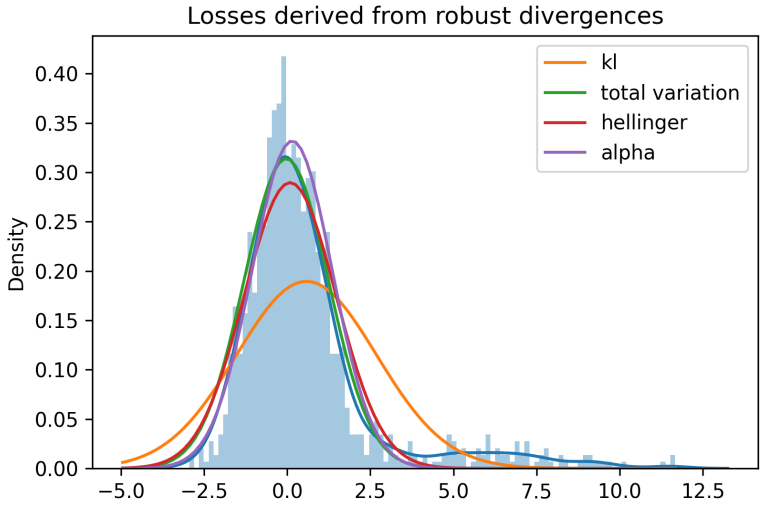
— le coût issue de la divergence de variation d'Hellinger,

— le coût issu de la divergence α.

Les résultats sont disposés sur la figure 4.5. On observe que le log loss pénalise est trop sensible aux anomalies, car si la modélisation de la loi générant les données est parfaite alors les anomalies enverront le plus fort signal d'ajustement. Cela va de soit, car les hypothèses derrière le log loss sont les hypothèses de spécification. En d'autres termes le log-loss cherche



(a)



(b)

FIGURE 4.5: (a) Données initiales. (b) Estimation des données provenant de divers scores

à trouver une Gaussienne générant parfaitement les données, il suppose donc qu'il y a des données inconnus entre 0 et 5 qui doivent encore apparaître. Cet estimateur ne suppose pas la possibilité de corruption ou d'anomalies devant être ignorées. Toute anomalie aura alors un effet disproportionné sur l'apprentissage avec le maximum de vraisemblance. Dans la sous-section suivante, nous décrivons l'origine des trois scores robustes.

4.5.3 Théorie des scores robustes

Huber [Huber, 1964] a posé les bases de la M-estimation en généralisant la méthodologie de maximisation de la vraisemblance décrite plus haut. Dans le cas de l'estimation de densité à la place de minimiser $-\sum_{i=1}^{n} -\log(\mathcal{L}(x^{(i)}; w))$, on va minimiser $\sum_{i=1}^{n} \rho(x^{(i)}; w))$. Huber et Ronchetti [Huber and Ronchetti, 2009] montrent que suivant certaines conditions sur ρ ces estimateurs reprennent certaines bonnes propriétés de l'estimateur du maximum de vraisemblance à savoir : asymptotiquement sans biais et normal. Ici, asymptotiquement sans biais signifie que l'estimateur robuste trouvera le meilleur paramètre du modèle approximant les données. Cependant, on a pas encore donné les conditions sur la fonction ρ pour s'assurer qu'elle montre ces propriétés. On va citer deux approches centrées sur l'idée des scores robustes, c'est à dire considérant ρ comme un score. La première approche est celle de David [Dawid et al., 2016], qui remplace va chercher à minimiser $\sum_{i=1}^{n} S(x^{(i)}, p_w)$ ou S est un score correct. Dans son article, David montre que les scores corrects sont asymptotiquement sans biais et de variance normale. Nous ne rentrerons pas dans les détails de ces preuves, cependant, nous prendrons le temps de décrire et discuter des scores corrects.

Scores corrects

La définition des scores corrects peut varier dans la littérature suivant qu'on cherche à le maximiser avec une interprétation que le score représente une notion d'utilité d'un modèle [Gneiting and Raftery, 2007]. Ou qu'on cherche à le minimiser, c'est le cas où le score est assimilé au coût d'un modèle [Dawid et al., 2016] comme c'est le cas en apprentissage machine. C'est cette définition que nous présentons ci-dessous :

Définition 4.6. Un score S prenant en argument un vecteur de probabilité p et une observation y est dit correct si pour tout vecteur de probabilité r, p on a :

$$\mathbb{E}_{y\sim p} S(y, p) \leq \mathbb{E}_{y\sim p} S(y, r)$$

Le score est dit strictement, correct lorsque l'inégalité ci-dessus est stricte $\forall r \neq p$.

Cette propriété des scores corrects est une incitation pour le modélisateur à être honnête et à choisir la prédiction qu'il pense être la plus proche de la réalité. En effet, si la loi correspondant à l'observation de la classe est p, et que le modélisateur hésite entre deux modèles : le modèle p générant les données et un autre modèle r. Alors, un score strictement correct discriminera p, en lui attribuant un score moyen plus petit qu'avec r. Cette propriété permet à la procédure d'apprentissage de privilégier les modèles les plus proches de la vraie loi générant les données. Et donc, d'exposer la direction maximisant le signal d'apprentissage lors du calcul du gradient.

Robustesse des scores corrects

Un des apports de Hampel [Hampel, 2005] à la mesure de la robustesse a été avec l'introduction des fonctions d'influence. Les fonctions d'influence mesurent l'effet d'une contamination sur la performance d'un estimateur. Une contamination de donnée de masse fixée peut affecter différemment l'estimateur en fonction de sa position dans le support des données $\mathcal{X}$. Dans notre exemple, la contamination était une Gaussienne $\mathcal{N}(5, 3)$, et faire varier sa moyenne aurait affecté la prédiction effectuée avec le log-loss. Le suprémum de la fonction d'influence est une mesure de la sensibilité aux erreurs grossières. Ainsi, si ce suprémum est borné, alors l'estimateur est dit B-robuste. Comme cette fonction d'influence inclue le score utilisé, il suffit que le score soit borné pour qu'il soit considéré comme B-robuste. Ainsi, en choisissant des scores corrects bornés, on est sûr d'avoir un estimateur asymptotiquement sans biais, normal et B-robuste. On voit tout de suite que le log-loss n'est pas borné, donc n'est pas B-robuste : en effet, en contexte de classification binaire si un prédicteur p_w considère un exemple $(x, 0)$ mais lui donne en prédiction l'étiquette 1 (erreur absolue), alors son coût serait $\log(0) = \infty$. Pourtant le *log*-loss est un score correct. Bien que cette approche de la construction des estimateurs robustes par les propriétés des scores soit intéressante dans une perspective de construction de nouveaux scores robustes, elle ne nous donne pas des scores robustes prêt à l'emploi. Un approche plus pratique repose sur les scores issus des divergence robustes [Jewson et al., 2018].

4.5.4 Scores issus de divergences robustes

Une autre approche pour obtenir des scores robustes est de passer par les divergences robustes. En effet, les scores $S(p, y)$ engendrent des divergences généralisées : $d(p, q) = \mathbb{E}_{x\sim p}S(x, q) - \mathbb{E}_{x\sim p}S(x, p)$. On appelle $e(p) = \mathbb{E}_{x\sim p}S(x, p)$ l'entropie généralisée ou la fonction d'information issue du score. On peut partir d'une fonction d'information ou d'une divergence ayant certaines propriétés pour retrouver des scores corrects et/ou robustes. En effet, par exemple une entropie généralisée (strictement) concave est équivalente à un score (strictement) correct [Savage, 1971, Gneiting and Raftery, 2007]. Par ailleurs une entropie généralisée concave va donner lieu à une divergence de Bregman. Ainsi, une stratégie pour obtenir des scores robustes serait de chercher des divergences robustes. C'est ce que propose Jewson [Jewson et al., 2018] à travers l'étude de plusieurs divergences qui ont pour propriété d'être bornées, contrairement à la divergence KL qui engendre le log-loss. Dans notre expérience illustrée en figure 4.5, nous avons choisi la divergence de variation totale, la divergence de Hellinger, et la α-divergence. Le problème de minimisation de divergence requiert une approximation non-paramétrique de la loi générant les données g_n et un modèle paramétrique p_θ. Le but est d'approcher la loi non-paramétrique par un modèle paramétrique. Dans la suite, nous montrons comment obtenir l'expression d'un score à partir du problème de minimisation de divergence.

$$\hat{\theta} = \arg\min_{\theta\in\Theta} d(g_n, p_\theta) \tag{4.5}$$
$$= \arg\min_{\theta\in\Theta} \mathbb{E}_{x\sim g_n} S(x, p_\theta) - \mathbb{E}_{x\sim g_n} S(x, \mathbb{P}) \tag{4.6}$$
$$= \arg\min_{\theta\in\Theta} \mathbb{E}_{x\sim g_n} S(x, p_\theta) \tag{4.7}$$
$$\tag{4.8}$$

Le terme d'entropie $\mathbb{E}_{x\sim g_n} S(x, g_n)$ est éliminé, car il ne dépend pas du paramètre θ, Θ. Ce calcul nous permet d'obtenir respectivement les scores de variation totale, le score Hellinger et le α score :

— $S_{TV}(x, p_\theta) = \frac{1}{2}|1 - \frac{p_\theta(x)}{g_n(x)}|$

— $S_H(x, p_\theta) = \frac{1}{2}(1 - \sqrt{\frac{p_\theta(x)}{g_n(x)}})^2$

— $S_\alpha(x, p_\theta) = \frac{1}{\alpha(1-\alpha)}(1 - p_\theta(x)^{1-\alpha} g_n(x)^{\alpha-1})$

Cette approche fonctionne bien sur un exemple d'estimation de densité en une dimension, mais cependant serait difficile à utiliser pour des données en haute dimension, car il est plus difficile de trouver une loi même approximative générant les données. Nous nous arrêterons donc là sur cette investigation.

4.5.5 Perspectives des scores robustes

Bien que la robustesse algorithmique et statistiques qualifient des problèmes différents, on peut trouver une situation ou leurs objectifs se rejoignent. En effet, il serait intéressant de considérer les scores corrects en classification dans un contexte de glissement de sous-populations [Koh et al., 2021]. Imaginons un problème de classification binaire entre chiens et chats, on peut imaginer que la distribution source admet deux sous-classes de chiens (Berger Allemand et Rottweiler) et deux sous-classes de chats (Bengal et Abyssin).

— La distribution source admet 50% de chiens et 50% de chats, mais admets des sous distributions de 12% de Berger Allemands et 38% de Rottweiler, 22% d'Abyssin et 28% de Bengal.

— La distribution cible admette aussi 50% de chiens et 50% de chats, mais avec les sous distributions de 25% de Bergers Allemands, 25% de Rottweiler, 25% d'Abyssin et 25% de Bengal

Dans ce cas, la loi générant les données est une loi conditionnelle $p(y|x)$. Donc le problème technique de trouver une loi générant les données se poserait moins en sélectionnant un modèle de classification quelconque pour apprendre sur les données source. Puis à utiliser ce modèle comme g_θ, cela pourrait permettre d'obtenir une expression des fonctions coûts robustes, et pourrait permettre d'entraîner un second modèle sur les données cibles.

4.6 Conclusion

Dans ce chapitre, nous avons illustré le problème de changement de distribution. Ensuite, nous avons présenté trois types d'apprentissages ne reposant pas sur l'hypothèse $i.i.d$:

l'apprentissage par transfert, l'adaptation de domaine et la généralisation hors distribution. Nous avons aussi discuté des différences philosophiques entre ces formes d'apprentissages. Ensuite, nous avons situé l'apprentissage robuste comme un sous-ensemble de l'apprentissage pour la généralisation hors distribution. Nous avons ensuite étudié deux littératures reliées utilisant le mot robustesse : la robustesse algorithmique et la robustesse statistiques. Une difficulté de cette thèse est l'éparpillement des littératures traitant de problèmes reliés, mais légèrement différents. Ainsi, il était nécessaire d'entrer dans les détails de chaque littérature pour être sûr de ne pas manquer une subtilité sur la présence d'hypothèse commune ou la conception d'un problème commun. Cependant, nous voulons dans cette conclusion souligner au lecteur que le cas pratique que nous traitons au chapitre 6 ainsi que la plupart des expériences que nous menons au chapitre 7 et 8 sont des problèmes de robustesse comme sous-ensemble de la généralisation hors distribution. Et notre but est d'obtenir des algorithmes robustes comme décrit dans la section robustesse algorithmique. Avant d'en arriver là, nous proposons au chapitre 5 de faire l'état de l'art des méthodes les plus prometteuse pour faire face à ce problème.

Chapitre Cinq

Outils pour un apprentissage robuste

Sommaire

5.1 Introduction

Comme on l'avait vu au chapitre 2 avec la définition de Mitchel [Mitchell, 1997], il y a 3 points importants à définir dans un problème d'apprentissage : la tâche T, l'expérience E, et la mesure de la performance P. Pour concevoir au mieux le problème d'apprentissage robuste, on commence par poser la tâche à travers plusieurs benchmarks. En effet, les glissements de données se manifestent en pratique au travers des glissements synthétiques et naturels. Ensuite, après une revue de la littérature sur les méthodes améliorant la robustesse, nous choisissons de faire une recherche extensive sur l'augmentation de données, l'approche qui nous a semblé la plus prometteuse. L'augmentation de données correspond à une augmentation de l'expérience E, car on injecte de nouvelles sources d'information dans l'apprentissage. Enfin, pour observer une amélioration, il faut une ou plusieurs métriques pour mesurer la performance P. C'est ce dont nous discutons dans notre section sur la mesure de la robustesse. Nous investiguons essentiellement deux approches (1) l'utilisation de l'incertitude des modèles comme proxy à l'erreur de déploiement et (2) l'estimation directe de l'erreur de déploiement à partir d'algorithmes dédiés.

5.2 Typologie des glissements de données

La plupart des travaux que nous détaillons dans les paragraphes suivants ont été réalisés avec deux types de glissement de données. D'une part avec des jeux de données corrompus, dans ce cas, on parle de glissement synthétique de données. D'autre part avec des jeux de données comprenant les mêmes classes, mais venant de domaines différents, on parle de glissement naturel de données.

Définition 5.1. Glissement naturel et synthétique de données

— Un glissement synthétique correspond à un glissement covarié dû à un programme informatique.

— Un glissement naturel de données correspond à un glissement covarié dû à un échantillonnage dans un environnement différent ou à un adversaire.

La différence principale est que dans le premier cas, on connaît la fonction perturbant les données et dans le second cas, cette fonction de perturbation est inconnue.

5.2.1 Glissement synthétique : benchmarks

Hendrycks et Dietterich [Hendrycks and Dietterich, 2019] ont proposé un benchmark pour étudier la robustesse en vision par ordinateur. Le benchmark comprend trois jeux de données CIFAR-10, CIFAR-100 et ImageNet. Ces ensembles de données ont été modifiés avec 15 corruption sur 5 niveaux de sévérité pour créer respectivement CIFAR-10-C, CIFAR-100-C et ImageNet-C. Ils ont aussi proposé un jeu de données avec un niveau de perturbation plus fin sur un continuum afin de créer une vidéo pour chaque corruption / augmentation. Ces ensembles de données sont appelés CIFAR-10-P, CIFAR-100-P et ImageNet-P. Suivant la même méthodologie Mu et Justin (2019) ont introduit MNIST-C [Mu and Gilmer, 2019].

5.2.2 Glissement naturel de données : benchmarks

Parmi les benchmarks les plus utilisés pour les glissements de données naturels, notons ImageNet-A [Hendrycks et al., 2021b] un jeu de données basé sur un échantillon des classes d'ImageNet [Deng et al., 2009], mais avec la particularité d'avoir été constitué sur la base d'un apprentissage adverse. Ici, les auteurs ont pris un ResNet-50 et ont cherché sur Internet des images que le modèle n'arrivait pas à classifier. Ils ont ensuite constitué un jeu de données sur cette base. Aujourd'hui, il s'agit de l'un des jeux de données vers lequel il est le plus difficile de généraliser. ImageNet-R [Hendrycks et al., 2021a], va associer de nouveaux domaines au jeu original. Par exemple, un même objet se retrouverait en photo, peinture, dessin au crayon, etc. Au total, ce jeu propose 8 domaines. Notons aussi ObjectNet [Barbu et al., 2019] qui va prendre un même objet dans de nombreux contextes, par exemple dans ImageNet, une chaise peut être "debout" sans arrière-plan particulier, et ObjectNet proposerait une chaise renversée, ou une chaise dans une salle de bain, etc. ImageNet-Sketch [Wang et al., 2019a] représente toutes les classes d'ImageNet, sous forme de dessin au crayon ou au stylo noir. Voilà donc un petit aperçu non

exhaustif des jeux de données basés sur ImageNet. Cependant, d'autres jeux de données ont aussi été utilisés en généralisation hors distribution multi-source, par exemple : VLCS [Torralba and Efros, 2011], PACS [Li et al., 2017] plus tard rassemblés autour d'un benchmark plus large appelé DomainBed [Gulrajani and Lopez-Paz, 2020a]. Notons aussi WILDS [Koh et al., 2021] un benchmark rassemblant des jeux de données permettant l'apprentissage hors distribution multi-source sur des données incluant plusieurs modalités comme des images, du texte ou des graphes.

5.3 Méthodes pour augmenter la robustesse des modèles

Geirhos et ses collègues [Geirhos et al., 2018] ont comparés les réseaux de neurones profonds aux êtres humains sur une tâche de généralisation avec des données de test de plus en plus corrompues. Pour cela, ils ont choisi trois réseaux de neurones profonds en classification d'images pré-entraînés : VGG-19, GoogleLeNet et ResNet-152. Ces réseaux de neurones ont été comparés avec des humains sur un ensemble de données subissant un bruitage d'une intensité croissantes suivant 12 procédures de corruptions. Ces corruptions incluaient des changements d'échelle, des changement de couleurs, des rotation, du floutage, etc. Ils ont montré que les réseaux de neurones profonds surpassaient les humains dès lors que la distribution d'apprentissage correspondait à la distribution de test. Ce qui repose sur l'hypothèse *i.i.d.* Cependant, les humains obtenaient de meilleurs résultats, lorsque le problème consistait à généraliser à des corruptions inconnues. Ces résultats suggèrent que les humains ont des représentations plus robustes que les réseaux de neurones profonds. En particulier, ce travail de Geirhos (2018) et ses collègues montrent qu'un point crucial dans la méthodologie d'apprentissage machine aujourd'hui est d'induire les réseaux de neurones à généraliser sur des corruptions inconnues. Cette étude a pavé la voie à plusieurs autres études introduisant des benchmarks pour mesurer les progrès vers cet objectif.

Les travaux d'Ovadia (2019) ont montré que les techniques de recalibration sur des réseaux de neurones simples (non Bayésiens) ne sont pas utiles lorsque le niveau de corruption des données augmente. La meilleure stratégie pour obtenir une bonne calibration consiste à utiliser des ensembles de réseaux de neurones qui sur-performent les réseaux de neurones Bayésiens entraînés avec l'inférence variationnelle.

Notons aussi le travail de Hendrycks [Hendrycks et al., 2019] avec Augmix qui a introduit une méthode combinant diverses augmentations génériques pour créer de nouvelles augmentations uniques en grande quantité. Cette méthode est parvenue à diminuer significativement l'erreur de corruption moyenne comparé à d'autres méthodologies d'augmentation de données. Cependant, les résultats étaient principalement sur des benchmarks de glissement de données synthétiques : CIFAR-10-C, ImageNet-C.

Geirhos et ses collègues [Geirhos et al., 2019] ont montré que les CNN actuels sont biaisés en faveur de la texture. Ils l'ont montré en utilisant une version stylisée d'ImageNet avec transfert d'image. Selon eux, les classificateurs ImageNet s'appuient sur les informations de texture locales, car c'est plus simple, un peu comme dans notre expérience de pensée en introduction. En effet, il est plus facile d'apprendre des corrélations incorrectes, mais qui expliquent la majorité des données (vaches dans une prairie entraîne prairie=vache) plutôt que d'apprendre la forme des objets. Ce biais de texture est détrimental pour l'apprentissage de représentations robustes, car changer l'environnement d'un objet fait perdre au

modèle ses repères. Les auteurs ont alors induit un biais de forme dans leur apprentissage pour forcer les CNN à ne pas faire attention aux informations de texture, mais bien à reconnaître les objets en fonction de leur forme. Cette approche a accru la robustesse contre des glissements de données synthétiques inconnues.

Récemment, Hendrycks et ses collègues [Hendrycks et al., 2021a] ont étudié plusieurs facettes de la robustesse. Ils ont étudié quatre hypothèses de modélisation pour améliorer la robustesse :

— Profondeur des modèles : augmenter le nombre de couche cachées d'un réseau de neurones améliore sa robustesse.

— Auto-attention : l'ajout de couches d'auto-attention aux modèles améliore leur robustesse.

— Augmentation de données diversifiée : plus on peut obtenir d'augmentation de données diversifiées plus on pourra améliorer la robustesse des modèles.

— Pré-apprentissage : le pré-entraînement sur des ensembles de données plus grands et plus divers améliore la robustesse.

Et trois propriétés abstraites de robustesse :

— Biais de texture : les réseaux convolutifs sont biaisés envers la texture, ce qui nuit à la robustesse.

— Seule l'exactitude $i.i.d$ compte : la précision sur des données de test indépendamment et identiquement distribuées est prédictive de l'exactitude sur des données hors distribution.

— Synthétique $\not\Rightarrow$ Naturel : les augmentations de données synthétiques (non-naturelles) ne contribuent pas à la robustesse sur des changements de distribution naturels.

Chacune des quatre hypothèses de modélisation peut être étudiée avec des benchmarks spécifiques. Mais des propriétés plus générales de robustesse ne peuvent être étudiées qu'avec de nouveaux benchmarks permettant d'isoler les effets de confusion. Les auteurs ont donc utilisé les jeux de données StreetView StoreFronts (SVSF) et DeepFashion Remixed (DFR) permettant de fixer la texture et de faire varier l'heure, la caméra, l'emplacement, etc.

Les expériences des auteurs ont entraîné les conclusions suivantes : deux hypothèses de modélisation permettent d'augmenter la robustesse :

1. l'augmentation des données diversifiées et
2. les modèles profonds.

Aussi, une hypothèse sur les propriétés abstraites de robustesse s'est révélée valide : le biais de texture en accord avec [Geirhos et al., 2019]. Cependant, bien que le fait de retirer le biais de texture et d'induire un biais de forme améliore la robustesse face à des glissements synthétiques, ce biais n'est pas utile face à des glissements de données naturels. La précision de l'$i.i.d$ n'est pas suffisamment prédictive de la perte de généralisation sur des glissements de données divers. Les auteurs ont conclu en émettant l'hypothèse que la robustesse est nécessairement multivariée et ne peut être capturée avec une seule métrique.

Notons aussi le travail de Taori et ses collègues [Taori et al., 2020] qui ont montré que collecter plus de données permet certes d'améliorer la généralisation $i.i.d$, mais que ce n'est

pas le plus efficace pour généraliser hors distribution. En particulier, ils ont montré que les augmentations synthétiques bien qu'elles peuvent améliorer la généralisation sur d'autres corruptions inconnues, ne sont que très peu efficaces sur des glissements de données naturels. En d'autres termes, la généralisation obtenue sur un glissement synthétique n'est pas prédictive d'une généralisation qu'on peut obtenir sur un glissement naturel. Ce résultat entre en contradiction avec les résultats de [Hendrycks et al., 2021a]. En effet les travaux de [Hendrycks et al., 2021a] ont trouvé une amélioration de la robustesse sur des glissements naturels et synthétique, mais seuls trois jeux de données avaient été considérés : ImageNet-C et Real Blurry Image pour les glissements synthétiques et ImageNet-R pour le glissement naturel. Finalement Milner et ses collègues [Miller et al., 2021] ont montré une tendance linéaire entre la généralisation obtenue sur la distribution du jeu d'apprentissage et la généralisation obtenue sur un glissement hors distribution. Les valeurs ont été calculées avec divers modèles et architectures. En un sens, ils ont montré que les glissements de données unidirectionnels entraînent des comportements similaires pour de nombreux modèles d'apprentissage. En d'autres termes, leurs expériences montrent que si nous avons de nombreux modèles entraînés sur une source de données et que nous évaluons leur perte de performance sur des distributions cibles, alors les modèles devraient perdre en précision de la même manière sur chaque distribution cible. Cependant, leur analyse n'indique pas si la précision $i.i.d$ est prédictive de la précision hors distribution en général comme ils l'affirment. En effet, pour avoir cette réponse, ils auraient dû fixer un modèle et évaluer sur différents domaines pour évaluer si la précision hors distribution varie autour de la précision obtenue sur le jeu d'apprentissage. En d'autres termes, l'effet statistique aurait dû être sur le nombre d'autres domaines évalués pour un modèle donné.

Ce que nous pouvons conclure de leur étude cependant, c'est le comportement commun de diverses architectures. Peut-être est-ce là un signe d'un comportement de la fonction de perte coût en commun, ou peut-être est-ce un comportement inhérent à l'algorithme de la minimisation des risques empirique [Vapnik, 1992] qui aura des résultats linéaires en augmentant le nombre de paramètres avec des architectures capables de passer à l'échelle.

L'une des hypothèse de modélisation qui s'est révélée utile pour améliorer la robustesse est l'augmentation de données diversifiée [Hendrycks et al., 2021a], cela fait consensus avec [Taori et al., 2020]. Nous allons nous pencher plus en profondeur sur l'augmentation de données dans la prochaine section.

5.4 Focus sur l'augmentation de données

L'augmentation de données a été largement utilisée pour améliorer les performances de généralisation des modèles d'apprentissage profonds [Dao et al., 2019]. Cette technique a permis d'atteindre l'état de l'art sur de nombreuses tâches telles que la vision par ordinateur [Shorten and Khoshgoftaar, 2019] et le traitement de la parole [Ko et al., 2015, Park et al., 2019]. Dans cette section, nous étudions la technique d'augmentation de données en profondeur, nous la mettons en perspective comme une technique de régularisation suivant la perspective de Kukacka et al. [Kukačka et al., 2017] présentée au chapitre 2. En effet, selon Kukacka et al. l'augmentation de données est la méthode offrant le plus de liberté pour induire des connaissances a priori sur un problème d'apprentissage. Par ailleurs, nous montrerons des équivalences entre l'augmentation de données et l'utilisation

d'un terme de régularisation.

5.4.1 Historique du terme

Le premier qui a mentionné le terme d'augmentation de données [Tanner and Wong, 1987] l'a utilisé pour simuler la distribution a posteriori dans l'inférence Bayésienne. Leur interprétation consistait à utiliser des données augmentées comme variables latentes et fournissait un algorithme Markov Chain Monte Carlo itératif. Comme l'a fait remarquer [Van Dyk and Meng, 2001], le terme a évolué dans la littérature physique et a ensuite été appelé la méthode de la variable auxiliaire, il a été utilisé par exemple pour échantillonner à partir des modèles Ising et Post [Swendsen and Wang, 1987]. Cependant, la technique d'augmentation de données telle qu'elle est utilisée aujourd'hui dans la littérature sur l'apprentissage machine a été bien formulée par Dao et al. (2019) qui ont décrit son effet de régularisation par induction de robustesse avec invariance et pénalisation de la complexité du modèle. L'augmentation de données a permis d'obtenir des résultats d'état de l'art dans de nombreux domaines tels que la vision par ordinateur [Shorten and Khoshgoftaar, 2019] et le traitement de la parole [Ko et al., 2015, Park et al., 2019].

5.4.2 Augmentation de données et régularisation

On peut trouver une équivalence entre l'augmentation de données et l'ajout d'un terme de régularisation. C'est ce que propose Tangeant-prop [Simard et al., 1991] Cela est possible en ajoutant un terme de régularisation qui garantisse que le modèle ait un gradient nul dans le sens d'une transformation de données. En effet, il existe une hypothèse en apprentissage machine appelée hypothèse de la variété [Goodfellow et al., 2016]. Cette hypothèse stipule que les données en grandes dimensions existent en réalité sur une sous variété de dimension intérieure. Cela fait sens, en effet les chiffres de MNIST (photos de dimension 28×28) sont plus vraisemblablement sur une sous-variété de dimension 2 (car ce sont des images sur un plan) que sur un espace de dimension $28{\times}28{=}784$. Cette hypothèse laisse penser qu'on pourrait "glisser" d'un exemple x à un autre exemple z si on avait connaissance de la sous-variété sous-jacente. Ainsi, si on considère que les données existent sur une sous-variété, appliquer respectivement des rotations comprises dans $[\frac{-\pi}{4}, \frac{\pi}{4}]$ à une image est revient à "découvrir" des données existant sur la sous-variété des données suivant un chemin. Ainsi, en ajoutant un terme de régularisation tel que le gradient suivant ce chemin soit nul revient à rendre un modèle invariant aux rotations comprises entre $[\frac{-\pi}{4}, \frac{\pi}{4}]$. En effet, un gradient nul signifie que le modèle ne change pas ses prédictions sur des données qui varient sur le chemin définit par les rotations.

Dao et al. [Dao et al., 2019] ont utilisé une approximation de Taylor de la fonction coût lorsque les données étaient augmentées. La décomposition au premier ordre a montré que l'augmentation de données affectait le réseau de neurones de sorte qu'il apprenne des couches cachées invariantes en moyenne aux données augmentées. La décomposition du second ordre a mis en avant un terme de régularisation calculant la variance du modèle suivant l'amplitude de l'augmentation des données. Ainsi la minimisation de la fonction coût approximative entraînait deux propriétés au réseau de neurones : la recherche de couches

cachées invariante, et de faible variance face aux données augmentées. Leurs résultats expérimentaux montrent que cette approximation de la fonction de Taylor de la fonction coût donnait une généralisation entre 70% et 100% proche de la généralisation obtenue avec l'augmentation de données brute, tout en gagnant du temps face à l'augmentation de données.

5.4.3 Recherche de nouvelles augmentations de données

Un autre axe de recherche se concentre sur la recherche de schémas d'augmentation bons ou optimaux avec les données disponibles. Ces schémas d'augmentation reposent sur des modèles génératifs [Tran et al., 2017, Lemley et al., 2017, Shrivastava et al., 2017] ou d'apprentissage par renforcement [Lim et al., 2019, Cubuk et al., 2019]. La recherche de bons schémas d'augmentation est similaire à la recherche de facteurs de variation non-informatifs dans les données (e.g. rotation, flou, bruitage Gaussien). Ceci peut être vu comme une tâche orthogonale à l'extraction de facteurs de variation informatifs via un pré-apprentissage. De plus, la combinaison des deux approches a été utilisée avec succès pour obtenir des résultats de pointe en matière de classification de texte et de vision par ordinateur en apprentissage semi-supervisé [Xie et al., 2019].

5.4.4 Augmentation de données en ligne ou hors ligne

Lorsque l'ensemble de données disponible est petit, l'augmentation de données est souvent appliquée hors ligne, cela signifie qu'une ou plusieurs transformations sont appliquées plusieurs fois sur l'ensemble de données d'origine pour augmenter la taille de l'ensemble de données. La quantité d'augmentation est appelée facteur d'augmentation, par exemple lorsque la taille totale de l'ensemble de données est 16 fois plus grande, le facteur d'augmentation est de 16. Le problème de cette approche est sa consommation de mémoire, par exemple, un grand ensemble de données comme Imagenet [Deng et al., 2009] pourrait difficilement être augmenté d'un facteur 64, car il prendrait trop de place dans la mémoire des ordinateurs portables personnels. Lorsque l'ensemble de données est volumineux, une alternative consiste à appliquer l'augmentation de données en ligne, ce qui signifie que chaque lot de données est augmenté durant la propagation avant d'être introduit dans le modèle. Par conséquent, l'augmentation de données en ligne est économe en mémoire, mais consomme plus de ressources du processeur, car une opération de perturbation des données est ajoutée à la boucle d'apprentissage. De plus, l'augmentation en ligne entraîne une distribution de données plus bruyante que l'augmentation hors ligne, en raison du changement constant pendant la formation.

5.4.5 Augmentation de données, invariance, et généralisation hors domaine

Dans le contexte de la généralisation hors domaine, l'augmentation de données a souvent été utilisée que ça soit en contexte multi-source [Zhou et al., 2021], ou en contexte mono-source avec des méthodes comme Augmix, DeepAugment [Hendrycks et al., 2019, Hendrycks et al., 2021a]. Mais en plus de ça, notre expérience avec l'augmentation de données sur les modèles génératifs de lettres décrits en section 7.2 nous avait donné des résultats

remarquables en termes d'efficacité d'apprentissage. En effet, nous étions capables d'apprendre une distribution de données avec très peu d'exemples (autour de cent exemples). Ce qui contraste beaucoup avec l'apprentissage profond, qui a typiquement besoin de données massives.

En cherchant les raisons pouvant entraîner un apprentissage efficace, nous avons découvert la conjecture de Anselmi et ses collègues [Anselmi et al., 2016]. Selon eux, la clé de l'apprentissage avec peu d'exemples est l'invariance. Les auteurs ont fait valoir que la majeure partie de la complexité des tâches de reconnaissance est due aux changements d'angles ou d'éclairage qui permutent les caractéristiques intrinsèques des objets pour les modèles. Cela suggère que si un oracle pouvait factoriser toutes les transformations concernant le point de vue, l'échelle, la position, etc. Le problème de la catégorisation deviendrait facile et pourrait être fait avec très peu d'exemples étiquetés.

Les nuisances telles que décrites par [Anselmi et al., 2016] sont également appelées facteurs de variation non-informatifs par Bengio [Bengio et al., 2013b]. Ce dernier classe le facteur de variations en deux catégories : les facteurs de variation non-informatifs et les facteurs de variation informatifs pour la tâche à accomplir. Des facteurs de variation informatifs sont des caractéristiques intrinsèques aux objets. Par exemple, pour une tâche de classification entre chien et araignée, le nombre d'yeux est un facteur de variation informatif.

Les facteurs de variation non-informatifs préservent les étiquettes, par exemple, nous pourrions appliquer des rotations, des flous ou des traductions sur une image de chien, les images résultantes représenteraient toujours un chat pour la perception humaine, mais pourraient tromper un modèle qui n'est pas invariant à de telles transformations. En d'autres termes, les facteurs de variation informatifs sont intrinsèques aux objets, là où les facteurs de variation non-informatifs sont relatifs aux capteurs qui mesurent ou enregistrent l'objet. Ainsi, un modèle idéal serait sensible aux facteurs de variation informatifs et invariant aux facteurs de variation non-informatifs.

5.4.6 Catégorisation des méthodologies d'augmentation de données en vision par ordinateur

Selon Zhou [Zhou et al., 2021], on peut ranger les augmentations visuelles suivant 4 catégories voir la figure 7.1.

1. les transformations génériques : elles concernent toutes les transformations de données que l'on peut implémenter sous forme de code simple, par exemple du bruit Gaussien, des flou, des rotations. Voir [Shorten and Khoshgoftaar, 2019] pour une revue de littérature plus complète.
2. les transformations adverses : il s'agit des transformations d'image visant à les perturber suffisamment pour faire changer la prédiction d'un modèle. Ces transformations sont basées sur l'apprentissage adverse [**?**]. Par exemple les travaux de Volpi [Volpi et al., 2018] sont dans cette catégorie, mais notons aussi [Wong et al., 2019].
3. les transformations profondes : ce sont toutes les augmentations de données basées sur des modèles d'apprentissage profond. Par exemple celles basées sur des modèles génératifs [Tran et al., 2017, Lemley et al., 2017, Shrivastava et al., 2017] ou d'ap-

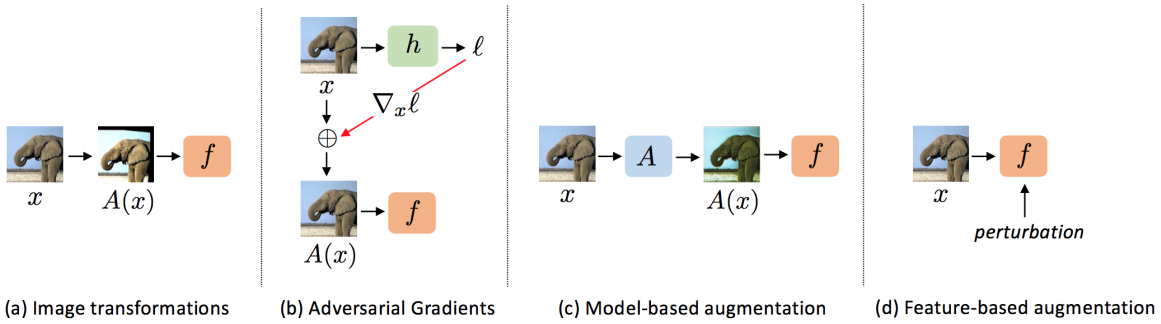


FIGURE 5.1: Les 4 catégories d'augmentation de données selon [Zhou et al., 2021] : (a) les transformations génériques (b) les transformations adverses , (c) les transformations profondes (d) le transfert d'invariance direct au modèle.

prentissage par renforcement [Lim et al., 2019, Cubuk et al., 2019], mais aussi plus récemment celles utilisant le transfert de style [Geirhos et al., 2019]

4. le transfert d'invariance direct au modèle : transférer les effets de l'augmentation de l'espace des données à l'espace des paramètres d'un modèle avait été établi depuis [Simard et al., 1991], mais l'équivalence a été formalisée théoriquement [Dao et al., 2019]. Citons les approches de Nalisnick [Nalisnick and Smyth, 2018] qui injecte l'invariance directement dans l'a priori d'un modèle Bayésien, mais notons aussi [Wang et al., 2019c, Yang et al., 2020] qui manipulent directement l'avant-dernière couche caché en tirant profit des informations apprises par les classes précédentes. Ou les plus récentes approches spécifiques à la généralisation hors domaine [Arjovsky et al., 2020, Krueger et al., 2021].

Notons que certains développements récents appartiennent à plusieurs catégories, c'est le cas d'AugMix [Hendrycks et al., 2019] par exemple, qui à la fois va combiner des transformations génériques, mais va aussi modifier la fonction coût.

5.5 Mesure de la robustesse

Une approche de la mesure de la généralisation hors distribution vient de la sélection du domaine ayant le risque le plus élevé. C'est le cadre qu'a proposé Arjovsky [Arjovsky, 2020] pour mesurer le risque hors distribution, c'est-à-dire :

$$\mathcal{R}_{OOD}(f) = \max_{e \in \mathcal{E}} \mathcal{R}_e(f) \tag{5.1}$$

avec $\mathcal{E}$ l'ensemble des environnements, et $\mathbb{P}^e$ la distribution des données venant de l'environnement $e \in \mathcal{E}$, et $l(z, f(z))$ la perte subie par le prédicteur f sur les données z et $\mathcal{R}_e(f) = \mathbb{E}_{z \sim \mathbb{P}^e} l(z, f(z)))$ le risque pour l'environnement e :

Cependant, on peut étendre naturellement la méthodologie traditionnelle d'apprentissage automatique en prenant une moyenne plutôt qu'un maximum. En effet, si on suppose que les environnements e suivent une loi génératrice de domaine $\mathbb{P}^{\mathcal{E}}$, alors on peut calculer l'erreur moyenne. C'est ce que propose Hendrycks avec l'erreur moyenne sur plusieurs corruptions [Hendrycks and Dietterich, 2019] :

$$\mathcal{R}_{OOD}(f) = \mathbb{E}_{e\sim\mathbb{P}^{\mathcal{E}}}\mathcal{R}^e(f) \tag{5.2}$$

Plus spécifiquement Hendrycks [Hendrycks and Dietterich, 2019] propose l'erreur moyenne sur les 5 niveaux de corruption pour les 15 corruptions (erreur de corruption moyenne). Cependant, ces métriques ne peuvent pas être utilisées si les données de déploiement ne sont pas étiquetées. En effet, dans le chapitre 4, nous avions discuté des aspects philosophiques de la généralisation hors distribution et de la robustesse. Et on se place dans un contexte où nous n'avons pas forcément accès aux données de déploiement étiquetées. Dans ce cas, il y a deux approches que nous avons observées dans la littérature. Dans la suite, l'erreur de généralisation hors distribution est aussi appelée erreur de déploiement.

— L'utilisation de l'incertitude des modèles de prédiction comme proxy à l'erreur de généralisation hors distribution. Par exemple, l'erreur de calibration évalue la qualité de l'incertitude d'un modèle lorsque ce dernier est évalué sur un nouveau domaine. Elle est mesurée par l'erreur de calibration quadratique moyenne et le score de Brier.

— L'utilisation d'algorithmes dédiés pour estimer l'erreur commise lorsqu'un modèle est en contexte de glissement de données

5.5.1 L'incertitude des modèles comme proxy à l'erreur de déploiement

Ovadia [Ovadia et al., 2019] et ses collègues ont étudié la qualité de l'incertitude sur deux formes de décalage de jeu de données : (1) dégradation continue de l'ensemble de données (2) détection d'exemple hors distribution. Les auteurs ont considéré de nombreux modèles probabilistes Bayésiens ou non sur des gros jeux de données en vision par ordinateur avec notamment les jeux CIFAR-10-C and ImageNet-C proposés par Hendrycks et Dietterich (2019) et en traitement du langage naturel. Ce travail de benchmarking à montré que la calibration et l'exactitude des modèles décroît lorsque la corruption des données augmente ou que les modèles sont soumis à des exemples hors distribution ce qui est conforme à l'intuition. Cependant, la calibration obtenue sur le jeu de test n'est pas indicative de la calibration sur les jeux de données dégradés. En d'autres termes, les performances sur le jeu de test ne sont pas indicatives de futures performances sur des jeux corrompus si les corruptions sont inconnues. Par ailleurs, plusieurs auteurs ont cherché à obtenir une métrique prédictive de la chute de généralisation lorsque le jeu de test subi un glissement de données, citons [Ovadia et al., 2019] qui a évalué la calibration, et [Hendrycks et al., 2021a, Miller et al., 2021] qui ont évalué la généralisation sur le jeu test. Dans chaque cas, la métrique utilisée n'était pas prédictive de la perte de généralisation. Selon Hendrycks et al. [Hendrycks et al., 2021a] la précision sur le jeu test pouvait être utilisée comme indicateur de la généralisation hors distribution, mais a constaté qu'elle n'était pas décisive. Il a conclu que les mesures univariées comme la précision sur le jeu de test ne pouvaient pas capturer les changements subtils dans la distribution des données responsables de la perte de généralisation. Nous investiguons cette litérature en section 5.6.

5.5.2 Estimation de l'erreur de déploiement

Une tendance récente dans la littérature sur la mesure de l'erreur de déploiement a été de s'éloigner des métriques fixes basées sur les sorties des modèles comme la calibration. Au lieu de cela, des algorithmes prêts à l'emploi spécialement conçus pour estimer la précision en cas de glissement de jeux de données sont utilisés. Nous appelons ces algorithmes, estimateurs d'erreur de déploiement. L'erreur de déploiement (DE) ou le risque hors distribution [Arjovsky, 2020] est la perte de performances à laquelle un modèle doit faire face lorsqu'il est réellement déployé sur un nouveau domaine. Nous étudions cette littérature en section 5.7

Nous pensons que trouver un proxy à la généralisation lors d'un glissement des données est une tâche très importante. En effet, les progrès en apprentissage machine quelle que soit la tâche sont tributaires de la spécification d'une bonne mesure de la performance [Mitchell, 1997]. Le challenge serait d'obtenir une métrique de l'erreur de déploiement pertinente sur une grande diversité de glissements de données.

5.6 Incertitude des modèles

5.6.1 Quantité et qualité de l'incertitude en régression

On interprète la sortie d'une régression par une loi Gaussienne, c'est à dire par une moyenne et une variance associée pour chaque point (voir figure 5.2 a). La variance permet de calculer des intervalles de confiance et des intervalles de prédictions. Les intervalles de confiance capturent un paramètre d'un échantillon, par exemple la moyenne avec une précision α. On parle alors d'intervalle de confiance de niveau α avec α souvent égal à 95 %. Les intervalles de prédictions eux capturent les éléments d'un échantillon avec un niveau α. Pour apprécier la qualité des intervalles de prédictions, nous discutons trois métriques issues de Pearce et al. [Pearce et al., 2018] pour apprécier les incertitudes calculées : la probabilité de couverture de l'intervalle de prédiction ("Prediction Interval Coverage Probability" cf PICP), la largeur moyenne de l'intervalle de prédiction ("Mean Prediction Interval Width" cf MPIW) et la largeur de l'intervalle naturel ("Natural Interval Width" cf NIW) que nous décrivons ci-dessous. Le PICP est calculé comme suit :

$$PICP = \frac{1}{n}\sum_{i=1}^{n} k_i$$

avec $k_i = 1$ si $\hat{y}_{L_i} \leq y_i \leq \hat{y}_{U_i}$ et $k_i = 0$ sinon. Et $\hat{y}_{L_i}$ et $\hat{y}_{U_i}$ respectivement les bornes inférieures et supérieures estimées. En d'autres termes, le PICP nous indique le pourcentage des données capturées par notre intervalle de crédibilité. Le MPIW est calculé comme suit :

$$MPIW = \frac{1}{n}\sum_{i=1}^{n} \hat{y}_{U_i} - \hat{y}_{L_i}$$

le MPIW nous informe sur le degré d'étroitesse de l'intervalle de prédiction en moyenne, plus il est bas, mieux c'est. Enfin, pour avoir une perspective, nous calculons le NIW

$$NIW = \max_{i=1,\cdots,n} y_i - \min_{j=1,\cdots,n} y_j$$

Le NIW est simplement l'intervalle dans lequel les données vivent.

Le PICP et le MPIW sont en tension. En effet, le meilleur intervalle de prédiction capturerait une grande partie des données, donc il impliquerait un PICP élevé. Cependant, l'intervalle doit également être serré, car des intervalles serrés signifient que l'estimateur d'erreur de déploiement est assez certain de l'effet des glissements de données sur l'erreur de déploiement. Des intervalles larges signifient que l'estimateur d'erreur de déploiement est très incertain quant à l'effet de nouveaux glissements de données sur l'erreur de déploiement. Il y a donc un équilibre entre essayer de capturer autant de données que possible dans l'intervalle crédible et le faire aussi serré que possible. Enfin, le NIW permet de mettre en perspective les intervalles que nous calculons, en effet de bons intervalles de confiance devraient avoir un MPIW plus petit que le NIW. En effet, l'intervalle de confiance trivial pourrait consister à juste prendre le minimum et le maximum de données. Dans ce cas, il obtiendrait un PICIP de 100%.

5.6.2 Quantification de l'incertitude en classification avec l'entropie de Shannon

Pour la classification, il y a plusieurs manières de mesurer l'incertitude :

— On peut quantifier l'incertitude avec l'information de Shannon.

— On peut évaluer la qualité de l'incertitude en mesurant la confiance associée à une prédiction. Par exemple en classification binaire, un exemple étant classifié avec $P(X = 0) = 0.4, P(X = 1) = 0.6$, alors la confiance en la prédiction sera de 60%. Cela permet de déduire la qualité de l'incertitude avec la calibration du modèle, voir ci-dessous.

Ces deux approches de mesure de l'information sont complémentaires et permettent de déduire des informations différentes sur l'incertitude d'un modèle et sur sa qualité. Définissons l'entropie de Shannon et voyons comment elle est utilisée pour mesurer l'incertitude d'un modèle.

Définition 5.2. Soit une loi de probabilité p sur un espace de paramètres $\Theta \subset \mathbb{R}^d$, alors on appelle entropie de Shannon associée à la loi p et on note $\mathbb{H}$ la quantité suivante :

$$\mathbb{H}(p) = -\int_{\Theta} p(\theta) \log(p(\theta)) d\theta$$

Lorsque le logarithme suit la base 2, on la mesure en *bit* sinon on parle de *nat*.

L'entropie de Shannon définit la quantité de *surprise* associée à une prédiction. Par exemple, imaginons une pièce de monnaie non équilibrée que l'on modélise avec une loi de Bernoulli de paramètre p : $\mathcal{B}(p)$ et notons X le résultat du lancé. Le tirage de cette pièce peut renvoyer 0 ou 1 . Si le lancé de la pièce suit une loi de paramètre $\mathcal{B}(0)$ avec $P(X = 0) = 1, P(X = 1) = 0$. Alors cette pièce, on aura tout le temps même le même résultat même après 1000 lancés. Ainsi la surprise lors du tirage de cette pièce est minimale. En d'autres termes, le tirage peut être décrit avec peu d'information avec un seul symbole le résultat 0. L'entropie de Shannon sera alors minimale. Le raisonnement est similaire si la pièce renvoie tout le temps 1. Cependant, si la loi sous-jacente est $\mathcal{B}(0.5)$ avec $P(X =$

$0) = 0.5, P(X = 1) = 0.5$, cas d'un tirage d'une pièce de monnaie équilibré, alors, on ne saura jamais à l'avance le résultat. La surprise est maximale, et l'information décrivant le résultat doit être codée avec au minimum deux symboles 0 et 1. Soit un bit d'information. C'est ce que mesure l'entropie de Shannon (voir Fig. 5.2 b).

L'entropie de Shannon nous permet alors de comparer quantitativement l'incertitude de plusieurs modèles. Quantifier l'incertitude est fondamentalement basé sur l'entropie donc peu importe le nombre de classes la démarche reste la même. Illustrons cela avec un exemple simple de classification binaire.

5.6.3 Exemple de quantification de l'incertitude en classification binaire

Nous allons entraîner deux réseaux de neurones, le premier est un réseau à une couche caché de 10 neurones, et le second est un réseau à une couche cachée de 10 neurones, cependant on va appliquer Monte Carlo Dropout [Gal and Ghahramani, 2016] à chaque itération pour rendre le second modèle incertain. En effet, Gal et ses collègues ont montré qu'en utilisant dropout plusieurs fois lors de la propagation, cela revient à induire une loi de probabilité sur les neurones, et ainsi cela permet au réseau d'estimer l'incertitude. Ces réseaux sont appelés réseaux de neurones Bayésiens, nous en discutons plus en détail au chapitre 3.

Pour le jeu d'apprentissage, on se place sur le plan $\mathbb{R}^2$ et on tire 100 exemples suivant respectivement les lois suivantes $\mathcal{N}([-1, 1]), \Sigma)$ et $\mathcal{N}([1, 1], \Sigma)$ avec Σ la matrice de covariance égale à l'identité. Voir la première ligne et la première colonne de la Fig. 5.3 Chaque modèle est entraîné avec 500 epochs, utilisant la descente du gradient stochastique avec Adam [Kingma and Ba, 2014].

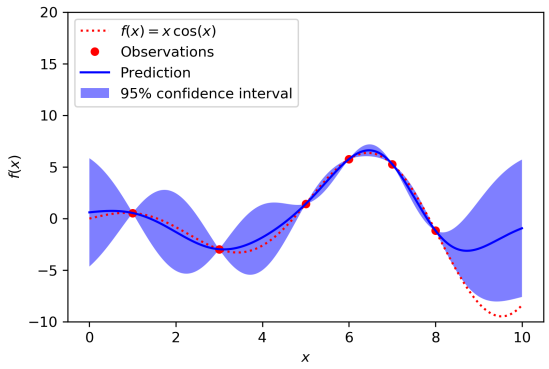
Pour quantifier l'incertitude du modèle, on va utiliser des données en dehors de la distribution de départ, on évaluera leur entropie et on tracera la densité et la fonction de répartition comme cela a été fait dans la littérature [Lakshminarayanan et al., 2017, Louizos and Welling, 2017]. Cette évaluation sera répétée 4 fois avec des données binaires de plus en plus confuses à savoir suivant respectivement les couples de lois $\{(\mathcal{N}([-0.8, 1], \Sigma), \mathcal{N}([0.8, 1], \Sigma)); (\mathcal{N}([-0.6, 1], \Sigma), \mathcal{N}([0.6, 1], \Sigma))$ $; (\mathcal{N}([-0.4, 1], \Sigma), \mathcal{N}([0.4, 1]), \Sigma); (\mathcal{N}([-0.2, 1], \Sigma), \mathcal{N}([0.2, 1], \Sigma))\}$. Voir la première ligne de la Fig. 5.3

Dans notre exemple, l'entropie est considérée comme une variable aléatoire et est définie pour chaque exemple x par la formule suivante :

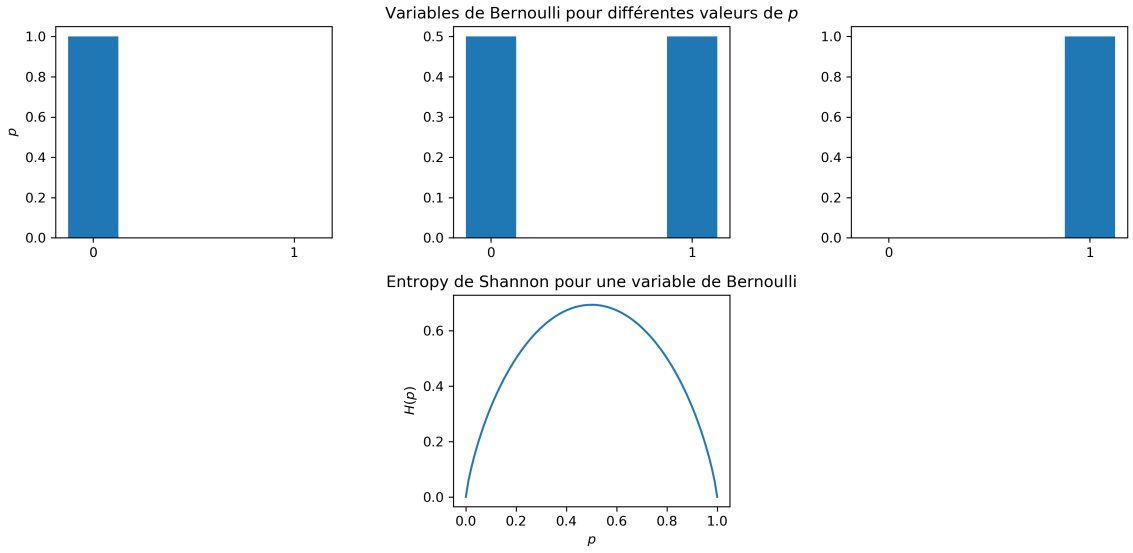
$$\mathbb{H}(p(y|x)) = p(y = 0|x) \log p(y = 0|x) + p(y = 1|x) \log p(y = 1|x)$$

Dans la suite, notons $\mathbb{H}_S$, l'entropie du réseau de neurone simple et $\mathbb{H}_D$ l'entropie du réseau de neurones entraîné avec Monte-Carlo Dropout.

Une fois que l'entropie est calculée pour chaque exemple du jeu d'évaluation concernée, on peut se représenter la loi de probabilité de l'entropie de chaque modèle en utilisant la fonction de répartition (figure 5.3 ligne 2) ou la fonction de densité (figure 5.3 ligne 3). Chaque méthode permettant d'avoir une perspective différente sur la distribution de l'entropie calculée sur les jeux de données.



(a)



(b)

FIGURE 5.2: (a) Régression, un processus Gaussien effectue une prédiction et quantifie l'incertitude avec un intervalle de confiance. (b) Classification, on utilise l'entropie de Shannon pour mesurer l'incertitude d'un modèle. Dans le cas d'une classification binaire, on utilise une loi de Bernoulli de paramètre p :$\mathcal{B}(p)$. On observe que plus la loi se rapproche d'une loi uniforme, plus l'entropie est élevée.

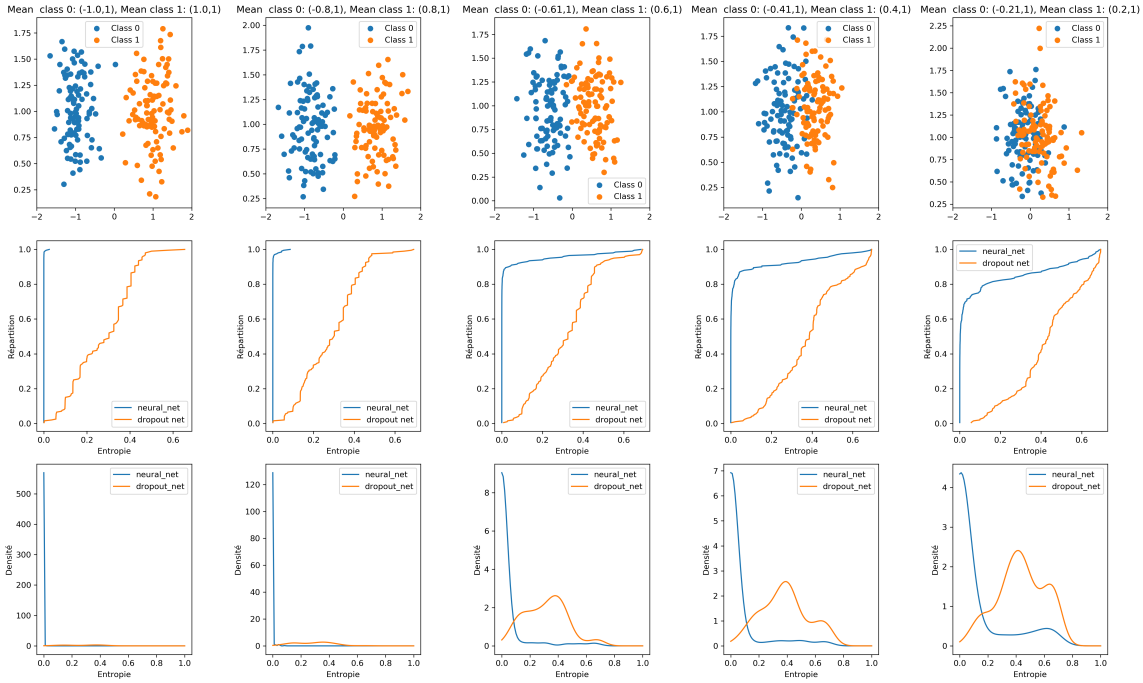


FIGURE 5.3: Quantification de l'incertitude. La première colonne représente les résultats sur le jeu d'entraînement. Les quatre colonnes suivantes représentent les résultats sur des jeux d'évaluation de plus en plus confus. La première ligne représente les données sur lesquelles les modèles sont évalués. La deuxième ligne représente la fonction de répartition de l'entropie, et la dernière ligne la fonction de densité de l'entropie. Les lignes bleu et orange représentent respectivement les modèles simples et modèles Dropout sur les lignes 2 et 3.

Quantification de l'entropie avec la fonction de répartition

Pour rappel, la fonction de répartition caractérise une loi de probabilité tout comme la fonction de densité. Pour une variable aléatoire X elle suit la formule suivante $F_X(x) = P(X \leq x)$. On peut observer en ligne 2 que le modèle dropout est plus incertain que le réseau de neurones classique. En effet, pour chaque jeu d'évaluation (chaque colonne) la courbe orange est sous la courbe bleue. Par exemple pour la ligne 2 colonne 1, on observe que le réseau de neurones à une entropie de 0 pour presque tous les exemples i.e $P(\mathbb{H}_S \leq 0.1) = 1$ alors que le réseau dropout à une fonction de répartition quasi-linéaire, i.e. les éléments ayant une entropie inférieure ou égale à 0.2 (axe des abscisses) ne représentent que 20% de tous les exemples $P(\mathbb{H}_D \leq 0.2) = 0.2$. Cela signifie que le réseau dropout assigne une entropie plus élevée à un plus grand nombre d'exemples que le réseau simple. Cela signifie qu'il est moins certain de ses prédictions pour un grand nombre d'exemples. On peut aussi l'interpréter comme une forme de conservatisme, pour un couple (x, y) donné le réseau dropout aura une probabilité $\mathcal{B}(p)$ avec p plus proche de 0.5 que le réseau simple, en effet plus le paramètre est proche de 0.5 plus l'entropie est élevé. (voir Fig. 5.2 b)

On remarque aussi qu'à mesure que les données deviennent confuses, les deux modèles gagnent en incertitude, de manière générale, cela se voit en observant que les courbes ont tendance à se rapprocher de l'angle en bas à droite sur chaque graphe de la ligne 2.

Cependant, cette évolution est plus visible en observant la fonction de densité.

Quantification de l'entropie avec la une fonction de densité

Les fonctions de densités sont tracées en ligne 3. On observe en colonne 1 que les prédictions du réseau simple sont tellement sûre qu'elles ont toute une entropie de 0, il suffit d'observer l'échelle en ordonnée pour s'en convaincre, on n'arrive même pas à distinguer les nuances de la courbe orange. Cela montre que les modèles se comportent très différemment. Pour rappel, la colonne 1 correspond aux résultats sur les données d'apprentissage. Donc cela peut être toléré. Mais on peut voir d'emblée que le modèle dropout est plus conservateur et que le modèle simple est sur confiant. En observant la courbe orange, on remarque qu'à mesure que les données deviennent confuses la densité de prédictions ayant une entropie élevée augmente. Prenons par exemple la colonne 5. On observe que la grande majorité des prédictions du réseau dropout ont une entropie entre 0.2 et 0.8. Cependant sur le réseau simple la majeure partie des prédictions à une entropie inférieure à 0.2. Cela signifie que le réseau simple garde un excès de confiance sur la majorité des exemples, malgré des données confuses. Alors que le réseau dropout à une grande sensibilité à la confusion.

En conclusion, ces outils permettent de comparer plusieurs modèles avec des données corrompues et de sélectionner un modèle étant capable de demander la confirmation d'un expert lors du déploiement. C'est une approche proposée dans le contexte médical par exemple [Filos et al., 2019]. Cependant, la quantité d'incertitude d'un modèle ne nous informe pas sur sa qualité, en effet un modèle peut-être incertain et se tromper la plupart du temps, ou être incertain et avoir raison la plupart du temps. C'est pour résoudre ce problème que la calibration a été défini, pour avoir une mesure qualitative de l'incertitude des modèles.

5.6.4 Qualité de l'incertitude en classification

Nous avons vu comment quantifier l'incertitude des modèles, c'est un bon début, car cela permet de savoir si les données présentées aux modèles sont conformes aux données d'entraînement. Par exemple, en médecine dans la tâche de détection de tumeur avec des radios du poumon, lorsqu'une image est très différente des données d'entraînement, le modèle exhiberait une plus grande incertitude et un médecin pourrait compléter le diagnostic si besoin. Cependant, cela est insuffisant. Imaginons le scénario suivant : deux radios du poumon sont présentées au modèle, ce dernier prédit un cancer avec une confiance de 55% pour la première radio et 90% pour la seconde. Cependant, la prédiction de la première radio est plus incertaine et devrait demander l'attention d'un médecin pour confirmer le diagnostic, mais pas la seconde radio. Mais qu'en est-il vraiment ? Dans quelle mesure peut-on faire confiance au niveau d"incertitude du modèle ? C'est pour répondre à cette question que nous introduisons la notion de calibration :

La calibration

Définition 5.3. Soit $\{(x_i, y_i)\}_{i=1}^{N}$ un échantillon $i.i.d$ suivant une loi de probabilité $\mathbb{P}$ avec $y \in \{1, \cdots K\}$ la classe, h un modèle probabiliste et $h(x) = (\hat{y}, \hat{p})$ respectivement la classe

prédite et le niveau de certitude. On dit que h est parfaitement calibré si $\mathbb{P}(Y = y|\hat{p} = p) = p \ \forall p \in [0, 1]$.

La calibration nous permet de savoir dans quelle mesure on peut faire confiance au niveau d'incertitude du modèle, en jugeant chaque niveau de confiance. En effet, reprenons notre exemple de détection du cancer avec des radios du poumon. C'est un problème de classification binaire, on peut supposer que x représente donc une radio du poumon et $y \in \{0, 1\}$ représente l'état du patient avec $y = 1$ signifiant que le patient est atteint d'une tumeur. Supposons que $\mathbb{P}(Y = 1|\hat{p} = 0.9) = 0.55$, ou en d'autres termes que seulement un peu plus de la moitié des exemples pour lesquels le réseau prédit une tumeur à 90% ont en réalité une tumeur. Dans ce cas, le niveau de certitude du modèle est exagéré et on dit qu'il est mal calibré. Ainsi, un modèle peut être très performant et mal calibré. Donc si un modèle à une exactitude de 99% en général, mais qu'il est mal calibré, alors on ne peut pas faire confiance à son niveau de certitude. C'est-à-dire que si le modèle prédit qu'une radio du poumon comporte une tumeur à 90% de confiance ou à 70% de confiance, cela revient au même. À l'inverse, si le modèle est bien calibré alors son niveau d'incertitude sera informatif. C'est pour cela qu'on dit que la mesure de la calibration permet d'obtenir la qualité de l'incertitude d'un modèle.

Métriques de mesure de la calibration

Plusieurs approches ont été proposées dans la littérature pour mesurer la calibration. Notons l'erreur moyenne de calibration (*Expected Calibration Error cf ECE*) qui estime l'erreur moyenne de calibration en valeur absolue. En d'autres termes la distance par rapport à l'identité qui est la calibration parfaite :

$$\mathbb{E}_{\hat{p}}\big[|\mathbb{P}(\hat{y} = y|\hat{p} = p) - p|\big]$$

Cette erreur de calibration moyenne peut être estimée avec la formule suivante :

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |\text{exact}(B_m) - \text{conf}(B_m)|$$

Où N est le nombre d'exemples. Aussi, on subdivise l'intervalle $[0, 1]$ en M sous intervalles, ici les différents niveaux de certitude du modèle sont compris dans $[0, 1]$. Notons $(I_m)_{m=1}^{M}$ la suite des sous-intervalles. Puis on répartit les prédictions de toutes les données dans ces sous-intervalles, et note B_m l'ensemble des indices associés aux prédictions de l'intervalle I_m. Alors, on peut calculer l'exactitude moyenne dans chaque sous-intervalle de niveau de certitude avec la formule suivante :

$$\text{exact}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}(\hat{y}_i = y_i)$$

On peut aussi calculer le niveau de certitude moyen dans chaque sous-intervalle :

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i$$

Ainsi, pour revenir à la définition, un modèle est bien calibré lorsque exact(B_m) = conf(B_m). En d'autres termes, le niveau de certitude du modèle sur chaque sous-intervalle correspond au niveau d'exactitude correspondant dans les données.

L'ECE donne une mesure globale de la calibration et permet aussi de comparer des modèles entre eux. Cependant, ce n'est pas un score correct. La négative log-vraisemblance, et le score Brier permettent aussi de mesurer la calibration.

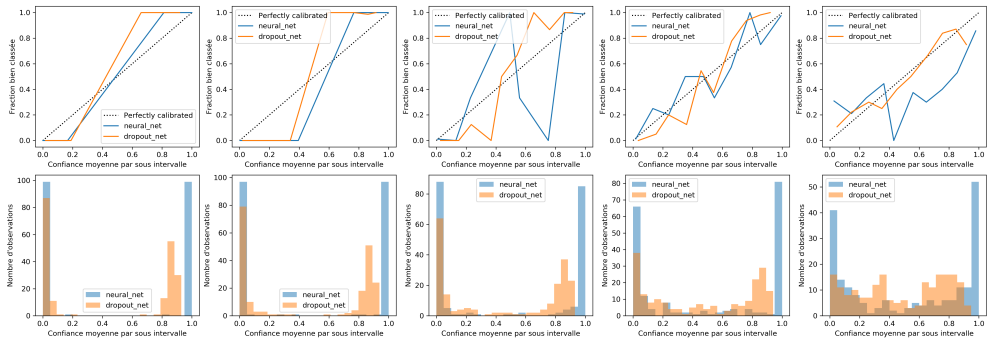
Diagramme de fiabilité

On peut aussi se représenter graphiquement la calibration avec un diagramme de fiabilité comme en figure. 5.4 a Ces diagrammes représentent l'exactitude en fonction de la confiance du modèle. Ainsi, un modèle parfaitement calibré est représenté avec l'identité : en pointillé sur la première ligne de la figure. 5.4 a. Toute déviation par rapport à l'identité est un signe de mauvaise calibration.

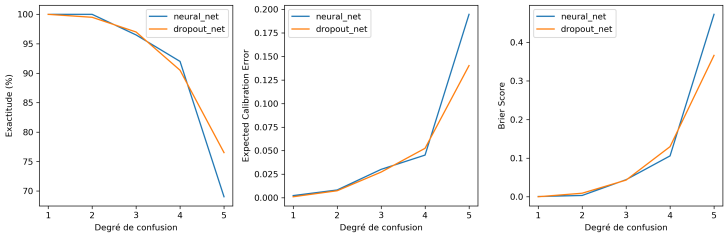
Les colonnes de la figure 5.4 a correspondent aux résultats obtenus sur des données de plus en plus confuses. Ce sont les données présentées dans la première ligne de la figure 5.3. La première ligne de la figure 5.4 a aux diagrammes de fiabilité respectivement du réseau de neurones standard et du réseau de neurones dropout décrit plus haut. Pour interpréter de tels diagrammes, il est utile de tracer aussi l'histogramme des prédictions en deuxième ligne de la figure. 5.4. Ici, on a subdivisé $[0, 1]$ en 10 sous-intervalles de même taille pour tracer ces histogrammes.

En première colonne, on peut voir que les prédictions de chaque modèle sont concentrées autour des extrémités. L'histogramme nous permet d'observer un haut niveau de certitude du réseau simple pour classer chaque point, le réseau dropout est un peu plus incertain. Les deux réseaux ont obtenu une prédiction avec $\hat{p} \sim 0.2$, cependant le diagramme nous montre qu'il se sont tous deux trompés, car la courbe de fiabilité est à 0 pour le sous-intervalle $[0.1, 0.2]$. Cela signifie que les deux réseaux ont eu une prédiction $\hat{p} = p(y = 1|x) \sim 0.2$ c'est-à-dire qu'ils avaient respectivement peu de certitude que le x considéré soit de la classe $y = 1$ alors que c'était le cas. Il semble cependant que lorsque les prédictions sont supérieures à 0.6, les deux réseaux soient toujours corrects. Cependant, le réseau simple est mieux calibré, car il n'a pas de prédiction inférieure à 0.8 pour des exemples ayant la classe $y = 1$, alors que le réseau dropout admet une prédiction correcte à $\hat{p} = 0.6$. Cela signifie qu'il est incertain là où il n'avait pas de raison de l'être.

Cependant, à mesure qu'on augmente la confusion des données, le modèle dropout se retrouve plus proche de l'identité que le modèle standard. Enfin en figure 5.4 b, nous traçons la précision, l'ECE et le score Brier en fonction du niveau de confusion. On voit que la précision chute, et que l'erreur de calibration augmente. C'est pour cela que de nombreux auteurs sont passés par des mesures du niveau d'incertitude pour prédire l'erreur de déploiement en contexte de glissement de données.



(a)



(b)

FIGURE 5.4: a) Diagramme de fiabilité pour chaque niveau de confusion des données. (b) Niveau de généralisation (précision) et métrique de calibration (ECE et score Brier).

5.7 Estimation de l'erreur de déploiement

5.7.1 Estimateurs de l'erreur de déploiement

Récemment, de nombreux chercheurs ont commencé à estimer la généralisation d'un modèle sur un nouveau domaine à partir de données non étiquetées avec des méthodologies théoriques ou empiriques. Par exemple, Jiang[Jiang et al., 2022] a montré que le désaccord d'ensembles bien calibrés est prédictif d'une erreur de généralisation. Guillory et al, et Garg et al. [Guillory et al., 2021, Garg et al., 2022] ont exploité la confiance du modèle pour apprendre un régresseur capable de prédire la généralisation d'un modèle sur un nouveau glissement de données. L'augmentation des données a été utilisée pour renforcer les modèles contre les glissements de distribution avec un certain succès dans le cas des glissement synthétiques [Hendrycks et al., 2019, Hendrycks et al., 2021a]. Mais ces méthodes échouent souvent dans le cas de glissement de données naturelles. Mais l'augmentation des données a également été envisagée pour l'évaluation des performances du modèle en contexte de glissement de données. L'un des premiers auteurs à avoir proposé de telles approches était [Elsahar and Gallé, 2019], ils ont utilisé des distances entre jeux de données qu'ils ont appliqués sur des données augmentées comme proxy pour évaluer l'erreur de généralisation dans le cas de données textuelles. Cette approche a permis aux auteurs de prédire la précision d'un modèle sur un nouveau glissement de données avec une régression linéaire. Certains auteurs ont utilisé des métadonnées caractérisant les sous-populations dans leur ensemble de données pour obtenir une meilleure généralisation hors distribution [Arjovsky et al., 2020, Krueger et al., 2021]. Mais les métadonnées ont également été utilisées pour l'estimation de l'erreur de déploiement. Par exemple, Chen et al. [Chen et al., 2021] a exploité le fait que les praticiens peuvent avoir des connaissances préalables sur le type de glissement de données le plus probable. Ils exploitent les connaissances préalables des praticiens avec des fonctions de découpage ou avec des métadonnées pour estimer la distribution de sous-population dans des données cibles non étiquetées. Ensuite, ils ont estimé la précision du modèle sur les données cibles avec l'ensemble de données source repondéré suivant la distribution de sous-population des données cibles. Deng et al. [Deng et al., 2021] ont utilisé un réseau à deux tâches pour prédire la généralisation hors distribution. Elle consistait en une tâche prétexte prédisant l'angle de rotation d'une image entre (0,90, 180, 270), et une tâche de classification. Ensuite, ils ont montré que la prédiction de la rotation sur la tâche auto-supervisée était corrélée avec la généralisation hors distribution.

Au final, les approches d'estimation de l'erreur de déploiement peuvent être séparées en deux catégories en termes de méthodologies. Le premier groupe [Elsahar and Gallé, 2019, Deng and Zheng, 2021, Deng et al., 2021], utilise une estimation sur la base d'un proxy corrélée à l'erreur de déploiement. L'autre groupe de méthodes [Guillory et al., 2021, Garg et al., 2022, Chen et al., 2021] prédit directement l'erreur de déploiement sur un nouveau domaine.

5.7.2 Erreur de déploiement

Considérons un jeu de données source S et un modèle h_S ayant appris sur S. Considérons maintenant un domaine cible T sur lequel déployer h_S. Alors, l'erreur de déploiement de

h_S sur T est la suivante :

$$DE(h_S, T) = |\mathcal{R}_T(h_S) - \mathcal{R}_S(h_S)| \tag{5.3}$$
$$= |\mathbb{E}_{z\sim T} l_{0-1}(z, h_S) - \mathbb{E}_{z\sim S} l_{0-1}(z, h_S)| \tag{5.4}$$

Avec l_{0-1} la fonction coût associé à la précision. En d'autres termes, l'erreur de déploiement mesure la déviation de l'erreur commise sur les données cibles par rapport à l'erreur commise sur les données sources. Cette définition dépend donc du modèle, du domaine source et du domaine cible. Cela a du sens, car différents modèles entraînés sur les mêmes données sources S peuvent obtenir des chutes de précision différentes lors du passage au domaine cible T.

5.7.3 Estimation de l'erreur de déploiement

Pour estimer l'erreur de déploiement d'un prédicteur h_S sur un domaine cible, de nombreux auteurs [Elsahar and Gallé, 2019, Deng and Zheng, 2021, Deng et al., 2021] utilisent une approche par stress test discutée par [D'Amour et al., 2020] de façon modulaire. Tout d'abord, ils calculent une distance entre la source et les nombreux domaines cibles disponibles $S^{(1)}, \cdots, S^{(n)}$. Deuxièmement, ils apprennent un régresseur pour tirer parti des informations de tous les domaines cibles pour estimer l'erreur de déploiement attendue sur un nouveau domaine inconnu. Nous détaillons ces étapes dans l'algorithme 1.

Data : $S^{(0)} \sim \mathbb{P}$ données sources, $\mathcal{T} = \{S^{(1)}, \cdots, S^{(n)}\}$ un ensemble de jeux de données ayant subis des glissements synthétiques ou naturels, $h : \mathcal{X} \to \mathcal{Y}$ un modèle, $d : (S, S) \to \mathbb{R}$ une distance distributionnelle, $r : \mathbb{R} \to \mathbb{R}$ l'estimateur de l'erreur de déploiement
Result : Estimateur de l'erreur de déploiement entraîné
for $i = 0$ **to** n **do**
  $d_i = d(S, S^{(i)}))$
  $DE_i = |\mathcal{R}_{S^{(0)}}(h) - \mathcal{R}_{S^{(i)}}(h)|$
end
Train r **on** $\{(d_i, DE_i)\}_{i=1}^n$
return r

Algorithme 1 : Apprentissage d'un estimateur de l'erreur de déploiement.

Distances distributionnelles

Ici, nous appelons distance distributionnelle, une fonction d'un couple de jeux de données $d(S, T)$ à valeur dans $\mathbb{R}$. Nous n'investiguons pas cet objet au sens mathématique du terme, car nous souhaitons utiliser une grande variété de ces distances distributionnelles. Le but de ce distances est de jouer un rôle de proxy pour estimer la difficulté de généralisation auquel un modèle h_S entraîné sur S sera confronté en lorsqu'il sera déployé sur T. On pourrait aussi parler de diversité des données T par rapport à S pour un modèle h_S. Cette notion de diversité par rapport à un modèle peut sembler étrange de prime abord, mais nous en discutons en section 7.6. En effet, la diversité est toujours relative à une représentation des données. En pratique, le domaine cible T peut

être composé uniquement des données non étiquetées $T = (T^U)$, c'est le cas de la plupart des algorithmes : [Elsahar and Gallé, 2019, Chen et al., 2021, Deng and Zheng, 2021, Deng et al., 2021, Guillory et al., 2021, Garg et al., 2022]. Mais dans le cas où le modélisateur a accès à des données étiquetées et non étiquetées $T = (T^L, T^U)$ nous pouvons inclure des méthodes comme Optimal Transport Dataset Distance (OTDD) [Alvarez-Melis and Fusi, 2020]. Le calcul de la distance entre les jeux de données est la tâche la plus complexe pour estimer l'erreur de déploiement, car on réduit des informations sur des jeux de données en haute dimension à un scalaire.

Transformer une méthode directe en distance distributionnelle

Average Threshold Confidence (ATC) [Garg et al., 2022] est l'un des algorithmes les plus récents pour estimer directement la précision avec des données non étiquetées du domaine cible. Cet algorithme exploite les niveaux de confiance d'un modèle entraîné sur l'ensemble de données source h_S. Comme ATC estime la précision sur un nouveau jeu de données T, on peut toujours le transformer en distance $d(S,T) = |ATC(h_S, S) - ATC(h_S, T)|$. Et ainsi tirer profit des stress tests pour encore affiner l'erreur de déploiement. Selon Garg et al. [Garg et al., 2022] la méta-distribution des données utilisés pour les stress test $T_1, \cdots, T_n$ peut ne pas représenter le type de glissement de données présents dans la réalité. Cependant, une bonne distance entre les jeux de données doit au moins permettre d'ordonner les jeux de données en fonction de leur difficulté de généralisation. Et cette information est pertinente pour considérer un glissement de données réel. Ainsi, on peut interpréter tous les estimateurs de l'erreur de déploiement avec le cadre de l'algorithme 1.

Régression

De nombreux auteurs ont utilisé une régression linéaire pour estimer l'erreur de déploiement à partir d'une distance entre jeux de données [Elsahar and Gallé, 2019, Deng and Zheng, 2021, Deng et al., 2021]. D'autres auteurs comme Garg et al. [Garg et al., 2022] ont souhaité directement estimer l'erreur de déploiement. Cependant, il est toujours possible d'interpréter ces méthodes comme distance distributionnelles comme on l'a vu pour ATC dans le paragraphe précédent.

5.8 Conclusion

Dans cette revue de littérature, nous avons présenté plusieurs benchmarks pour mesurer les progrès sur la tâche de robustesse. Nous avons vu que les méthodes d'augmentations de données sortent du lot après plusieurs études empiriques. Nous avons ensuite discuté de plusieurs axes de recherches avec l'augmentation de données, et quelques hypothèses concernant son utilité pour renforcer l'invariance des modèles. En nous penchant sur la mesure de la robustesse, nous avons investigués deux approches : la quantification de l'incertitude comme proxy à l'erreur de déploiement et l'estimation directe de l'erreur de déploiement à partir de distances entre jeux de données. Ces outils seront très utilisés au chapitre 7 ou nous allons chercher à comprendre dans quelle mesure l'augmentation de données permettrait d'améliorer la robustesse des modèles. Nous proposerons aussi de quantifier l'incertitude autour de l'estimation de l'erreur de déploiement au chapitre 8. Mais avant

cela, nous présentons un cas pratique de déploiement avec des données acoustiques. C'est ce travail au chapitre 6 qui a éveillé notre intérêt pour ce problème de généralisation hors distribution.

Chapitre Six

Déploiement d'un modèle de classification en contexte de glissement de données

Sommaire

6.1 Introduction

Notre intérêt pour le problème de robustesse lors du déploiement est venu d'un problème d'application de modèles d'apprentissage profond à un glissement de données. À un haut niveau, nous avions deux jeux de données issus de campagnes de collecte de données en mer. Un premier jeu de données collecté en 2011, et un second jeu de données collecté en 2015. Notre objectif était d'apprendre un classifieur sur les données de 2011 et de le déployer sur les données de 2015. Notre surprise est venue d'un résultat étrange. En effet, après avoir entraîné un réseau de neurones convolutif sur le jeu de 2011, nous obtenions une meilleure précision sur le jeu de test de 2015 que sur le jeu de test de 2011. Ici, le principal facteur expliquant un tel phénomène est le glissement de données. Nous allons dans la suite présenter le contexte d'application, le problème d'apprentissage et le résultat d'amélioration des performances en situation de glissement de données. Ce chapitre se concentre principalement sur le déploiement en contexte de glissement de données, cependant le travail complet a été publié dans [Sarr et al., 2021]. Le papier justifiait la sélection d'un réseau de neurones convolutifs sur des données d'acoustiques actives grâce à une étude comparative avec 3 autres modèles : des réseaux de neurones standards, des forêts aléatoires et des machine à support de vecteur. Par ailleurs, nous renverrons le lecteur aux Annexes pour plus de détails.

6.2 Contexte applicatif

L'acoustique des pêches est la principale méthode non-destructive pour estimer l'abondance des poissons pélagiques et semi-pélagiques [Simmonds and MacLennan, 2008, Brehmer, 2006] . Ces estimations sont essentielles pour les pêcheries du monde entier. Elles permettent aux gestionnaires de fournir des recommandations sur l'ajustement de l'effort de pêche pour éviter la surexploitation, maintenir la santé des écosystèmes et assurer la sécurité alimentaire.

6.2.1 Correction de la ligne de fond pour l'évaluation de stock avec l'acoustique active

L'une des premières opérations nécessaires avant d'évaluer l'abondance des poissons est d'identifier la profondeur du fond le long de la trajectoire d'étude [Korneliussen, 2004, MacLennan et al., 2004]. En effet, une correction précise de la ligne de fond est nécessaire, car l'abondance des poissons est estimée en intégrant le signal acoustique du fond à la surface. En effet, si la profondeur du fond estimée est mal calculée et tombe en dessous de la vraie profondeur du fond, l'echo-intégration peut contribuer à des erreurs substantielles, car elle interprétera la quantité d'énergie rétrodiffusée par le fond comme des ressources biologiques [Ona and Mitson, 1996, Villalobos et al., 2013].

De nos jours, les algorithmes de détection de fond avec un échosondeur reposent sur des mesures d'amplitude d'écho avec une plage de profondeurs spécifiée par un opérateur à bord. Ce dernier donne les limites de profondeur supérieure et inférieure les plus susceptibles d'être utilisées lors de la campagne de collecte de données en mer. Cependant, cette procédure peut échouer pour diverses raisons, par exemple, soit à cause d'erreurs dans le réglage manuel de l'instrument, soit à cause du bruit dans le signal réfléchi lui-même qui peut perturber l'algorithme de détection de fond. Par exemple, sur des sols mous et faiblement réfléchissants, le fond peut être détecté en dessous de son niveau réel. De plus, une forte densité de poissons présents près du fond marin peut générer une fausse détection de l'écho de fond [MacLennan et al., 2004].

Ainsi, avant de produire des évaluations de stocks de poissons à partir de ces observations, le signal doit être post-traité manuellement par un expert pour trouver et corriger ces erreurs [Socha et al., 1996]. Cela consiste à examiner visuellement l'ensemble de l'échogramme, rechercher les données avec une mauvaise détection du fond, puis à supprimer les données douteuses et à redéfinir le fond si nécessaire [Bartholomä, 2006]. Cette tâche est coûteuse, car elle nécessite à un expert de parcourir tous les échogrammes, c'est-à-dire plusieurs millions de pings, selon la durée de la croisière. Il s'agit d'un cas typique de processus d'étiquetage complexe ou plusieurs experts peuvent avoir des étiquettes différentes.

6.2.2 Apprentissage machine et aide à la correction de la ligne de fond

Deux approches sont possibles en utilisant les méthodes d'apprentissage : (1) prédire directement la valeur du fond à partir de l'échogramme et (2) évaluer la qualité des pings en les classant en deux groupes, selon que le fond nécessite ou non une correction. La

première approche pourrait être de développer un système pour automatiser entièrement l'intervention humaine, mais les erreurs seraient plus difficiles à détecter. En effet, lorsque la ressource biologique ciblée pour l'écho-intégration nécessite une grande précision, une intervention humaine sera toujours nécessaire. Par ailleurs, la première procédure d'automatisation ne donnerait aucun aperçu à l'expert pour repérer les erreurs éventuelles. Nous avons donc choisi d'aborder le problème d'aide à la correction du fond estimé automatiquement pour faciliter la tâche de post-traitement des données. C'est une tâche basique, mais aussi plus longue, nécessaire à l'évaluation directe des stocks de poissons. Nous avons conçu un système qui aide l'expert à gagner du temps en mettant en évidence les pings ayant une forte probabilité de nécessiter une correction. Ici, l'idée est de définir une méthodologie qui tirerait parti des ensembles de données collectés lors des campagnes précédentes pour soutenir le processus de correction du fond sur un ensemble de données nouvellement collecté. Par conséquent, la tâche d'apprentissage est de fournir une étiquette de qualité pour chaque ping afin que l'expert puisse se concentrer uniquement sur les sections d'échogramme susceptibles de nécessiter une correction.

6.2.3 Apprentissage profond et acoustique de pêche

Étonnamment, peu d'études dans le domaine de l'acoustique des pêches se sont concentrées sur l'amélioration de la détection ou de la correction du fond [Foote et al., 1991], et la plupart se sont concentrées sur la discrimination de la classification des sédiments ou des fonds marins par exemple, [Bartholomä, 2006] et évidemment l'identification des cibles biologiques [MacLennan et al., 2004, Brehmer et al., 2019, Brautaset et al., 2020].

Cependant, seules quelques tentatives d'application de l'apprentissage profond ont été réalisées en acoustique halieutique. Nous citons [Williams, 2016] et [Denos et al., 2017] qui ont d'abord tenté de classer les objets détectés sous l'eau. Cependant, ils étaient confrontés au problème d'une insuffisante disponibilité des données d'entraînement. Récemment, [Brautaset et al., 2020] ont utilisé une architecture d'autoencodeur convolutif pour la détection et la classification acoustiques des bancs de poissons, et ont obtenu des résultats prometteurs, mais toujours confrontés au problème de la qualité de l'ensemble de données d'entraînement. Dans le cas de l'identification des bancs de poissons, surmonter le problème de la qualité du jeu de données d'entraînement est compliqué en raison de la difficulté à obtenir des étiquettes fiables pour les espèces reflétés par le signal [Simmonds and MacLennan, 2008].

6.3 Matériel et méthodes

Dans la section 6.3.1, nous fournissons une description de la collecte de l'ensemble de données brutes. Enfin, en annexe, nous présentons le prétraitement fait en mer, le formatage et la méthode d'étiquetage. Par ailleurs, nous décrivons aussi une expérience comparative pour montrer comment nous avons sélectionné les réseaux convolutifs à une dimension pour la tâche de classification des pings.

6.3.1 Données acoustique active

Acquisition de l'ensemble de données brut

Les données acoustiques proviennent d'une collaboration internationale de centres de recherche halieutique d'Afrique du Nord-ouest, qui ont rassemblé leurs données au niveau sous-régional. Les données consistent en deux ensembles de données correspondant à deux campagnes du projet Nansen (Fisheries Research Vessel Dr. Fridtjof Nansen) qui ont eu lieu en 2011 et 2015 au large du nord-ouest de l'Afrique (Fig 6.1) [Sarré et al., 2018]. Les données ont été acquises avec un échosondeur fixé à la coque du navire. Le navire a envoyé des impulsions d'ondes acoustiques de quatre fréquences distinctes dans l'eau à une durée d'impulsion de 1 ms. Dans cette étude, nous avons utilisé la fréquence de 38 kHz car il s'agit d'une fréquence couramment utilisée dans l'acoustique des pêches et n'est pas limitée au plateau continental (500 à 600 m de profondeur maximale).

Tâche de classification

Ici, chaque impulsion acoustique est appelée un ping, et nous appelons échogramme la matrice obtenue en rassemblant les signaux rétrodiffusés d'une séquence de pings. Les échogrammes que nous pouvons voir en Fig 6.2 proviennent de relevés acoustiques prétraités en mer. La profondeur du fond pour chaque ping a été estimée par un algorithme automatique lors du prétraitement. C'est ce que nous appelons "Bottom" sur la figure 6.2 en rouge. Des erreurs typiques de la procédure automatique de détection de ligne de fond sont illustrées sur la Fig 6.2 a., et on peut voir que l'expert (ligne verte) a grossièrement coupé cette partie pour éviter d'inclure le signal du fond dans l'écho-intégration. Ces pings sont ceux que nous cherchons à identifier dans l'échogramme. En effet de tels pings laissent une grande quantité d'énergie au-dessus du fond (jaune), et cette énergie pourrait être confondu avec de la ressource aquatique par les procédures d'estimation de stock. D'où notre intérêt de signaler ces pings à l'expert en amont de l'estimation de stock. Le but étant de faire gagner à l'expert du temps en l'aidant à se concentrer uniquement sur les pings ayant une forte probabilité d'être corrigé plutôt que de corriger plusieurs million de pings (6.1). Ainsi, le but de l'apprentissage est une tâche de classification binaire, ou on cherchera à exhiber les pings ayant une forte probabilité de nécessiter une correction, des pings ayant une faible probabilité de nécessiter une correction. Le tableau 6.1 décrit les données et la figure 6.3 montre respectivement les distributions des classes des données issues des campagnes de 2011 et 2015.

TABLE 6.1: Jeu de données des relevés en mer 2011 et 2015 après avoir formaté les données (voir Annexe pour plus de détails.). Les données sont des séquences de 2550 niveaux de profondeurs et 1 851 950 pings, chaque valeur représente le niveau d'énergie enregistré. Voir Fig 6.2 pour un shéma.

	Echogram	Label
jeu prétraité 2015 : [No lignes, No colones]	[2550, 1 851 950]	[1, 1 851 950]
jeu prétraité 2011 : [No lignes, No colones]	[2550, 2 321 967]	[1, 2 321 967]

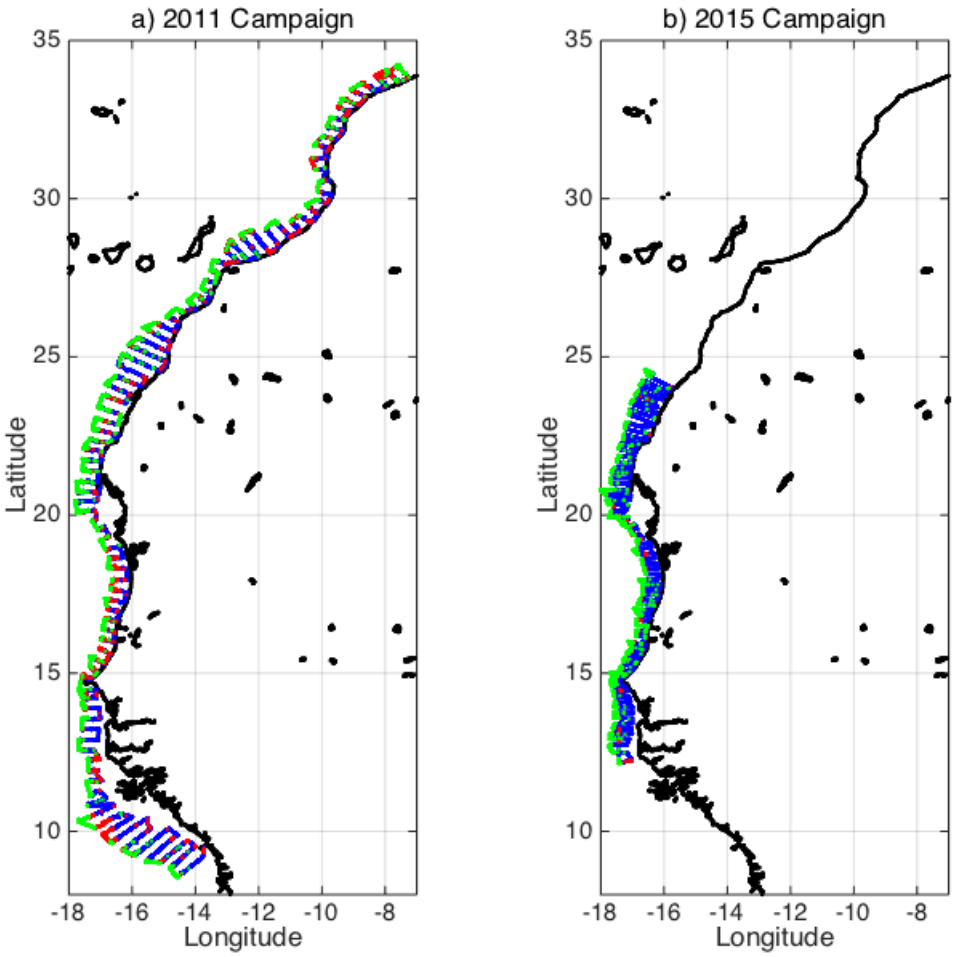


FIGURE 6.1: Campagne réalisée au large du nord-ouest de l'Afrique lors de campagnes d'évaluation acoustique annuelles en mer en 2011 et 2015 (Research Vessel Dr. Fridtjof Nansen). Les classes apparaissent dans l'ordre suivant : en vert, la partie de l'échogramme sans fond ; en bleu, la partie de l'échogramme ne nécessitant qu'une faible correction de la part de l'expert ; en rouge, les pings nécessitant une forte correction de la part de l'expert)

6.3.2 Glissement de données entre 2011 et 2015

L'échogramme brut présente quelques irrégularités dues aux paramètres d'enregistrement embarqués. En particulier, les valeurs NaN (Not a number) étaient généralement présentes entre 500 m (la profondeur d'enregistrement maximale dans cette étude). Mais les NaN sont aussi présente ∼20 à 30 m en dessous du fond prévu. En effet, lors de la collecte des données en mer, le ou les opérateurs de l'échosondeur fixent la profondeur maximale pour limiter l'acquisition des données à la colonne d'eau. Néanmoins, dans certains cas, les valeurs réelles continuent d'exister à des profondeurs sous le fond réel (Fig 6.2). De plus, ces irrégularités étaient inégalement réparties entre les jeux de données 2011 et 2015 (Fig 6.2) car elles dépendaient de la configuration de la campagne en mer. Ces irrégula-

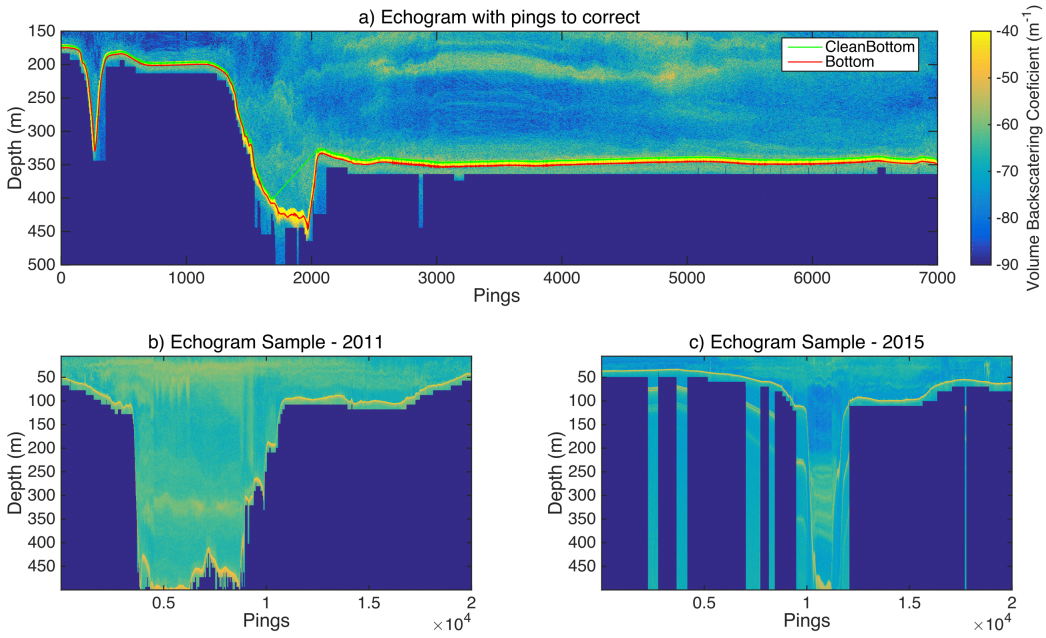


FIGURE 6.2: Échogramme extrait du relevé acoustique en mer utilisé dans l'étude. (a) Un exemple dans lequel la procédure automatique de détection du fond (ligne verte) a échoué vers le ping 2000. Il s'agit typiquement de pings ayant une forte probabilité de nécessiter une correction. (b) Échantillons aléatoires des échogrammes de 2011 et (c) 2015, avec la même taille de ping et le même numéro de cellule ; montrant les différences par rapport aux paramètres des NaN (" pas un nombre " , couleur bleue intense) sous le bas. Unité : voir barre de couleur dans le panneau (a).)

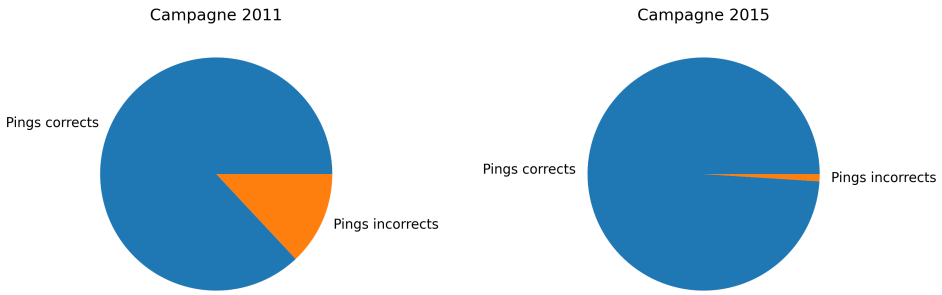


FIGURE 6.3: Distribution des classes pour les données issues des campagnes de 2011 et 2015. On observe que les données de 2015 requièrent moins de correction que les données de 2011. Les pings incorrects sont les pings avec une forte différence entre le fond et le fond corrigé, et les pings corrects sont les pings avec une faible différence entre le fond et le fond corrigé.

rités, courantes en acoustique halieutique, posent généralement problème pour appliquer les approches traditionnelles d'apprentissage. Une autre difficulté est que les classes sont

déséquilibrées d'un jeu de données à un autre comme on peut le voir en figure 6.3.

6.3.3 Cause du glissement de données

Bien que les ensembles de données de 2011 et 2015 aient été collectés dans la même zone, c'est-à-dire le plateau continental nord-ouest de l'Afrique, à l'aide du même navire, il y avait une grande divergence dans la distribution des erreurs de l'estimation initiale du fond, qui a ensuite été corrigée par des experts. Mais il y avait également des irrégularités dans la répartition du bruit des données entre les échogrammes de 2011 et 2015 (Fig 6.2). Ceci peut être dû, par exemple, aux différentes conditions météorologiques rencontrées lors des deux relevés en mer (vent et agitation de la mer). En effet, le signal de réflexion du fond peut être altéré par les bulles d'air (générées par le déferlement des vagues), ainsi que par le roulis et le tangage du navire. La correction du fond dépend également des ressources biologiques ciblées. Par exemple, l'expert corrige plus soigneusement la ligne de fond lorsque l'écho-intégration concerne des ressources proches de la ligne de fond, par opposition aux espèces pélagiques présentes dans la colonne d'eau, car une estimation précise du fond est moins cruciale pour ces dernières. En conséquence, même lors de la correction du fond par l'expert, on peut avoir une partie de l'échogramme corrigée avec moins de précision.

Des erreurs peuvent être causées par la détection de ressources biologiques en contact avec le fond (par exemple, couche planctonique ou banc de poissons) (Fig 6.4 a) ou peuvent survenir lors d'un changement dans les réglages de profondeur du transducteur à bord du navire (Fig 6.4 b).

6.3.4 Modèle

Pour apprendre à effectuer cette classification binaire, nous avons conduit une étude comparative entre quatre modèles :

— les machine à support de vecteur (SVM),

— les forêt aléatoire,

— les réseaux de neurones standards,

— les réseaux convolutifs à une dimension.

Nous avons optimisé les hyperparamètres à l'aide d'un algorithme d'optimisation Bayésien. La meilleure architecture était celle des réseaux convolutifs à une dimension. En annexe, nous décrivons les hyperparamètres choisis, ainsi que la comparaison des modèles. Dans la suite, nous présentons les effets de l'apprentissage inter-domaine avec un réseau de neurones convolutif à une dimension. L'apprentissage consistait à 50 epochs effectué avec l'optimiseur Adam [Kingma and Ba, 2014] et la librairie tensorflow [Abadi et al., 2016].

6.3.5 Apprentissage simple et inter-domaine

Le but de cette expérience se place dans le contexte où on souhaite déployer un modèle entraîné à partir des données de 2011 à des données issues d'une campagne inconnue (les données de 2015). On souhaite mettre en évidence l'effet d'un petit jeu de donné étiqueté de 2015 sur le jeu d'apprentissage issu de 2011. Pour cela, nous allons afficher deux courbes d'apprentissage (Dans la suite K représente 1000 ainsi 100K = 100 000.).

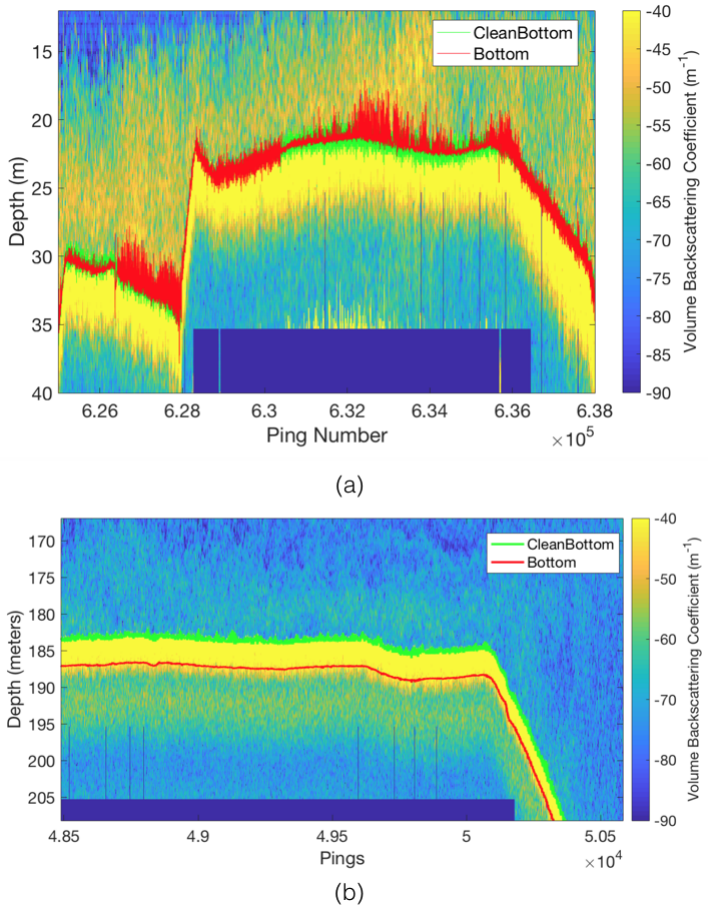


FIGURE 6.4: (a) Exemple d'échogramme dans lequel la détection automatique du fond (ligne rouge) a échoué en raison de la présence d'une couche planctonique directement au-dessus du fond marin (jaune pâle). La ligne verte est la ligne du bas après correction par l'expert. (b) Exemple d'échogramme dans lequel la correction automatique du fond (ligne rouge) a échoué en raison du déplacement de la profondeur du transducteur vers le bas pendant la campagne en mer montrant un décalage constant.

1. une courbe d'apprentissage où le jeu d'apprentissage est composé successivement de 100K pings (tirés de 2011), 300K pings (tirés de 2011) et enfin 550K pings (tirés de 2011). Notés respectivement ST-100K, ST-300K, ST-550K (avec ST pour "simple training", suivi de la quantité de données)
2. une courbe d'apprentissage où le jeu d'apprentissage est composé successivement de 100K pings (tirés de 2011), 300K pings (tirés de 2011) et enfin 550K pings (dont 500K tirés de 2011 et 50K tirés de 2015). Le dernier jeu de données est noté CDT-550K (pour "cross domain training")

Précisons, que les pings de 2011 sont tirés au hasard, car déjà étiquetés. Cependant, comme on considère que le jeu de 2015 vient d'une campagne inconnue, et qu'on veut évaluer nos performances dessus, il faut en étiqueter un sous-échantillon. Pour cela, on tire un sous-échantillon de 100 000 pings successifs de 2015. En effet, l'expert à besoin que les pings se suivent pour discerner la ligne de fond. Sans ordre, il lui serait difficile de percevoir

le fond correctement. Le tableau 6.2 résume les jeux d'apprentissage et de test que nous avons utilisés.

TABLE 6.2: Résumé des données d'apprentissage et de validation

	ST-100K	ST-300K	ST-550K	CDT-550K
jeux d'apprentissage	100K (2011)	300K (2011)	550K (2011)	500K (2011) et 50K (2015)
jeu de test 2011	50K (2011)			
jeu de test 2015	50K (2015)			

6.4 Résultats

TABLE 6.3: Précisions évaluées par chaque modèle sur l'ensemble de données de 2011 et l'ensemble de données de 2015. Il semble que l'ajout d'un sous-ensemble des données de 2015 dans l'ensemble d'entraînement aide le modèle à obtenir de meilleures performances sur les deux ensembles de tests. En gras les précisions supérieure et inférieure.

Précision finales (%)	Jeux d'apprentissage	Jeu de test (2011)	Jeu de déploiement (2015)
ST 100k	91.6	90.5	87.1
ST 300k	92.2	92.2	92.3
ST 550k	92.8	92.8	93.2
CDT 550k	94.8	94.9	95.9

Après avoir entraîné le CNN respectivement avec les jeux ST-100K, ST-300K, ST-550K et CDT-550K, la performance a été évaluée sur le jeu d'apprentissage (ligne bleue continue), un jeu de test extrait de l'ensemble de données de 2011 (pointillés oranges) et un jeu de données de 2015 (pointillés vert) (Voir Fig 6.5 ou tableau 6.3 pour un résumé). Dans la suite, on entre dans le détail de ces résultats, avec essentiellement trois perspectives :

1. L'amélioration de la généralisation (performance sur le jeu test 2011) due à l'augmentation de la taille des jeux de données (ST-100K, ST-300K, ST-550K).
2. L'évolution de l'erreur de généralisation (i.e. différence entre l'erreur d'apprentissage et l'erreur de test 2011) et de l'erreur de déploiement (i.e. différence entre l'erreur de test 2011 et l'erreur de test 2015) pour les jeux de données ST-100K, ST-300K et ST-550K.
3. L'analyse de l'effet de l'apprentissage inter-domaine (CDT-550K) sur les erreurs de déploiement.

6.4.1 Amélioration de la généralisation

Sur le jeu test 2011, le modèle a respectivement classé 90.5 %, 92.2 %, 92.8 % ce qui correspond à une amélioration linéaire de la généralisation. On peut supposer qu'en augmentant simplement la taille des données, nous allons simplement obtenir un modèle plus performant sur le jeu de 2011. Ce qui va dans le sens de la réduction de l'erreur réelle

par la réduction de l'erreur d'estimation ε_{est} discuté au chapitre 2. C'est aussi le résultat de la loi des grands nombres, i.e. plus on a de données, plus précise est notre estimation et donc plus faible est notre erreur.

6.4.2 Évolution des erreurs de généralisation

L'apprentissage sur ST-100K entraîne un sur-apprentissage, en effet l'erreur de généralisation est de 0.6 %. Cela s'explique, car nous n'avons pas appris avec assez de données. L'erreur de déploiement sur 2015 est encore plus élevée avec un gap de 3.7 %. Apprendre avec ST-300K permet d'éviter le sur-apprentissage, en effet l'erreur de généralisation est nulle. Cela signifie que le modèle généralise bien sur la distribution de 2011 dès que la taille de l'échantillon atteint 300K pings. L'erreur de déploiement sur 2015 est aussi très faible (0.1 %). Augmenter le jeu d'apprentissage à ST-550K permet d'augmenter la performance par rapport à ST-300K comme discuté plus haut, tout en évitant le sur-apprentissage. On peut penser qu'à partir d'un certain seuil dans la taille des données il n'y a plus de sur-apprentissage. Cependant, on remarque que l'erreur de déploiement sur 2015 augmente (0.4 %) par rapport à ST-300K.

6.4.3 Analyse de l'apprentissage inter-domaine

Avant d'aller plus loin, observons que le jeu d'apprentissage CDT-550K n'est identiquement distribué à aucun des jeu de test (ni celui de 2011, ni celui de 2015). Donc, ici, on ne peut pas vraiment parler d'erreur de généralisation sans un jeu test identiquement distribué à CDT-550K, cependant, on va faire l'hypothèse que l'erreur sur un éventuel jeu test CDT-550K serait identique à l'erreur d'apprentissage CDT-550K. En effet, on a observé qu'il n'y avait plus d'erreur de généralisation dès que la taille du jeu d'apprentissage dépassait 300K pings. En d'autres termes, on considère que l'erreur d'apprentissage est égale à l'erreur de test sur CDT-550K. Ainsi, on suppose qu'il y a donc deux erreurs de déploiement associées respectivement aux jeux test de 2011 et de 2015. Ainsi, on observe une erreur de déploiement de 0.1 % sur le jeu test de 2011, et une erreur de déploiement de 0.6 % sur le jeu test de 2015.

6.5 Discussion

Ici, il est intéressant d'observer l'évolution de l'erreur de déploiement sur le jeu de 2015 allant de 3.7 % après apprentissage sur ST-100K, à 0.1% après apprentissage sur ST-300K, à 0.4 % après apprentissage sur ST-300K pour finir à 0.6 % après apprentissage sur CDT-550K. Il y a deux points intéressants à mettre en lumière qui sont liés au fait que les données d'apprentissage et de déploiement ne sont pas identiquement et indépendamment distribuées.

Premièrement, on observe que pour les jeux d'apprentissages ayant plus de 300K pings, les performances sur le jeu de déploiement sont supérieures aux performances sur le jeu d'apprentissage. Ce résultat serait très étrange si les données étaient *i.i.d*, en effet revoir la section décomposition de l'erreur au chapitre 2 pour s'en convaincre. En effet, lorsque l'hypothèse *i.i.d* est vérifiée l'erreur d'apprentissage est toujours plus petite que l'erreur

FIGURE 6.5: Courbe d'apprentissage obtenu en évaluant les modèles sur respectivement les jeux d'apprentissage (ligne bleue) le jeu de test 2011 (ligne pointillée orange) et le jeu test de 2015 (ligne pointillée verte). Les modèles ayant été entraînés sur les données d'apprentissage figurant en abscisse.

de test. Ce qui est normal car le modèle a l'opportunité de sur-apprendre les données d'entraînement. Ici, une performance supérieure sur le jeu de déploiement signifie que le jeu de déploiement est plus facile à classifier que le jeu d'entraînement. Ou en d'autres termes, la difficulté de généralisation est négative pour utiliser les notions que nous avons introduites en section 5.7. Deuxièmement, on constate qu'un apprentissage inter-domaine permet un bon de généralisation sur les distributions issue de 2011 et de 2015. En effet, on observe une performance passant de 92.8 % à 94.9 % sur le jeu test 2011, soit un bon de 2.1 %. À titre de comparaison passer d'un jeu d'apprentissage de 100K à 550K à permis d'augmenter la généralisation de 90.5 % à 92.8 %, soit un bon de 2.3 %. Ce qui corrobore avec les travaux de Taori et al [Taori et al., 2020]. En effet, selon eux, l'augmentation de la taille des données à des performances décroissante comparée à une augmentation de la diversité des données. Cela conforte davantage notre choix d'explorer l'augmentation de données pour mieux comprendre le problème de robustesse des modèles.

6.6 Conclusion

Le problème spécifique que nous avons abordé est la tâche d'aider un expert à étiqueter un nouveau jeu de données (2015) plus rapidement grâce à un modèle entraîné sur un jeu de données récolté antérieurement (2011). Le but étant d'aider l'expert dans la correction du fond des échogrammes en amont de l'estimation du stock de poissons. La modélisation consistait à évaluer le modèle entraîné sur les données de 2011 sur les données de 2015. En effet, en situation réelle, la correction du fond se fait sur des données fraîchement collectées et encore non étiquetées. Donc cet exercice nous permet d'étudier une situation de déploiement en contexte de glissement de données.

Une hypothèse importante en apprentissage machine est que les données de déploiement doivent être identiquement et indépendamment distribuées aux données d'apprentissage (hypothèse $i.i.d$) [Goodfellow et al., 2016]. Cependant, c'était loin d'être le cas pour nos données comme nous le décrivons à la section 6.3.2 autant pour la distribution des étiquettes (Fig 6.3) que pour la distribution du bruit (Fig 6.2). Nous sommes donc en terrain inconnu en ce qui concerne les performances de généralisation attendues. Comme on l'avait vu au chapitre 4, il s'agit d'une situation d'un glissement général de données hors glissement covarié et hors glissement de concept. Selon Moreno et al. [Moreno-Torres et al., 2012], cette situation est rare et difficile à résoudre. Nous sommes d'accord que cette situation est difficile à résoudre, mais à notre avis, cette situation est extrêmement courante. C'est d'ailleurs la motivation principale de cette thèse, nous pensons que l'apprentissage profond sera surcoté tant qu'on ne pourra pas garantir la fiabilité des réseaux de neurones pour un grand nombre de situations. Dans le chapitre 7, nous essaieront de comprendre les effets de la diversité des données sur l'erreur de déploiement.

Chapitre Sept

Invariance et robustesse en apprentissage machine

Sommaire

7.1 Introduction

Ce chapitre présente une réflexion qui globalement cherche à répondre à 2 grandes questions de recherches :

— pourquoi l'augmentation de données améliore la généralisation ?

— Dans quelle mesure l'augmentation de données peut être utile à la généralisation hors distribution ?

Les trois premières expériences cherchent à comprendre la magnitude du gain de généralisation qui peut être fait avec l'augmentation de données à travers différents scénarios. Ensuite, les expériences 4,5,6,7 évaluent l'effet de divers types d'augmentation de données pour rendre les modèles plus robustes aux glissements de données. Nous comprenons précisément comment l'augmentation de données est limitée pour la généralisation hors distribution.

7.2 Augmentation de données et estimation de densité

Le but de ce travail était de créer un système de tutorat intelligent pour enseigner l'écriture, le travail a été publié [Sarr et al., 2020]. Ici, nous voulons apprendre une distribution de lettres manuscrites dans une situation avec peu de données. Et on veut voir l'effet de l'augmentation de données sur la capacité à apprendre la distribution des données.

Question de recherche Quel est l'effet de l'augmentation de données en ligne sur l'estimation de densité.

7.2.1 Matériel et méthodes

Modèle expert et générativité Dans ce travail, nous avons utilisé un autoencodeur variationnel ("Variational Autoencoder" cf VAE) [Kingma and Welling, 2013] ayant appris à générer un jeu de données UCI open source de dessin de lettres [Llorens et al., 2008]. Huit lettres sont implémentées : a, b, c, e, m, p, r, s. Semblable à un auto-encodeur standard [Goodfellow et al., 2016], un VAE a deux composants : un encodeur et un décodeur. L'encodeur apprend une représentation compressée d'un ensemble de données dans un espace latent [Kingma and Welling, 2013]. Le décodeur apprend à reconstruire l'ensemble de données à partir de la représentation compressée. Pour le VAE l'espace latent est modélisé avec une loi de probabilité, ce qui permet d'échantillonner de nouvelles données. De cette façon, nous nous assurons que le bruit introduit dans le processus de génération est similaire à la variabilité naturelle des dessins humains.

Données Nous avons choisi huit jeux de données de lettres manuscrites correspondants à : a, b, c, e, m, p, r, s. Ces jeux de données sont issus de la base de données open source UCI [Llorens et al., 2008] de dessin de lettres. Chaque lettre n'a que 120 exemples pour 60 grapheurs différents (deux exemples différents par grapheur). Et certaines lettres présentent une plus grande diversité que d'autres.

Sketch-RNN Sketch-RNN [Ha and Eck, 2018] est un autoencodeur variationnel ("Variational Autoencoder" cf VAE) [Kingma and Welling, 2013] qui utilise une architecture de réseau de neurones récurrents (RNN) et a été conçue pour apprendre sur des séquences de croquis de personnes tels que des oiseaux, des fruits, des camions, etc. Les données sont formatées au format "stroke-5", où les deux premiers paramètres correspondaient aux coordonnées x et y, les trois autres paramètres p_1, p_2, p_3 désignent l'état du stylo :

— p_1, le stylo touche actuellement le papier.

— p_2, le stylo sera levé du papier après le point actuel.

— p_3, le dessin est terminé.

Sketch-RNN a utilisé un RNN bidirectionnel comme réseau d'encodage et un RNN standard comme décodeur. Afin de modéliser une sortie continue multivariée, par ex. Pour les coordonnées x et y, le Sketch-RNN original [Ha and Eck, 2018] utilise une mixture de fonctions gaussiennes bivariées [Bishop, 1994], qui se sont avérées utiles pour modéliser et synthétiser l'écriture manuscrite [Graves, 2013]. Enfin, comme les RNN ont du mal à capturer les

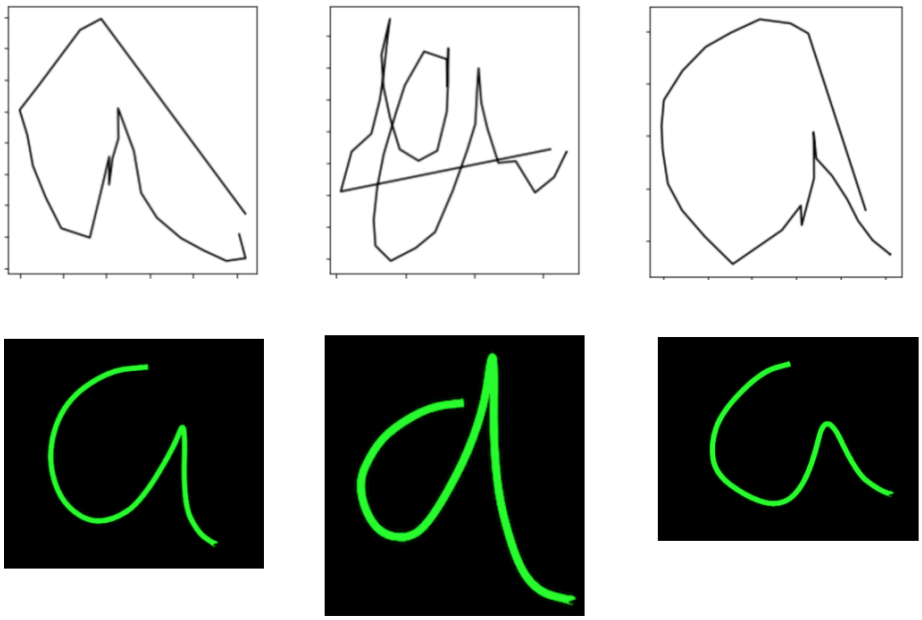


FIGURE 7.1: Effet d'une augmentation de données appropriée sur la génération avec peu d'exemples

dépendances à long terme, l'architecture LSTM [Hochreiter and Schmidhuber, 1997] a été utilisée à la place d'un RNN simple. Nous avons utilisé une version simplifiée de sketch-RNN, avec deux modifications mineures. L'architecture GRU [Chung et al., 2014] est utilisée à la place de LSTM pour réduire la complexité globale du modèle. Les GRU nécessitent moins de paramètres, mais sont compétitifs par rapport aux LSTM [Chung et al., 2014]. Deuxièmement, Djehuty n'utilise que des matrices de covariance diagonales au lieu d'utiliser des matrices de covariance complètes dans le mélange de fonctions gaussiennes. Ce choix a été fait suite à l'argument [Bishop, 1994] de l'article original selon lequel un modèle de mixture Gaussien avec covariance diagonale peut approximer n'importe quelle fonction de densité avec une précision arbitraire à condition que le coefficient de mélange et les paramètres Gaussiens (moyennes et variance) soient correctement choisis. Nous nous référerons toujours au modèle simplifié décrit ci-dessus sous le nom de Sketch-RNN, car il ne diffère que légèrement de la mise en œuvre d'origine.

Augmentation de données Nous avons comparé Sketch-RNN avec ou sans l'utilisation d'une technique d'augmentation de données. Cette technique consistait à générer des paramètres (δ_x, δ_y) suivant une loi uniforme $\mathcal{U}[0.9, 1.1]$ et à ajouter ces paramètres aux coordonnées (x, y) de chaque exemple lors de l'apprentissage. Cela avait pour effet de tordre les lettres dans la direction (δ_x, δ_y) qui changeait à chaque itération pour toutes les lettres dans le lot.

7.2.2 Résultats

On observe dans la figure 7.1 deux lignes, la première ligne correspond à des lettres générées par Sketch-RNN lorsque le modèle était entraîné sans augmentation de données. Et

la deuxième ligne correspond à des lettres générées par Sketch-RNN lorsque le modèle était entraîné avec l'augmentation de données. On peut observer que les lettres de la deuxième ligne sont plus réalistes. La différence de couleur vient du fait que le modèle que nous avons intégré dans l'application mobile (Voir [Sarr et al., 2020] pour les détails.) utilisait l'augmentation de données (deuxième ligne). Alors que le modèle entraîné sans augmentation de données n'a jamais été déployé.

7.2.3 Conclusion

Nous observons que lorsque la base de données est très petite, une augmentation de données pertinente peut faire la différence entre un modèle réaliste et un modèle irréaliste. Nous voulons exprimer que cette expérience nous a tellement impressionné qu'elle motive en grande partie notre désir de mieux comprendre le fonctionnement de l'augmentation de données en général. En effet, que signifie une augmentation de données pertinente, pourrait-on utiliser cette intuition en classification ? C'est ce que nous avons fait dans les sections suivantes.

7.3 Stabilité des techniques induisant l'invariance et complexité d'échantillonage

Dans cette expérience, nous avons voulu comparer trois types de méthodes générant de l'invariance : augmentation de données en ligne et hors ligne, et utilisation d'un a priori invariant. Le but était de voir leur effet sur la généralisation et sur la stabilité de l'apprentissage. La stabilité comme dans l'analyse des perturbations fait référence à la variabilité du résultat d'apprentissage obtenu par un changement dans la procédure d'apprentissage [Bousquet and Elisseeff, 2002]. Ici, le résultat d'apprentissage considéré est le niveau de généralisation mesuré par la précision sur le jeu test. [1]

Question de recherche Est-ce que l'incertitude introduite par un a priori invariant permettrait de mieux généraliser et donc de régulariser l'a posteriori comparé à l'augmentation de données classique ?

7.3.1 Matériel et méthodes

Les méthodes d'invariance ont été appliquées sur les augmentations de données suivantes : rotation, bruit uniforme et rotation + bruit uniforme sur l'ensemble. La rotation de l'image était échantillonnée entre $[-20°, 20°]$; un bruit uniforme entre $[0, 255]$ a été ajouté à l'image. La dernière augmentation était simplement une combinaison de rotation et de bruit.

Pour évaluer l'effet des différentes formes d'augmentation, quatre réseaux de neurones Bayésiens ont été utilisés. Les trois premiers BNN impliquaient une loi Normale centrée et entièrement factorisée comme a priori $\mathcal{N}(0, 1)$ et l'apprentissage a posteriori s'est fait

1. Ce travail a été accepté à la conférence "Second International Conference, PAAISS 2022, Dakar, Senegal, November 2-4, 2022".

respectivement avec : aucune augmentation (cas contrôle), augmentation de données hors ligne avec un facteur d'augmentation de 8 et augmentation de données en ligne. Le dernier réseau de neurones Bayésien a utilisé un a priori approximativement invariant et l'a posteriori a appris sans aucune augmentation de données. L'a priori approximativement invariant approximatif a été entraîné avec 2000 itérations avec un arrêt précoce après 500 époques sans progression. La taille de l'échantillon a été fixée à 1024 et le facteur d'augmentation à 64. Enfin, l'hyperparamètre devant l'entropie était $\frac{i}{2000}$, où i était l'itération actuelle.

Les quatre réseaux de neurones Bayésiens ont été comparés sur les jeux de données MNIST et Fashion MNIST [Deng, 2012, Xiao et al., 2017].

De plus, pour avoir une base de comparaison, nous avons fixé la quantité de calcul disponible pour chaque modèle en termes de mini-lots traités. Par conséquent, 800 itérations ont été utilisées pour chaque modèle, à l'exception de l'augmentation de données hors ligne qui a nécessité seulement 100 itérations, car son ensemble de données était 8 fois plus volumineux, et donc entraînait 8 fois plus de mises à jour des paramètres. La taille du lot pour chaque modèle a été fixée à 1024, et nous avons calculé des moyennes de Monte Carlo avec 30 propagations par lot de données pour calculer la borne inférieure moyenne (ELBO). Enfin, chaque modèle a été testé avec l'ensemble de test respectif des ensembles de données MNIST et Fashion MNIST. Pour obtenir la prédiction sur l'ensemble de test, 100 passages par lot de données ont été effectués pour calculer les prédictions, et nous avons exécuté l'évaluation 10 fois pour obtenir la moyenne et l'écart-type pour chaque modèle.

7.3.2 Résultats

Les résultats sont résumés dans le tableau 7.1. Nous observons que l'utilisation d'a priori approximativement invariants n'a jamais donné de moins bons résultats que le modèle contrôle. Dans l'ensemble, il a produit les meilleures améliorations de généralisation. De plus, il faut à peu près autant de temps pour s'entraîner que pour le réseau de neurones Bayésien contrôle. L'augmentation en ligne prend le plus de temps pour entraîner l'a posteriori et est la méthode la moins stable, par exemple, elle a échoué plusieurs fois : c'est le cas avec le bruit sur Fashion MNIST, la rotation et le bruit sur MNIST et Fashion MNIST. Nous expliquons que l'augmentation en ligne modifie considérablement la distribution des données, car beaucoup plus de bruit est ajouté au processus d'apprentissage, ce qui conduit probablement le processus d'optimisation à être plus facilement piégé dans des minima locaux. Par conséquent, l'augmentation de données en ligne peut nécessiter une stratégie d'optimisation plus prudente pour obtenir de bons résultats. Pourtant, cette dernière approche a obtenu la meilleure précision de généralisation pour le bruit sur Fashion MNIST. L'augmentation de données hors ligne est plus rapide à former et plus stable comparé à l'augmentation de données en ligne, il obtient le meilleur résultat pour la rotation sur MNIST.

7.3.3 Conclusions

Nous apprenons que les a priori approximativement invariants permettent de transférer efficacement l'information d'une corruption donnée à un modèle. En effet, il suffisait d'un facteur 64 appliqué sur un échantillon de 1024 exemples pour que l'a priori apprenne à être

TABLE 7.1: Comparaison des différents paramétrages pour l'apprentissage a posteriori. N.A signifie qu'aucune augmentation est appliqué lors de l'apprentissage a posteriori.

Rotation - MNIST			
Prior	DA	Accuracy (%)	Time (s)
$\mathcal{N}(0,1)$	Online	96.7 ± 0.2	11726
$\mathcal{N}(0,1)$	Offline	**96.9 ± 0.1**	7649
$\mathcal{N}(0,1)$	N.A	96.8 ± 0.1	10483
Invariant	N.A	96.8 ± 0.1	10715
Noise - MNIST			
Prior	DA	Accuracy (%)	Time (s)
$\mathcal{N}(0,1)$	Online	95.3 ± 0.1	17276
$\mathcal{N}(0,1)$	Offline	79.5 ± 0.1	7660
$\mathcal{N}(0,1)$	N.A	96.8 ± 0.1	10483
Invariant	N.A	**97.0 ± 0.1**	10411
Rotation+Noise - MNIST			
Prior	DA	Accuracy (%)	Time (s)
$\mathcal{N}(0,1)$	Online	19.8 ± 0.6	18934
$\mathcal{N}(0,1)$	Offline	82.2 ± 0.2	7643
$\mathcal{N}(0,1)$	N.A	96.8 ± 0.1	10483
Invariant	N.A	**97.0 ± 0.1**	10431
Rotation - Fashion MNIST			
Prior	DA	Accuracy (%)	Time (s)
$\mathcal{N}(0,1)$	Online	**77.5 ± 0.3**	11529
$\mathcal{N}(0,1)$	Offline	76.4 ± 0.5	7554
$\mathcal{N}(0,1)$	N.A	44.3 ± 0.2	10476
Invariant	N.A	74.7 ± 0.2	10659
Noise - Fashion MNIST			
Prior	DA	Accuracy (%)	Time (s)
$\mathcal{N}(0,1)$	Online	26.5 ± 0.3	16951
$\mathcal{N}(0,1)$	Offline	78.4 ± 0.3	7593
$\mathcal{N}(0,1)$	N.A	44.3 ± 0.2	10476
Invariant	N.A	**83.3 ± 0.3**	10221
Rotation+Noise - Fashion MNIST			
Prior	DA	Accuracy (%)	Time (s)
$\mathcal{N}(0,1)$	Online	21.1 ± 0.3	18600
$\mathcal{N}(0,1)$	Offline	70.6± 0.6	7570
$\mathcal{N}(0,1)$	N.A	44.3 ± 0.2	10476
Invariant	N.A	**83.8 ± 0.2**	10290

suffisamment invariant pour être compétitif face à l'augmentation de données hors ligne et en ligne. L'intérêt de cette réalisation est que la quantité de corruption à injecter durant l'apprentissage pour rendre un modèle invariant est relativement limitée. Pareillement, il n'est pas nécessaire d'appliquer cette augmentation à tous les exemples du jeu de données.

7.4 A priori invariant et apprentissage actif

Dans la première expérience, nous voulions confirmer que l'a priori invariant apportais une information utile à la généralisation. Désormais, nous souhaitons savoir à quel point l'information d'un a priori invariant peut booster l'apprentissage de l'a posteriori dans un contexte de petites données en induisant une meilleure qualité de l'incertitude par exemple. Pour ce faire, nous allons utiliser l'apprentissage actif [Settles, 2010].

Question de recherche Donc, avec ce cadre, est-ce que l'a priori invariant peut apporter une information comparable à une fonction d'acquisition ?

7.4.1 Matériel et méthodes

Pour tester cela, nous avons suivi le même formalisme que [Gal et al., 2017]. Nous avons effectué 100 itérations, nous commençons avec un jeu de 10 exemples pris au hasard (x, y). Puis avant chaque itération, nous apprenons l'a priori invariant et l'a posteriori avec la méthode Flipout [Wen et al., 2018]. (1) L'a priori apprend à être invariant au 15 corruptions de la figure 7.3 appliquées aux 10 données du jeu. (2) Puis nous apprenons l'a posteriori sur les données avec le jeu de 10 exemples. Puis, nous re-sélectionons 10 exemples au hasard, et effectuons le même procédé jusqu'à remplir le jeu avec 1000 exemples. Pour avoir matière à comparer, nous avons appris en utilisant trois modes d'apprentissages :

1. A priori normal centré réduit (Chaque neurone suit une loi Gaussienne centré réduite $\mathcal{N}(0, 1)$.), et pas de fonction d'acquisition.
2. A priori invariant, et pas de fonction d'acquisition.
3. A priori normal centré réduit, et utilisation de la fonction de maximisation de l'entropie comme fonction d'acquisition.

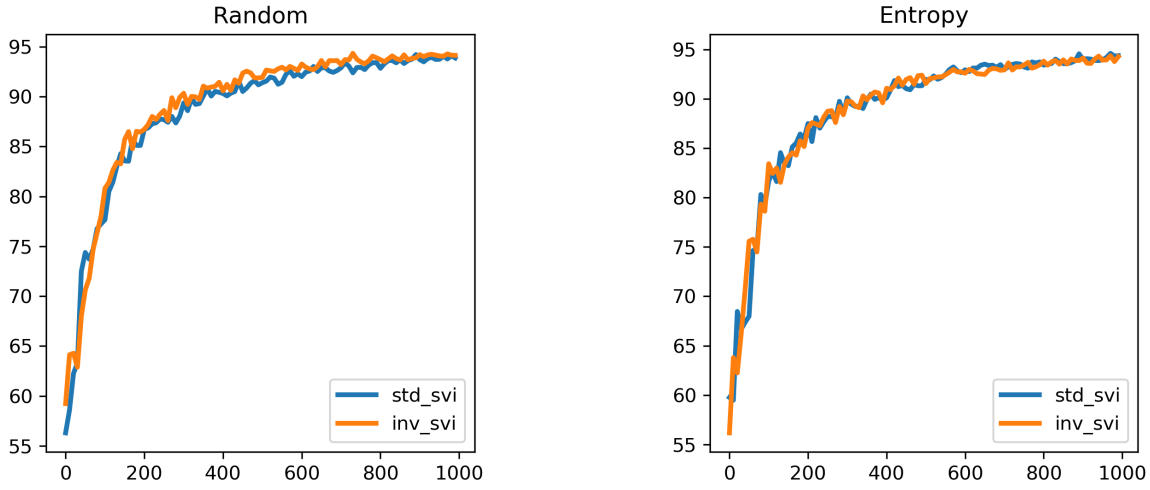
Les résultats sont décrits dans la figure 7.2

7.4.2 Résultats

Notre première expérience sur l'apprentissage actif nous montre que l'a priori invariant permet un échantillonnage plus efficace. En effet la figure 7.2 montre la croissance de la généralisation obtenue sur un jeu test séparé de 10 000 exemples. Ainsi, nous avons les résultats lorsque l'apprentissage n'était contraint qu'à 10 exemples, puis 20, et ainsi de suite jusqu'à 1000 exemples.

Sur la figure 7.2 a, la courbe bleu représente un réseau de neurones Bayésien avec a priori Gaussien centré réduit. La courbe orange représente la performance obtenue par un réseau de neurones Bayésien incluant un a priori apprenant à être invariant aux 15 corruptions décrites en figure 7.3 à chaque itération. Cette figure 7.2 a, montre l'évolution

des performances avec un échantillonnage aléatoire à chaque itération, tandis que la figure 7.2 b montre les résultats dus à un échantillonnage sélectionnant pour l'a posteriori les exemples maximisant son entropie.



(a) (b)

FIGURE 7.2: Apprentissage actif : chaque courbe est obtenue avec un réseau de neurones Bayésien. La couleur bleue est obtenue avec un a priori Gaussien centré réduit, et la courbe orange avec un a priori invariant. (a) Performances obtenues avec la fonction d'acquisition échantillonnant les données au hasard. (b) Performances obtenues avec la fonction d'acquisition maximisant l'entropie des données échantillonnées.

7.4.3 Conclusion

On peut constater que dans le cas d'un échantillonnage aléatoire, le modélisateur gagne à incorporer sa connaissance a priori sur les invariances durant chaque itération, car cela améliore les performances par rapport à un a priori non-informatif. Cela confirme le fait que sur des petites données une bonne connaissance a priori permet d'obtenir une meilleure généralisation. Cependant, on observe que la stratégie d'inclure cette connaissance a priori dans l'apprentissage actif est au mieux comparable à l'utilisation traditionnelle d'une fonction d'activation. Nous pensons que cela est dû au fait que la connaissance a priori ne change pas beaucoup les données ayant le plus d'entropie lors de l'acquisition. C'est comme si cette connaissance n'était que marginale pour la fonction d'acquisition.

7.5 Type d'invariance et robustesse

Question de recherche Est-ce que l'incertitude introduite par un a priori invariant permettrait de mieux généraliser hors distribution ? En d'autres termes, est-ce que la variance

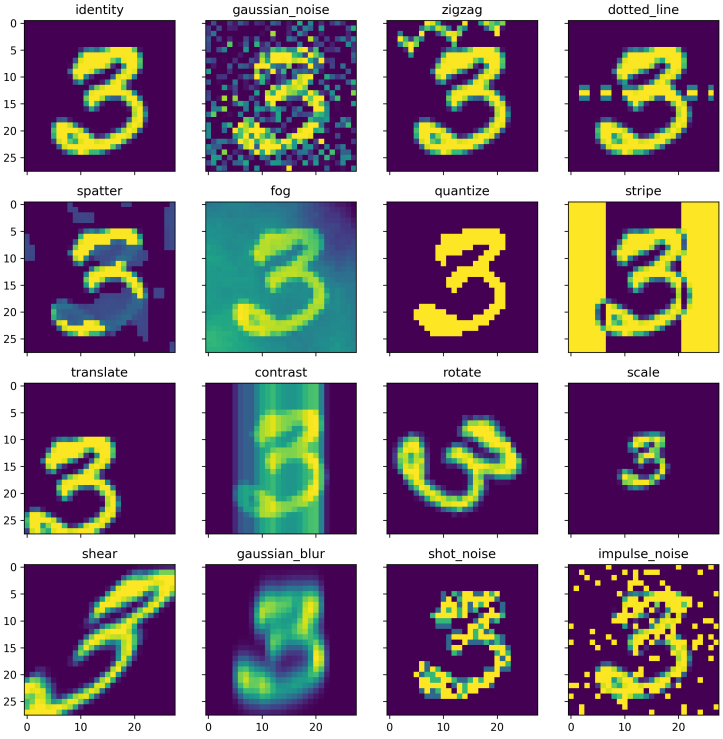


FIGURE 7.3: Ensemble des corruptions utilisées sur MNIST. Les titres correspondent au nom des corruptions.

autour des poids de l'a priori invariant contient une information utile pour généraliser hors distribution ?

7.5.1 Matériel et méthodes

Pour savoir cela, nous proposons d'utiliser un jeu de données MNIST corrompu, voir figure 7.3. Il s'agit du jeu original MNIST avec 15 corruptions ajoutées.

On considère que nous avons un ensemble de domaines D comprenant les 15 corruptions, lors de l'apprentissage, on échantillonne une méta distribution d'apprentissage D_{tr} composé de 10 domaines, et une méta distribution de test D_{ts} composée de 5 domaines. L'augmentation est faite en ligne durant l'apprentissage. Dans chaque expérience, nous utilisons l'architecture LeNet-5 avec 5 types d'apprentissages différents :

1. Softmax : ici, on utilise un réseau déterministe pour apprendre D_{tr}
2. Dropout : ici, on apprend en utilisant Monte-Carlo Dropout qui modélise un réseau de neurones Bayésien avec l'inférence variationnelle [Gal and Ghahramani, 2016]. L'avantage de cette méthode, est qu'elle est facile à implémenter et ne demande pas d'implémenter l'a priori, en effet cette dernière suppose une loi Gaussienne centrée

réduite pour chaque neurone de l'a priori.

3. Stochastic Variational inference (SVI), nous utilisons la méthodologie Flipout [Wen et al., 2018] pour l'apprentissage des réseaux a priori et a posteriori. Et nous testons trois cas de figure en modélisant l'exposition au jeu du réseau de neurones Bayésien (a priori, a posteriori) aux corruptions D_{tr} par un couple de booléens :
   — True False : on apprend un a priori invariant aux corruptions de D_{tr}, mais l'a posteriori apprend uniquement sur le jeu ordinaire.
   — False True : on utilise un a priori Gaussien centré réduit, et on augmente les données avec les corruptions D_{tr} durant l'apprentissage de l'a posteriori.
   — True True : on apprend un a priori invariant aux corruptions D_{tr}, et on augmente les données avec les corruptions D_{tr} durant l'apprentissage de l'a posteriori.

Lorsque les modèles étaient tous entraînés, nous avons évalué 4 métriques : l'erreur moyenne de corruption (mCE), l'entropie, et la calibration avec le score de Brier et l'ECE pour reprendre les métriques proposées par Ovadia (2019) [Ovadia et al., 2019]. Pour réduire la variabilité des résultats nous avons utilisé la validation croisée [Goodfellow et al., 2016] à 3 blocs. C'est-à-dire que D a été partitionné en trois groupes de 5 domaines : $D = \{D_1, D_2, D_3\}$ et l'apprentissage a eu lieu trois fois successivement avec :

1. $D_{tr} = \{D_1, D_2\}$ et $D_{ts} = \{D_3\}$.
2. $D_{tr} = \{D_1, D_3\}$ et $D_{ts} = \{D_2\}$.
3. $D_{tr} = \{D_2, D_3\}$ et $D_{ts} = \{D_1\}$.

Enfin, les résultats de 3-blocs ont été agrégés et présentés en figure 7.4 et 7.5.

7.5.2 Résultats

Nous avons une comparaison d'un réseau déterministe avec deux approximations Bayésiennes : Dropout [Gal and Ghahramani, 2016] et Flipout [Wen et al., 2018] en figure 7.4 et 7.5. Pour toutes les figures :

— la couleur bleue représente les performances obtenues sur le jeu de test sans corruption,
— la couleur orange correspond aux performances obtenues sur le jeu de test avec les corruptions utilisées durant l'apprentissage,
— enfin la couleur verte représente les performances obtenues sur le jeu de test avec les corruptions inconnues.

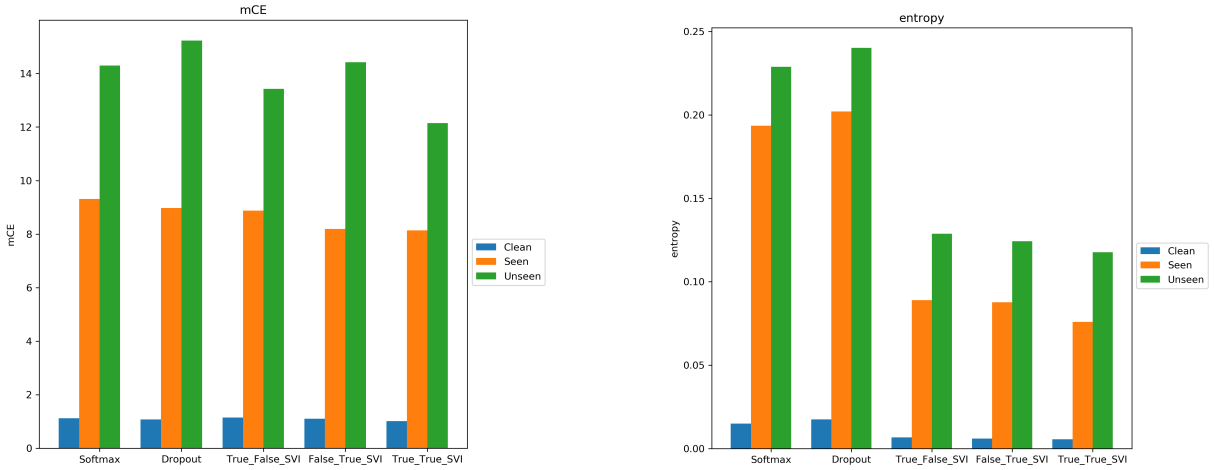
Globalement, pour chaque variable étudiée : erreur moyenne de corruption (mCE), entropie, ECE et Brier score, on observe une détérioration en passant du bleu, à l'orange puis au vert. En effet :

— l'erreur de corruption moyenne ("mean corrupted error" cf mCE) indique la proportion d'erreur mesurée sur le jeu concerné. Une erreur croissante indique un modèle dont la généralisation se dégrade à mesure que les corruptions se font de plus en plus étrangère/inconnues.

— L'entropie indique la quantité d'incertitude d'un modèle. Un modèle incertain est moins catégorique dans ses prédictions, et donc est plus en mesure d'indiquer au modélisateur les exemples sur lesquels il pourrait se tromper. Par exemple dans le cas de la classification binaire, une prédiction de 55 % est plus incertaine qu'une prédiction à 90 %, et la prédiction de 55 % exigerait plus de circonspection.

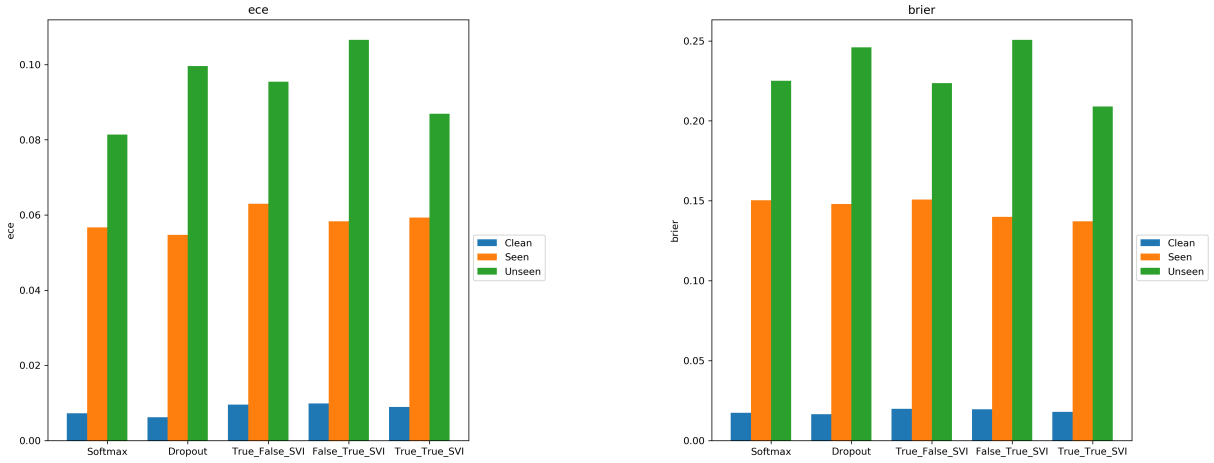
— la score Brier et l'ECE mesurent l'erreur de calibration : un indicateur de la qualité de l'incertitude. Par exemple dans quelle mesure peut-on dire qu'une prédiction de 55 % est certaine. Peut-être que le modèle sur-estime l'incertitude de cette prédiction. Si on considère tous les exemples au niveau avec un niveau de certitude de 55 %, si 90 % d'entre eux sont une bonne prédiction, alors cela indique une certitude sous-estimée. Ainsi, à mesure que ces métriques croissent, cela indique que la calibration se dégrade.

La seule différence notable se produit dans l'évaluation de l'entropie où la méthodologie SVI est bien plus sûre de ses prédictions que les modèles Softmax et Dropout, ce qui n'est pas forcément une bonne chose, car cela indique que ces modèles ne savent pas quand ils ne savent pas. À part ça, nous observons que quel que soit le modèle, l'erreur, l'entropie et l'erreur de calibration croissent à mesure que le jeu de données est corrompu avec des corruption de plus en plus inconnues. Ce qui illustre bien les limites de la théorie de l'apprentissage en contexte de glissement de données.



(a) (b)

FIGURE 7.4: Chaque colonne représente un des modèles : A priori invariant et généralisation hors domaine". Dans chaque cas, la couleur bleue représente les performances obtenues sur le jeu de test sans corruption, la couleur orange correspond aux performances obtenues sur le jeu de test avec les corruptions utilisées durant l'apprentissage, et enfin, la couleur verte représente les performances obtenues sur le jeu de test avec les corruptions inconnues. (a) Mean Corrupted Error (mCE) : il s'agit de l'erreur moyenne sur les corruptions. (b) Entropy : il s'agit de l'entropie globale calculée sur chaque jeu de test.



(a) (b)

FIGURE 7.5: Chaque colonne représente un des modèles. Dans chaque cas, la couleur bleue représente les performances obtenues sur le jeu de test sans corruption, la couleur orange correspond aux performances obtenues sur le jeu de test avec les corruptions utilisées durant l'apprentissage, et enfin, la couleur verte représente les performances obtenues sur le jeu de test avec les corruptions inconnues. (a) ECE : Expected Calibration Error est la mesure de l'erreur de calibration moyenne empirique, (b) Brier score ou score de Brier est une mesure de la calibration théorique.

7.5.3 Conclusion

Globalement l'augmentation de données seule ou transférée à travers un a priori invariant ne permet pas d'améliorer significativement les performances lors d'un glissement de données dû à des corruptions inconnues. On peut se demander si la quantité de corruptions contenues dans le méta jeu d'apprentissage suffiraient pour obtenir des informations permettant de bien généraliser au méta jeu de test. En effet, si les informations contenues dans le méta jeu d'apprentissage ne sont que marginalement utile pour le méta jeu de test, alors cela indique peut-être un manque d'information dans le méta jeu d'apprentissage. En somme, on a besoin de beaucoup plus de corruptions pour que les modèles puissent avoir suffisamment d'informations pour généraliser face à de nouvelles corruptions inconnues. C'est ce que nous investiguons dans la section suivante avec le transfert de style.

7.6 Transfert de style et diversité des données

Question de recherche Dans quelle mesure augmenter la diversité des données d'apprentissage améliore la généralisation hors distribution ?

7.6.1 Matériel et méthodes

Pour tester si augmenter la diversité du méta jeu d'apprentissage améliore la généralisation hors distribution, nous nous sommes inspiré de la méthodologie du transfert de

style de Geirhos [Geirhos et al., 2019]. Le transfert de style est une méthode permettant de transférer le style d'une image s à une image x pour créer une image stylisée x_s. Dans cette expérience, notre but est d'augmenter significativement la diversité des données en associant chaque style à un domaine. Avec cette perspective chaque style s peut perturber tout un jeu de données D et ainsi créer un glissement synthétique D_s.

— La base de données de style S que nous avons choisi est Wiki-Art [Nichol, 2016].

— Pour la base de données D à styliser nous avons choisi CIFAR-10 [Krizhevsky and Hinton, 2009].

— Pour transférer un style $s \in S$ à une image $x \in D$, nous avons utilisé l'algorithme Adaptative Instance Normalization (AdaIN) [Huang and Belongie, 2017].

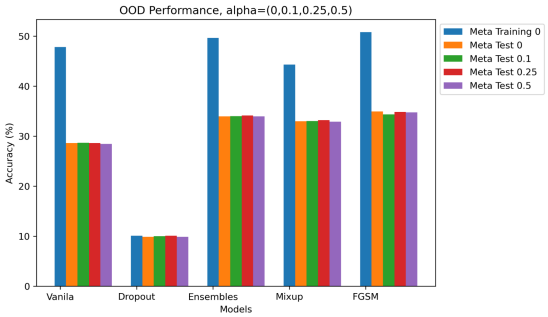
Pour répondre à la question de recherche, nous posons deux hypothèses :

1. L'intensité de la corruption sur le jeu de test augmente l'erreur de généralisation et donc la diversité des données.
2. Une méta distribution de domaines plus divers améliore la généralisation hors domaine.

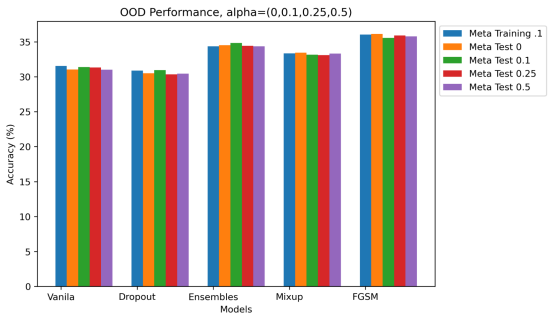
Comme nous cherchons à évaluer la capacité à généraliser hors distribution, nous avons donc besoin de pouvoir évaluer l'apprentissage sur des domaines inconnus. Nous avons partitionné le jeu de données de style S en deux : S_{tr}, S_{ts}. On peut considérer S_{tr}, S_{ts} comme des méta-distributions d'apprentissage et de test. Pour la base CIFAR-10, nous avons conservé la partition initiale avec 50,000 images pour le jeu d'apprentissage D_{tr} et 10,000 images pour le jeu de test D_{ts}. Nous avons utilisé l'architecture LeNet-5 pour apprendre sur les données stylisées, et nous avons comparé plusieurs algorithmes d'apprentissages : Empirical Risk Minimization (ERM), Dropout, Ensembles, Mixup, Fast Gradient Sign Method. Le but de cette comparaison est de savoir si un algorithme en particulier arriverait à généraliser à partir des styles. Ces algorithmes ont tous été entraînés avec la méthode d'optimisation Adam [Kingma and Ba, 2014]. Lors de l'apprentissage, les images stylisées ont toutes été transformées au format 32×32 pour pouvoir transférer le style durant l'apprentissage. Durant chaque itération, nous échantillonnons une portion de données : 128 exemples dans D_{tr}, et 128 styles dans S_{tr}, puis nous appliquons l'algorithme de transfert de style pour transformer chaque exemple avec un style différent.

Pour tester la première hypothèse, la partition du jeu de données de style était respectivement de 5933 et 2543 images stylisés pour les méta distributions d'apprentissage et de test S_{tr} et S_{ts}. Nous avons transféré la méta distribution d'apprentissage S_{tr} sur le jeu d'apprentissage D_{tr} avec respectivement un paramètre d'intensité de 0 (sans transfert) et de 0.1 (faible transfert). Puis nous avons évalué les modèles entraînés sur la méta distribution de test S_{ts} transférée au jeu de test D_{ts} avec plusieurs niveaux d'intensité de transfert : 0, 0.1, 0.25, 0.5. Les résultats des deux expériences sont affichés respectivement sur les figures 7.6 a et 7.6 b.

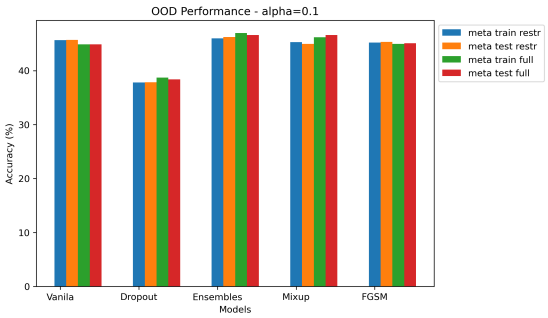
Pour tester la seconde hypothèse, nous avons créé deux méta distributions d'apprentissage $D_{tr}^{full}, D_{tr}^{restr}$ et deux méta distributions de test $D_{ts}^{full}, D_{ts}^{restr}$. D_{tr}^{full} et D_{ts}^{full} comprennenent respectivement 5933 et 2543 images stylisés, là où D_{tr}^{restr} et D_{ts}^{restr} comprennent respectivement 16 et 8 images stylisées. Les résultats sont affichés en figure 7.6 c.



(a)



(b)



(c)

FIGURE 7.6: (a) et (b) La seule différence est que sur la vignette (a) les modèles ont été entraîné sans transfert de style et sur la vignette (b) les modèles ont subi un entraînement avec transfert de style de paramètre 0.1. Puis les modèles ont été évalués sur le meta jeu d'apprentissage et sur le meta jeu de test avec plusieurs niveaux d'intensité, les légendes rapporte la couleur associé à chaque erreur. Enfin, la vignette (c) correspond aux performances obtenues sur $D_{tr}^{full}, D_{ts}^{full}$. D_{tr}^{restr} et D_{ts}^{restr} après apprentissage sur D_{tr}^{full} et D_{tr}^{restr}.

7.6.2 Résultats

Nous observons que l'intensité du transfert de style n'affecte pas la généralisation hors distribution. Cela est visible sur la figure 7.6 b, en effet si c'était le cas, nous verrions une chute de la performance pour tous les algorithmes à mesure que le paramètre augmente sur l'échelle 0.1,0.25,0.5. Or, nous n'observons pas cela. Cela signifie que l'intensité de la corruption n'augmente pas la diversité des données.

En observant la figure 7.6 a on voit une erreur de généralisation conséquente, c'est la différence entre la colonne bleue et la colonne orange. Cette chute de performance est due à un sur apprentissage. Par ailleurs, nous voyons que le transfert de style du méta jeu de test n'affecte pas les performances des algorithmes, malgré une intensité croissante. On remarque que l'apprentissage à échoué systématiquement pour Dropout, dans ce cas, on suppose que l'optimisation se bloque sur un point selle et ne parvient pas à en franchir.

Enfin, la figure 7.6 c nous montre qu'augmenter la quantité de données stylisées dans le méta jeu d'apprentissage n'améliore pas la diversité de la méta-distribution d'apprentissage. En effet, on supposait qu'entraîner un modèle sur D_{tr}^{full} serait plus performant qu'entraîner un modèle sur D_{tr}^{restr}. Hors, nous n'observons aucune différence entre ces deux méta distributions. De même, on avait supposé qu'apprendre avec une méta distribution plus importante aurait produit un méta sur-apprentissage un faible (i.e l'erreur en passant de D_{tr}^{full} à D_{ts}^{full}) qu'apprendre avec une méta distribution plus petite (erreur en passant de D_{tr}^{restr} et D_{ts}^{restr}).

Limitations Nous souhaitons attirer l'attention du lecteur sur le fait que nous n'avons conduit cette expérience qu'une seule fois, c'est-à-dire que nous n'avons pas contrôlé la variabilité. Cela induit des cas comme le Dropout en figure 7.6 a, mais aussi, vous observerez que les précisions oscillent entre 30 % et 45 %. Nous nous sommes limités à une expérience pour chaque figure, car les effets que nous cherchions à observer étaient les différences entre les niveaux de transfert (paramètre alpha et diversité). Ces effets étaient observables rien qu'avec une seule expérience, donc nous nous sommes arrêté là.

7.6.3 Conclusion et discussion

Nous pouvons conclure de cette expérience que la perspective de la diversité d'une personne est différente d'un modèle CNN. En effet, pour un humain, deux styles appliqués à la même image semblent beaucoup la changer, ce n'est pas le cas pour le CNN. C'est ce que nous pouvons interpréter des résultats 7.6 c. Par ailleurs pour un être humain, le niveau d'intensité du transfert semble beaucoup changer une image (voir [Huang and Belongie, 2017]), mais ce n'est pas ce que suggèrent les résultats pour le CNN.

Il semble que le transfert de style dégrade les performances, en effet, en observant les figures 7.6 a et b, on peut voir que l'erreur sur le jeu d'apprentissage diminue lorsque le transfert de style est appliqué. On peut supposer que le bruit ajouté rende l'extraction de caractéristique utile pour la classification plus difficile. Nous avons donc trouvé un exemple d'augmentation de données pouvant diminuer la généralisation.

Il semble que les performances OOD sont stables peu importe l'intensité ou le nombre d'images de style. Nous pensons que cela est dû au fait que le transfert de style se fait à l'aide d'un CNN, et que nous utilisons un CNN pour apprendre. En effet, le transfert

de style se fait au travers d'un traitement des données qui est identique à l'extraction des caractéristiques utiles pour l'apprentissage. Donc la diversité est inhérente au modèle ou à la représentation utilisée.

Nous pensions que le transfert de style permettait d'encoder une diversité arbitraire dans les données, mais nous nous sommes trompés. Nous ne pouvons pas répondre à la question de recherche, cependant nous pouvons essayer une autre approche ayant porté ses fruits dans la littérature avec [Hendrycks et al., 2019]. C'est ce que nous faisons dans la suite.

7.7 Augmentation de données en contexte de glissement synthétiques et naturels

Question de recherche Comment différentes catégories d'augmentation se comparent sur des glissements synthétiques et naturels ?

7.7.1 Matériel et méthodes

Nous avons choisi de comparer 3 catégories d'algorithmes d'augmentation de données à la simple minimisation du risque empirique (Empirical Risk Minimization - ERM) [Vapnik, 1992].

1. Implicit Semantic Data Augmentation (ISDA) [Wang et al., 2019c] est un algorithme tirant profit de la réduction de la dimension qu'opère un réseau de neurones sur les données. En effet lorsqu'une donnée de haute dimension, par exemple une image est passée à un réseau de neurones, cette image sera transformée à chaque couche en une représentation de plus en plus synthétique. Cette représentation synthétique sera utilisée dans la dernière couche pour faire la classification. Donc ISDA va modéliser chaque classe avec la dernière couche suivant une loi Gaussienne multivariée avec une moyenne et une matrice de covariance. Cela permet à l'algorithme de modéliser la variabilité intra-classe. Ensuite, ils peuvent augmenter les données au niveau de la dernière couche en y ajoutant un bruit échantillonné avec la Gaussienne décrit précédemment. Cela permettant d'augmenter les données avec des représentations associées aux autres objets de même classe. Enfin, au lieu de faire l'échantillonnage potentiellement coûteux durant l'apprentissage, ils ont proposé une fonction coût approximant asymptotiquement cette opération. Cet algorithme typiquement induit une forme d'invariance intra-classe directement dans le modèle.
2. Fast Gradient Sign Method (FGSM) est la méthode d'apprentissage adverse traditionnelle, nous avons opté pour une méthodologie améliorée permettant un apprentissage plus rapide [Wong et al., 2019]. Nous avons choisi cette approche pour répliquer les résultats dans [Hendrycks et al., 2021a], et pour avoir un représentant des méthodologies d'augmentation de données adverses.
3. Finalement, Augmix [Hendrycks et al., 2019] est une stratégie d'augmentation de données ayant montré les meilleurs résultats sur les benchmarks de robustesse face à des perturbations synthétiques. En effet, Augmix obtenait les meilleurs résultats sur

ImageNet-C, CIFAR-10-C, CIFAR-10-P, comparé à d'autres stratégies d'augmentation de la littérature : CutMix [Yun et al., 2019], Cutout [DeVries and Taylor, 2017], Mixup [Zhang et al., 2018], AutoAugment [Cubuk et al., 2019], et face à de l'apprentissage adverse. Par ailleurs, ces résultats étaient consistants avec plusieurs architectures.

L'apprentissage sur ImageNet étant très coûteux, nous avons téléchargé les réseaux pré-entraînées suivant chaque méthode d'apprentissage pour comparer leurs performances.

Nous avons utilisé un sous-ensemble d'ImageNet (38 classes) pour évaluer de nombreuses perturbations : ImageNet-A-C-R-Sketch (17 domaines au total). 38 classes, car les perturbations d'ImageNet n'incluent pas tous toutes les 1000 classes originales, et seules 38 classes sont partagées par chaque jeu de données. Par ailleurs, ces jeux de données ont été décrits aux chapitres 5.

7.7.2 Résultats

Nous présentons les résultats bruts avec le tableau 7.2, et avec la figure 7.7. Le tableau 7.2 permet d'évaluer la chute de performance sur des glissements de données spécifiques. La figure 7.7 dessine les distributions empiriques estimées pour les résultats de chaque méthode. Ces distributions nous donnent un point de vue global sur le succès de chaque méthode.

On observe qu'en moyenne FGSM est la meilleure méthode globalement, car le mode de la distribution (Le pic figure 7.7.) est le plus à droite. Cela signifie que FGSM obtient le plus de bons résultats, suivi de Augmix, ERM et enfin ISDA. Cependant, gardons en tête que cette évaluation comprend 3 glissements de données naturels : sketch, adversarial, rendition, et 14 glissements de données synthétiques. Donc, il peut être intéressant de regarder le détail. On observe sur le tableau 7.2 que le glissement le plus dommageable aux différents modèles est ImageNet-A, en effet les performances sont de l'ordre de 15 %. On peut observer aussi cette petite concentration sur la figure 7.7, sur l'axe des abscisses entre 10 % et 20%. On peut voir qu'ISDA se comporte le mieux sur ce type de glissement de données suivi d'Augmix, ERM et FGSM. Pour les deux autres glissements naturels : sketch et rendition, Augmix a une performance supérieure suivie de ERM, ISDA et FGSM.

Ensuite, nous avons synthétisé les résultats en fonction du type de glissement : naturel ou synthétique. Et nous avons aussi comparé deux critères de performance hors distribution :

— la moyenne des précisions obtenues sur chaque domaine comme proposé par [Hendrycks and Dietterich, 2019] au tableau 7.3,

— la précision minimale obtenue parmi tous les domaines comme proposé par [Arjovsky, 2020] au tableau 7.4.

On observe que si on choisit le critère de la moyenne pour évaluer la performance, alors Augmix a la meilleure performance sur les glissements naturels et FGSM a la meilleure performance sur les glissements synthétiques. Par contre si on choisi le critère de la pire performance, c'est ISDA qui est la meilleure méthode sur les glissements naturels et Augmix qui est la meilleure méthode sur les glissements synthétiques.

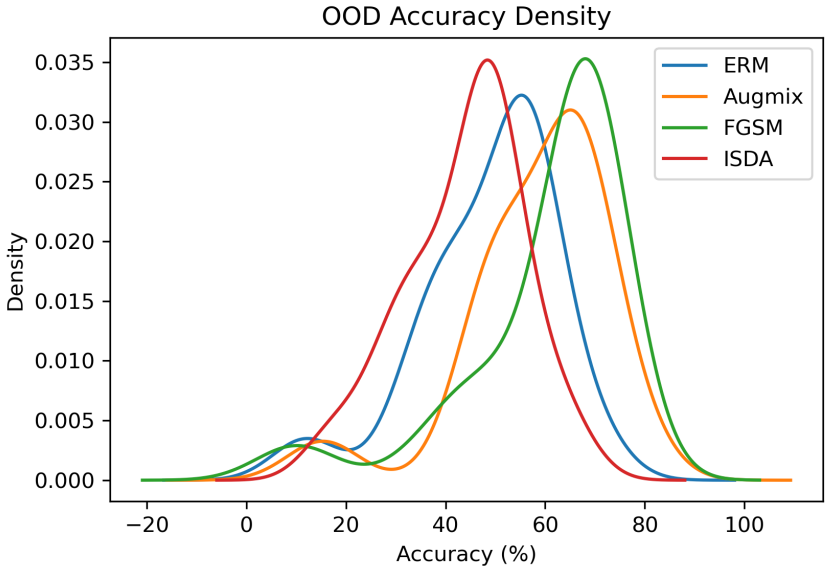


FIGURE 7.7: Vue globale des performances obtenues sur les 17 glissements de données d'ImageNet-A-C-R-Sketch.

TABLE 7.2: Détail des performances obtenues sur les 17 glissements de données d'ImageNet-A-C-R-Sketch.

	Précision algorithmes (%)			
Perturbation	ERM	Augmix	FGSM	ISDA
sketch	57,23	61,43	48,24	54,35
adversarial	12,06	15,35	10,1	17,56
rendition	45,35	50,74	38,45	44,29
brightness	69,4	77,98	71,37	64,56
contrast	34,76	48,35	47,32	28,84
focus blur	52,83	65,77	65,27	46,72
elastic transform	62,51	72,35	69,54	58,3
fog	57,54	66,65	65,44	50,43
frost	54,83	65,71	68,77	46,87
gaussian noise	34,08	46,3	68,88	28,34
glass blur	40,46	50,04	60,45	34,25
impulse noise	40,49	52,18	64,66	36,33
jpeg compression	55,89	66,14	72,08	48,5
pixelate	56,46	68,58	71,12	48,58
shot noise	44,49	56,31	70,35	38,19
snow	55,01	64,48	67,75	49,35
zoom blur	56,19	67,2	68,83	50,09

7.7.3 Conclusion et discussion

Nous voyons clairement que la nature du glissement, et la méthodologie d'évaluation influe sur la performance hors distribution. D'ailleurs, c'est l'un des points pointés par Gul-

TABLE 7.3: Moyenne empirique des performances obtenues sur les glissements naturels et synthétiques.

	Précision algorithmes (%)			
Perturbation	ERM	Augmix	FGSM	ISDA
glissements naturels	34.65	**38.4**	29.17	35.96
glissements synthétiques	50.69	61.25	**64.69**	44.91

TABLE 7.4: Performances minimales obtenues sur les glissements naturels et synthétiques.

	Précision algorithmes (%)			
Perturbation	ERM	Augmix	FGSM	ISDA
glissements naturels	12.06	15.35	10.1	**17.56**
glissements synthétiques	34.08	**46.3**	38.45	28.34

rajani [Gulrajani and Lopez-Paz, 2020b], en particulier que les méthodes de généralisation hors domaine doivent spécifier un critère de sélection des modèles. Ce critère reflète les connaissances a priori du déploiement du futur modèle. En effet, par exemple dans une application médicale, le critère de la pire performance est la plus intéressante. Car elle indique ce qui pourrait arriver à un patient dans le pire des cas si le modèle était déployé. Mais, dans des applications moins critiques, la moyenne est suffisante. Aussi, en segmentant clairement la performance en fonction de la nature du glissement, nous pouvons avoir une meilleure analyse du comportement des modèles en amont de leur déploiement.

Dans la suite, nous considérerons que la moyenne des performances obtenues sur les différents domaines est un bon critère de sélection des modèles. Nous voulons comprendre comment Augmix peut avoir une bonne performance sur les glissements naturels. Est-ce dû à la capacité de cette méthode à générer un grand nombre de combinaisons d'augmentations génériques ? Donc est-ce qu'AugMix est capable de générer des données plus diversifiées ? Et est-ce que cette performance sur des glissements naturels pourrait se maintenir sur un grand nombre de domaines ? Le problème avec les glissements naturels, c'est qu'ils sont difficiles à obtenir. En effet, collecter de nouvelles données est cher même si ce sont les types de données qui représentent mieux le contexte de déploiement. Heureusement, la communauté en généralisation hors domaine a proposé un benchmark composé de plusieurs jeux de données juste pour ça [Koh et al., 2021] que nous investiguons dans la suite.

7.8 Augmentation de données en contexte de glissement naturels

Question de recherche Est-ce qu'une grande diversité de corruptions synthétiques augmente la robustesse face à des glissements naturels ?

7.8.1 Matériel et méthodes

Pour cela, répondre à la question de recherche, nous avons seulement besoin d'une très bonne méthodologie de corruption synthétique et d'un benchmark de glissement sur des

données naturelles.

— Nous nous concentrons sur Augmix [Hendrycks et al., 2019] décrit précédemment.

— Comme benchmark, nous avons choisi deux jeux de données d'images de la suite WILDS : Camelyon17 et FMoW [Koh et al., 2021]. Camelyon17 est une base de données de mammographies pour classifier le cancer du sein, donc une classification binaire. Les données ont été échantillonnées de 5 hôpitaux différents. Donc, ici la base de données est répartie suivant 5 domaines. FMoW est composé d'images satellites représentant des infrastructures : ponts, centres commerciaux, résidences, institutions éducatives, etc. Le domaine ici est le couple lieu et l'année, en effet une même infrastructure peut changer dans le temps, et différer d'un continent à un autre.

L'approche est simple, WILDS a mis sa base de code en open source, et permet de fixer la plupart des hyper-paramètres d'apprentissage de sorte que le modélisateur se concentre uniquement sur un aspect à étudier. Ici, nous avons implémenté Augmix dans le code de WILDS, et nous avons comparé les résultats obtenus hors distribution les autres méthodologies évaluées et conservées sur le leaderboard du projet WILDS.

Pour Camelyon17 et FMoW, les modèles considérés sont respectivement un DenseNet-121 sans pré-entraînement et un DenseNet-121 avec pré-entraînement. L'apprentissage a été produit, puis les modèles résultants ont été évalués respectivement 10 fois et 3 fois avec des initialisations différentes. Cette approche permet d'obtenir un score moyen hors distribution et de quantifier la variabilité dans les performances de généralisation.

Les métriques considérées pour mesurer la performance hors distribution sont la précision moyenne sur les domaines inconnus pour Camelyon17, et pour FMoW.

7.8.2 Résultats

Les résultats sur le jeu Camelyon17 et le jeu FMoW sont disposés respectivement aux tables 7.5 et 7.6. Ils sont classés par ordre de performance, avec les algorithmes les plus performants en haut du tableau. On observe des résultats contradictoires. En effet, Augmix permet d'obtenir un bon significatif sur Camelyon17, et d'obtenir les meilleurs résultats parmi toutes les méthodes déjà répertoriées dans la littérature. En effet ERM seul donne une précision sur le jeu test de 70.3 % là où ERM et AugMix obtiennent 91.9 % pour le modèle DenseNet-121, soit un bon de 21.6 %. En ce qui concerne les résultats sur FMoW, Augmix s'est montré décevant, en effet, il a obtenu un score inférieur à la minimisation du risque empirique, c'est à dire que l'apprentissage a diminué la performance hors distribution. Pour le modèle DenseNet-121, Augmix obtient le score de généralisation hors distribution le plus bas. En effet ERM seul donne une précision sur le jeu test de 55.5 % là où ERM et AugMix obtiennent 47.7 % pour le modèle DenseNet-121, soit une perte de 7.8 %.

7.8.3 Conclusion et discussion

Les corruptions synthétiques n'entraînent pas mécaniquement une généralisation sur des glissements naturels. Sur certains glissements naturels comme Camelyon17, ces corruptions peuvent améliorer la généralisation hors distribution en accordance avec Hendrycks [Hendrycks et al., 2021a], mais on ne peut pas en faire une généralité comme l'a mentionné

TABLE 7.5: Résultats des modèles sur Camelyon17.

Camelyon17			
Algorithm	Model	Val Acc	Test Acc
ERM w/ AugMix	DenseNet121	91.1 (1.0)	91.9 (2.2)
ERM w/ data aug (Miller at al. 2021)	SE-ResNeXt101-32x4d	88.0 (4.2)	91.6 (1.9)
Fish (WILDS)	DenseNet121	83.9 (1.2)	74.7 (7.1)
ERM (WILDS)	DenseNet121	84.9 (3.1)	70.3 (6.4)
Group DRO (WILDS)	DenseNet121	85.5 (2.2)	68.4 (7.3)
IRM (WILDS)	DenseNet121	86.2 (1.4)	64.2 (8.1)
CORAL (WILDS)	DenseNet121	86.2 (1.4)	59.5 (7.7)

TABLE 7.6: Résultats des modèles sur FMoW.

FMoW			
Algorithm	Model	Val Avg Acc	Test Avg Acc
ERM (Miller et al 2021)	SE-ResNeXt101-32x4d	62.1 (0.24)	55.5 (0.14)
Fish (Shi et al. 2021)	DenseNet121	57.8 (0.15)	51.8 (0.32)
ERM (WILDS)	DenseNet121	59.5 (0.37)	53.0 (0.55)
CORAL (WILDS)	DenseNet121	56.9 (0.25)	50.5 (0.36)
Group DRO (WILDS)	DenseNet121	58.8 (0.19)	52.1 (0.5)
IRM (WILDS)	DenseNet121	57.4 (0.37)	50.8 (0.13)
ERM w AugMix	DenseNet121	54.4 (1.7)	47.7 (2.2)
ERM (Miller et al 2021)	CLIP (ResNet50)	41.6 (0)	36.8 (0)
ERM (Miller et al 2021)	CLIP (ViT-B/32)	41.6 (0)	36.3 (0)

[Taori et al., 2020], en effet nos résultats sur FMoW nous le montrent. La littérature a investigué que les corruptions synthétiques étaient particulièrement efficace même face à des glissements naturels, spécifiquement dans le cas des données médicales [Zhang et al., 2019]. En effet, dans le cadre des données médicales, le changement de domaine est principalement causé par un changement dans les capteurs, mais le protocole de collecte de données est souvent standardisé. Alors que dans d'autres cas comme FMoW, d'autres facteurs de variations s'appliquent.

Cependant, la chute de performance dans le cas de FMoW nous pousse à penser qu'Augmix a éloigné la distribution des données de la distribution naturelle par rapport à la distribution d'évaluation. Cela nous pousse à penser que la quantification de la distance entre jeux de données est une tâche primordiale pour apprendre à généraliser hors distribution.

7.9 Conclusion

7.9.1 Augmentation de données et généralisation

Notre première expérience nous a appris que le choix d'une augmentation de données pertinente peut faire la différence entre un modèle utile et un modèle inutile. Notre expérience étant utilisée dans le contexte d'un modèle génératif, le résultat ici était plus qualita-

tif que quantitatif. Cependant, la question de déterminer la pertinence d'une augmentation de données persistait. Nous avons appris au cours de notre deuxième expérience qu'il est possible d'induire un modèle invariant à une perturbation donnée avec peu d'exemples venant du jeu d'apprentissage. En d'autres termes, lorsqu'un modèle apprend à être invariant sur un petit échantillon du jeu d'apprentissage, alors il est capable d'utiliser cette invariance sur d'autres exemples du jeu d'apprentissage soumis à la perturbation. Cette réalisation a été faite avec un a priori invariant, dans le cadre d'un apprentissage Bayésien. Cela laisse penser qu'il est possible d'extraire le potentiel de généralisation d'une perturbation avec un apprentissage en ligne appliqué seulement avec une faible probabilité durant l'apprentissage. La conséquence de cela, est de gagner du temps d'apprentissage. En effet, au lieu d'augmenter les exemples d'un lot avec une probabilité de 1 (ce qui équivaudrait à perturber tous les exemples d'un lot durant l'apprentissage), autant utiliser une probabilité de 0.1 et ne perturber qu'une petite proportion du lot. Enfin, notre troisième expérience avec l'apprentissage actif nous a montré que l'augmentation de données permettait un apprentissage plus efficace. Dans le sens d'augmenter la généralisation dans un contexte ou seulement peu de données sont disponibles pour l'apprentissage. Notons, que nous avions aussi observé ce phénomène avec la première expérience. Autrement dit, l'augmentation de données améliore la généralisation des modèles, permet de mieux tirer profit des petits jeux de données. Ces résultats étaient déjà connus. Par ailleurs, nous montrons que les bénéfices de l'augmentation de données peuvent être obtenus rien qu'avec une application partielle lors de l'apprentissage. Nous voulons maintenant savoir dans quelle mesure cette technique peut servir à généraliser hors distribution.

7.9.2 Augmentation de données et généralisation hors distribution

Nos premières expériences de l'augmentation de données étaient positives et laissaient entrevoir un potentiel intéressant pour améliorer la performance hors distribution. Notre perspective était sans doute naïve. En effet, dans notre quatrième expérience, nous avons tenté d'augmenter les données avec certaines corruptions pour généraliser face à des corruptions inconnues. Cependant, notre perspective était irréaliste, car notre critère du succès implicite était de rapprocher l'erreur de généralisation hors distribution de l'erreur de généralisation. Une meilleure perspective aurait été de comparer un apprentissage avec augmentation et sans augmentation sur la généralisation hors distribution. Ce que nous avons fait en sixièmes et septièmes expériences finalement. Mais l'analyse des résultats de notre première expérience nous a plutôt poussé à penser que nous devions chercher à obtenir des augmentations plus diversifiées pour améliorer la généralisation hors distribution. C'est ce que nous avons tenté avec notre cinquième expérience. L'idée de notre cinquième expérience était d'utiliser le transfert de style comme moyen de générer une quantité arbitraire de corruptions. Le but était d'obtenir une grande diversité dans les données d'apprentissage. Cette expérience cependant, nous a appris que les modèles avaient une perspective sur les données. En effet, cela paraît évident a posteriori, et ce phénomène est bien connu dans la littérature des exemples adverses. Une petite perturbation change la prédiction d'un modèle alors que nous ne voyons aucun changement. Et bien avec le transfert de style, il semble que ce soit l'inverse qui se soit produit. Dans ce cas, modifier une image avec le transfert de style modifie visuellement une image pour un humain et nous pensions que la perspective de ce changement serait la même pour un réseau de neurones convolutif. Mais il n'en était

rien. Il semble que pour le modèle l'injection de styles différents revienne au même. Ainsi, toute représentation, que cela soit un réseau de neurones convolutif, ou nos propre réseaux de neurones biologiques sont avant tout une perspective sur les données. Notre sixième et septième expérience ont suivi la même méthodologie. À la différence près que nous avons considéré principalement des glissements de données synthétiques dans la sixième expérience, et des glissements de données naturels pour notre septième expérience. Il s'agissait de comparer l'effet d'un apprentissage avec et sans augmentation sur la généralisation hors distribution. Comme le transfert de style ne nous donnait pas satisfaction pour générer une grande diversité de données, nous nous sommes tourné vers AugMix une méthodologie de combinaison de chaînes d'augmentations. Ici, le résultat le plus intéressant, était le fait que dans un certain cas, l'augmentation de données améliorait significativement la généralisation hors distribution (Camelyon-17). Alors que dans un autre cas, l'augmentation de données diminuait la généralisation hors distribution (FMoW). Avec le recul, ce résultat n'est pas surprenant. En effet, si le domaine cible contient des données qui varient suivant un chemin, et que ce chemin est encodé dans l'augmentation de données, alors la performance s'améliorera. Par contre si l'augmentation de données ne contient aucun facteur de variation pouvant être utile au domaine cible, alors la performance se réduira. Pour résumer ce que nous apprenons ici, il faut se rappeler qu'une augmentation de données rend un modèle invariant au facteur de variation qu'elle expose. Ainsi, une augmentation de données est pertinente pour les données de déploiement seulement si le facteur de variation qu'elle expose existe pour dans la distribution de déploiement. Donc, une augmentation de données peut rapprocher ou éloigner le domaine source du domaine cible. Cette intuition avait déjà été capturée par la notion de $\mathcal{H}$-distance que nous utilisons dans notre dernier chapitre.

Chapitre Huit

Estimation de l'erreur de déploiement avec l'augmentation de données

Sommaire

8.1 Introduction : une estimation de l'incertitude bancale

Les régresseurs proposés dans la littérature mentionnent rarement les niveaux d'incertitudes. En effet, Deng et al. [Deng et al., 2021] n'ont pas mentionné de niveaux d'incertitude autour de l'erreur de déploiement estimé. Garg et al. [Garg et al., 2022] n'ont pas mentionné d'incertitude dans l'article principal, mais ils ont signalé l'écart-type dans l'annexe. Notons qu'Elsahar a publié l'écart-type [Elsahar and Gallé, 2019] et d'autres auteurs [Guillory et al., 2021, Chen et al., 2021] ont publié des intervalles de confiance. Toutefois, nous serions prudents avec ces valeurs. En effet, ces travaux admettent tacitement les hypothèses de la régression linéaire : (1) la variance des erreurs pour chaque nouveau domaine est la même (homoscédasticité), en effet, sinon la variance pour chaque domaine devrait être calculée et publiée. (2) Les erreurs sont normalement distribuées de sorte que l'écart-type puisse être naturellement étendu aux intervalles de confiance. (3) Il existe une relation linéaire entre les distances distributionnelles et les erreurs de déploiement. (4) Les erreurs sont indépendantes. Cela signifie qu'on suppose que les domaines perturbés l'ont été par des perturbations indépendantes. Cependant, si ces hypothèses ne sont pas vérifiées alors, cette quantification de l'incertitude devient trompeuse. Dans ce chapitre, nous proposons

de relaxer les hypothèses de linéarité et d'homoscédasticité. C'est-à-dire que nous supposons que la relation entre les distances distributionnelles et l'erreur de déploiement peut être non-linéaire. Par ailleurs, nous supposons aussi que la variance autour de l'erreur de déploiement varie d'un domaine à l'autre. Enfin, nous appliquons notre méthodologie dans un contexte de déploiement réel avec la base de données de mammographies Camelyon17 pour classifier le cancer du sein.

8.2 Estimation et mesure de l'erreur de déploiement

Dans les sous-sections suivantes, nous décrivons les méthodologies utilisées dans tout le chapitre, nous détaillerons les méthodologies spécifiques aux expériences dans les sections suivantes. Au chapitre 5 nous avions vu que toutes les méthodes actuelles d'estimation de l'erreur de déploiement peuvent s'interpréter dans un cadre modulaire en deux étapes :

1. le calcul de distances distributionnelles entre les données sources et les stress-test. Les stress-tests étant des glissements de données étiquetés.
2. L'entraînement d'un régresseur pour estimer l'erreur de déploiement à partir des distances distributionnelles.

Dans les deux prochains paragraphes, nous décrivons la manière dont on entraînera ce régresseur et comment mesurer sa performance.

Estimation de l'erreur de déploiement

Pour estimer l'erreur de déploiement à partir des distances distributionnelles, nous utilisons un réseau de neurones à deux couches cachés de dimensions 50, avec des fonctions d'activations non-linéaires SELU [Klambauer et al., 2017] et ayant pour sortie la valeur moyenne et la variance i.e. $h(x) = (\mu(x), \sigma(x))$ en nous inspirant de [Nix and Weigend, 1994] . Ainsi, ce modèle admet une relation non-linéaires entre les entrées et les sorties, et propose une estimation de la variance pour chaque exemple. Ce faisant, nous relaxons effectivement les hypothèses de linéarité et d'homoscédasticité. Nous dérivons la fonction coût du maximum de vraisemblance :

$$l(x, y, h) = \frac{(y - \mu(x))^2}{2\sigma^2(x)} + \log(\sigma(x))$$

Comme nous conservons l'hypothèse de normalité des données, on peut obtenir un intervalle de prédiction de niveau 1-α pour un point x en utilisant $[\mu(x) - z_{1-\alpha} \cdot \sigma(x), \mu(x) + z_{1-\alpha} \cdot \sigma(x)]$, avec $z_{1-\alpha}$ le z-score de niveau $1 - \alpha$. Enfin, nous entraînons le modèle avec Adam [Kingma and Ba, 2014], un taux d'apprentissage de 0.001 et durant 250 epochs.

Métrique pour mesurer l'erreur de déploiement

La plupart des estimateurs de l'erreur de déploiement [Elsahar and Gallé, 2019, Guillory et al., 2021, Garg et al., 2022, Chen et al., 2021] utilisent l'erreur moyenne absolue ("Mean Absolute Error" cf MAE) en pourcentage de précision pour reporter la précision de l'erreur de déploiement. À l'exception de Deng et al. [Deng and Zheng, 2021,

Deng et al., 2021] qui utilise l'erreur quadratique moyenne ("Root Mean Square Error" RMSE). Dans la suite, nous utilisons le MAE.

Pour quantifier l'incertitude de l'erreur de déploiement, nous utilisons le PICP et le MPIW. Pour rappel, le PICP est le pourcentage de données capturées par l'intervalle de prédiction. Et le MPIW est la taille moyenne de l'intervalle de prédiction sur les données considérées. Pour plus d'information, nous invitons le lecteur à se reporter au chapitre 5. Notons que toutes ces valeurs peuvent s'exprimer en pourcentages :

— le MAE, car les erreurs sont mesurées en pourcentages,
— le MPIW, car la taille de l'intervalle de prédiction concerne des erreurs mesurées en pourcentages,
— le PICP, car la proportion de données capturées est un pourcentage.

8.3 Benchmarking des distances distributionnelles

Question de recherche Quel est l'algorithme de distance distributionnelle permettant de prédire au mieux l'erreur de déploiement ?

8.3.1 Matériel et méthode

Dans ce qui suit, nous décrivons :

1. les données sources et les données cibles.
2. les modèles utilisés pour apprendre sur chaque jeu de données source.
3. les distances distributionnelles.

Jeux de données

Comme données sources, nous avons utilisé MNIST, CIFAR-10 et TinyImagenet. En tant que données cibles pour MNIST, nous choisissons MNIST-C et MNIST-C Leftover, qui représentent un total de 32 glissements de données synthétiques du jeu de données MNIST [Mu and Gilmer, 2019]. Pour CIFAR-10, nous avons utilisé CIFAR-10-C [Hendrycks and Dietterich, 2019], le jeu contient 15 glissements synthétiques chacun ayant 5 niveaux de corruption pour un total de 75 glissements de données. Dans ce travail, nous avons seulement sélectionné un niveau de corruption de 3 pour chaque glissement synthétique. Résultant en un ensemble de 15 glissements de jeux de données. Pour TinyImagenet, nous avons utilisé Imagenet-C [Hendrycks and Dietterich, 2019] et la même procédure qu'avec CIFAR-10-C. C'est-à-dire que nous avons sélectionné un ensemble de 15 glissements synthétiques. Nous avons aussi ajouté Imagenet-A [Hendrycks et al., 2021b] un glissement naturel, puis nous avons démêlé 10 domaines d'Imagenet-R [Hendrycks et al., 2021a] (art, cartoon, embroiderm graffiti, graphisme, divers, peinture, sculpture, croquis, jouet) correspondant à autant de glissements naturels de TinyImageNet. Il y avait un compromis entre ajouter des glissements synthétiques et conserver un grand nombre de classes. Nous avons maximisé le nombre de domaines à 10, ce qui nous a forcé à ne conserver que 23 classes communes à tous les glissements synthétiques et naturels. Ainsi pour TinyImagenet, l'hypothèse de décalage covariée est rompue, car la distribution des classes du sous-domaine ImageNet-R est différente de celle dans ImageNet-C et Imagenet-A.

Modèles

Pour MNIST, CIFAR-10 et TinyImagenet, nous avons respectivement utilisé une architecture LeNet-5 [LeCun and Bengio, 1995], une architecture ResNet-18 et une architecture ResNet-50 [He et al., 2015]. Chaque modèle a été entraîné sur les données sources avec Adam [Kingma and Ba, 2014] et un taux d'apprentissage de 0,01.

Distances distributionnelles

— La $\mathcal{H}$-distance [Ben-David et al., 2006] calcule la distance ou la divergence entre des données structurées de grande dimension. Il ne peut être qu'approximatif, car il ne peut pas être calculé à partir d'échantillons finis. Pour calculer un proxy de la distance $\mathcal{H}$, nous entraînons un modèle à partir d'une classe d'architectures $\mathcal{H}$ sur la tâche de classification binaire consistant à distinguer les données de deux domaines [Ganin et al., 2016]. Dans cette section, nous avons utilisé une architecture pré-entraînée sur les données étiquetées source et avons uniquement entraîné un classifieur binaire par-dessus. Par conséquent, ici $d(S,T)$ est le proxy $\mathcal{H}$-distance entre les deux ensembles de données. En pratique, nous avons utilisé la même architecture que celle utilisée pour l'entraînement sur l'ensemble de données source et n'avons formé le classifieur qu'en haut sur la tâche de classification binaire.

— Average Threshold Confidence (ATC) [Garg et al., 2022] exploite les niveaux de confiance d'un modèle entraîné sur l'ensemble de données source. Fondamentalement, pour chaque exemple, ATC comprime les niveaux de confiance dans un score, puis utilise un régresseur pour estimer la précision à partir des scores calculés sur les données sources. Dans sa forme actuelle, ATC n'est pas une distance distributionnelle comme nous l'avons décrit au chapitre 5, mais il est facile de le transformer en distance avec la fonction $d(S,T) = |ATC(h_S,S) - ATC(h_S,T)|$. Essentiellement, la différence entre l'estimation de la précision sur l'ensemble de données source et l'estimation de la précision sur le décalage de l'ensemble de données. En pratique, le régresseur ATC que nous avons utilisé était un réseau de neurones à deux couches cachées qui estimait la précision à partir d'un lot de 128 scores. Le réseau neuronal avait 128 dimensions cachées et prédisait pour chaque lot une précision. Comme score, nous utilisons l'entropie prédite.

— Optimal Transport Dataset Distance (OTDD) [Alvarez-Melis and Fusi, 2020] a été conçue à l'origine pour calculer une distance entre des ensembles de données étiquetés. Dans l'article, les auteurs ont estimé la distribution empirique $\alpha_y = \mathbb{P}(X|Y=y)$ avec une distribution gaussienne, puis ont utilisé la forme analytique de la distance de Wasserstein pour calculer la distance entre les ensembles de données. . Ici, $d(S,T)$ est l'OTDD entre deux jeux de données. Nous avons utilisé le code fourni sur le github de l'article [Alvarez-Melis and Fusi, 2020].

— Generalized Disagreement Error [Jiang et al., 2022] repose sur la méthodologie des ensembles de modèles. La méthode consiste à entraîner sur la même source de données S, deux réseaux de neurones h_S^1, h_S^2 avec pour seule différence une initialisation différente. L'idée des ensembles est que chaque modèle finira dans un maximum local différent, ainsi, ces modèles auront des désaccords sur le classement de certains exemples. Le degré de désaccord est ici utilisé pour mesurer la distance entre jeux de

données. En effet, les modèles auront peu de désaccord sur le jeu issu de la distribution d'apprentissage. Cependant, ces modèles auront des désaccords de plus en plus importants à mesure que les jeux de données vont diverger du jeu d'apprentissage. Ici $d(S,T) = |GDE(h_S^1, h_S^2; S) - GDE(h_{S^1}, h_S^2; T)|$.

Étant donné que chaque distance distributionnelle entre jeux de données produits des sorties à échelles différentes, nous avons normalisé les résultats entre 0 et 1 faciliter la comparaison. Cela ayant l'avantage de mesurer la distance en pourcentage.

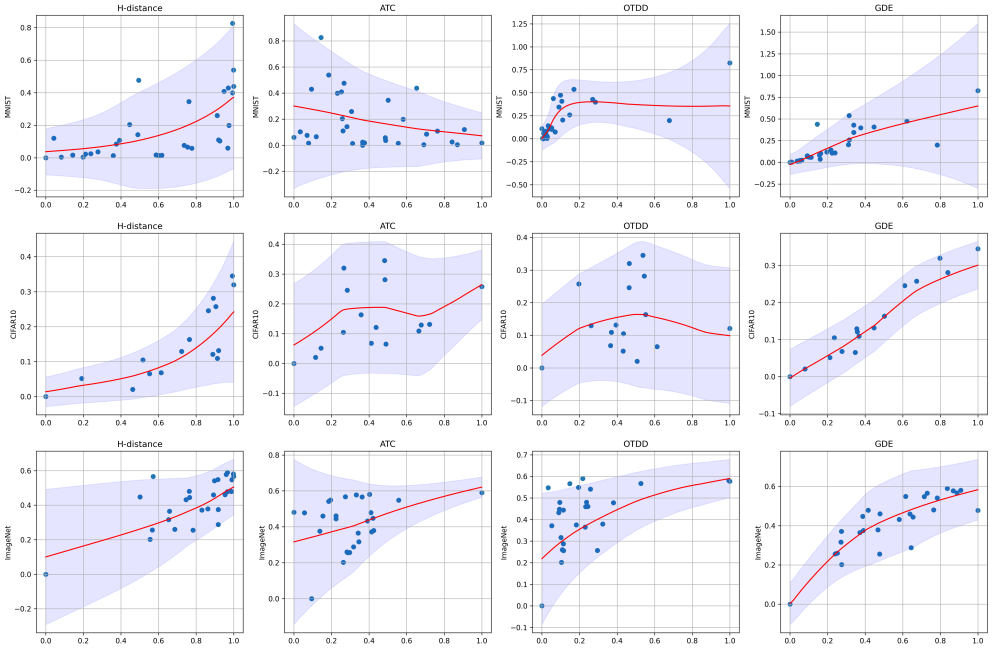


FIGURE 8.1: Les colonnes indiquent respectivement les algorithmes $\mathcal{H}$-distance, ATC, OTDD et GDE. Les lignes indiquent respectivement les ensembles de données MNIST, CIFAR-10 et TinyImagenet. L'axe des y présente l'erreur de déploiement $|\mathcal{R}^T - \mathcal{R}^S|$, et l'axe des x présente la distance entre jeux de données normalisée. Chaque point représente les performances calculées par l'algorithme sur un glissement de données spécifique. La ligne rouge représente la prédiction effectuée par le réseau de neurones, enfin la zone ombrée représente des intervalles de prédiction de 99%.

8.3.2 Résultats

Les résultats sont disponibles à la figure 8.1 et nous reportons trois métriques de performance : MAE, PICP et MPIW au tableau 8.1. On peut voir que l'algorithme GDE commet l'erreur moyenne (MAE) la plus faible sur chacun des jeux de données concernés : MNIST, CIFAR-10 et TinyImageNet. Concernant la fiabilité de l'incertitude GDE a le PICP le plus élevé sur CIFAR-10 à égalité avec OTD et ATC. GDE est aussi à égalité avec ATC

TABLE 8.1: Comparaison des algorithmes de distance distributionnelle pour leur capacité d'estimation de l'erreur de déploiement. 3 métriques sont considérées dont une métrique de performance brute, et deux métriques pour l'incertitude.

Dataset	MAE (%)			
	$\mathcal{H}$-distance	ATC	OTDD	GDE
MNIST	11.28	15.68	7.90	**6.56**
CIFAR-10	5.16	7.514	8.04	**2.0**
TinyImageNet	7.92	11.26	10.87	**5.05**

Dataset	PICP (%)			
	$\mathcal{H}$-distance	ATC	OTDD	GDE
MNIST	93.75	93.75	**100.0**	96.88
CIFAR-10	**100.0**	**100.0**	**100.0**	**100.0**
TinyImageNet	96.3	**100.0**	88.89	**100.0**

Dataset	MPIW (%)			
	$\mathcal{H}$-distance	ATC	OTDD	GDE
MNIST	64.34	79.62	**45.21**	50.69
CIFAR-10	23.65	42.24	41.2	**14.27**
TinyImageNet	41.21	53.76	42.35	**34.96**

sur TinyImageNet. Seul OTDD obtient une meilleure fiabilité sur MNIST. Enfin, GDE a l'intervalle de prédiction MPIW le plus étroit sur CIFAR-10 et sur TinyImageNet. Seul OTDD obtient un intervalle plus étroit sur MNIST.

8.3.3 Conclusion

Finalement GDE est plus performant sur CIFAR-10 et TinyImageNet et OTDD est meilleur MNIST. Dans la suite, nous choisissons donc de nous concentrer sur GDE comme distance distributionnelle principale que nous allons appliquer à la base de données de mammographies pour classifier le cancer du sein Camelyon17 [Koh et al., 2021].

8.4 Effet du rééquilibrage des classes des données cibles

Questions de recherche La connaissance de la loi conditionnelle du jeu d'apprentissage offre-t-elle un avantage en termes d'estimation de l'erreur de déploiement et de qualité de l'incertitude ?

8.4.1 Matériel et méthodes

Pour répondre à cette question, nous avons utilisé un DenseNet-121 que nous avons utilisé pour apprendre sur chacun des 5 jeux de données correspondant aux hôpitaux de Camelyon17. Le résultat de cet apprentissage correspond à 5 modèles entraînés respectivement sur les hôpitaux 1 à 5.

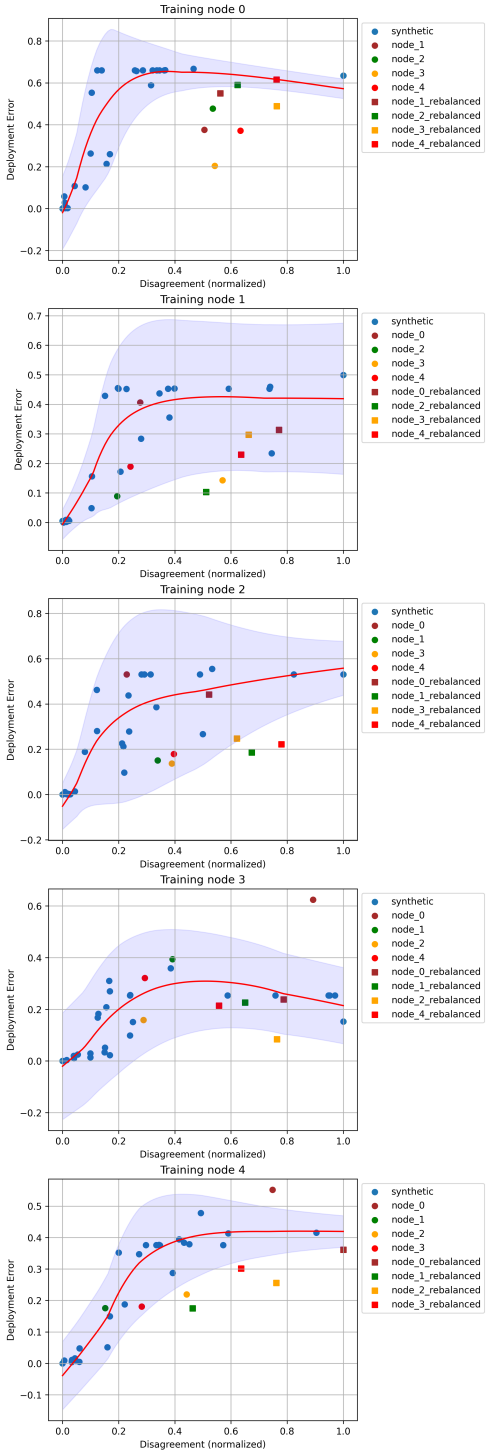


FIGURE 8.2: Les points bleus correspondent aux 26 domaines synthétiques, les points colorés correspondent hôpitaux cibles et les carrés colorés correspondent aux hôpitaux cibles rééquilibrés. La zone ombrée représente des intervalles de prédiction de 99%.

Ensuite, nous avons choisi 25 augmentations de données différentes. Nous avons déployé chacun des 5 modèles sur les jeux test sources respectifs perturbés par chacune des 25 augmentations sélectionnées.

Nous avons ensuite pour chaque modèle calculé 26 distances GDE sur les paires $\{(S,S),(S,\text{aug}_1(S)),\cdots,(S,\text{aug}_{25}(S))\}$. Avec le jeu de test issu de la distribution d'apprentissage, et aug_i l'augmentation de données i. Nous avons aussi calculé les distances GDE entre la distribution de l'hôpital source, et les 4 autres hôpitaux cibles. C'est-à-dire correspondants aux paires $\{(S,T_1),\cdots(S,T_4)\}$. Pour observer l'effet de la connaissance de la distribution conditionnelle du jeu d'apprentissage, nous avons aussi rebalancé les classes des hôpitaux cible suivant la distribution de l'hôpital source qu'on note $(\bar{T}_1,\cdots,\bar{T}_4)$. Ensuite nous avons calculé la distance GDE entre l'hôpital source et les 4 hôpitaux cibles rebalancés. C'est-à-dire correspondants aux paires $\{(S,\bar{T}_1),\cdots(S,\bar{T}_4)\}$, avec $\mathbb{P}^{\bar{T}_i}_{Y|X}=\mathbb{P}^{S}_{Y|X}$.

Nous avons ensuite entraîné un réseau le régresseur décrit en section précédentes sur les 26 couples distances distributionnelles résultantes. Enfin, nous avons déployé ce régresseur sur les 4 hôpitaux cibles $(T_1,\cdots,T_4)$ et sur les 4 hôpitaux cibles rééquilibrés $(\bar{T}_1,\cdots,\bar{T}_4)$, et nous avons calculé le MAE, PICP et MPIW. Pour avoir une perception qualitative de l'opération de rééquilibrage, nous montrons les résultats obtenus à la figure 8.2. Cependant pour réduire le facteur chance de nos résultats nous avons conduit 3 fois cette expérience d'apprentissage sur les 26 domaines synthétiques et d'évaluation sur les quatre hôpitaux cibles et les quatre hôpitaux cibles relancées. Cela nous permet d'obtenir des statistiques descriptives disponibles sur le tableau 8.2.

8.4.2 Résultats

TABLE 8.2: Rééquilibrage des classes.

Résultats aggrégés	MAE (%)	PICP (%)	MPIW (%)
hopitaux cibles	0.21 ± 0.01	**0.60** ± 0.0	0.45 ± 0.02
hopitaux cibles rééquilibrés	**0.15** ± 0.01	0.53 ± 0.05	0.34 ± 0.04

Nous observons que le rééquilibrage des classes des hôpitaux cibles tend à accroître la distance GDE. En effet, on observe que les carrés colorés sont plus à droite que les ronds colorés. Si l'on calcule l'augmentation moyenne des distances sur tous les hôpitaux, nous obtenons 0.25%. Mais la chute d'erreur de déploiement est moins perceptible. C'est-à-dire que les carrés colorés ne sont que légèrement au-dessus des ronds colorés. L'augmentation moyenne de l'erreur de déploiement n'est que de 0.01%.En regardant la différence agrégée, le rééquilibrage des classes des hôpitaux cibles réduit l'erreur de déploiement moyenne en passant de 0.21% à 0.15 %. En d'autres termes, l'estimation de l'erreur de déploiement est plus précise lorsque l'on rééquilibre les classes des hôpitaux cibles suivant la distribution des classes de l'hôpital source. Cependant, cela se fait au prix d'une perte de la fiabilité de l'incertitude avec un PICP passant de 0.60% à 0.53%. Cela s'explique, car lorsque les hôpitaux sont rééquilibrés, l'estimateur de l'erreur de déploiement admet une variance plus faible. De manière générale, même le PICP le plus élevé reste éloigné de la cible des 99 %. Cela s'explique par le manque de données.

8.4.3 Conclusion

La connaissance de la distribution des classes de l'hôpital d'origine peut permettre de rééquilibrer les classes du jeu de données cible et ainsi d'estimer plus finement l'erreur

de déploiement. Mais cela se fait au dépens de la fiabilité de l'incertitude autour de cette estimation de l'erreur. Aussi le PICP le plus élevé reste plus faible que l'objectif de capturer 99 % des données. Nous pensons qu'obtenir un grand nombre d'augmentations de données pourrait nous faire progresser vers cet objectif. Car nous pourrions fournir plus de données au régresseur. Cependant, est-ce que toutes les augmentations de données se valent ? C'est ce dont nous discutons dans la prochaine section.

8.5 Compromis exploration/exploitation de l'augmentation de données pour l'estimation de l'erreur de déploiement

Sur CIFAR-10, Deng et al. [Deng et al., 2021] ont utilisé la suite complète de CIFAR-10-C [Hendrycks and Dietterich, 2019] comme données hors distribution pour calculer le coefficient de corrélation. Ces ensembles de jeux perturbés sont composés de 15 corruptions du jeu original CIFAR-10 avec 5 niveaux d'intensité pour un total de 75 corruptions. De même, dans le cas de MNIST, Deng et ses collègues [Deng and Zheng, 2021] ont utilisé une méthodologie capable de générer un grand nombre de jeux de données perturbés à partir d'une seule procédure. En effet, le jeu de données source MNIST à été perturbé en changeant le fond noir par un patch tiré aléatoirement du jeu de données COCO [Lin et al., 2014]. Ainsi, chaque changement de fond correspondait à un nouveau domaine perturbé. Nous nous demandons en outre si 1000 jeux de données issus d'une seule procédure permettent d'obtenir les mêmes performances que 1000 glissements naturels différents. Deng et al. [Deng and Zheng, 2021] ont soulevé la question intéressante de la représentation des ensembles de données comme étant plus compliquée que la représentation d'exemples uniques, car le premier encode beaucoup plus d'informations que le second. Ainsi, nous sommes habitués à travers l'hypothèse $i.i.d$ à avoir des exemples identiquement et indépendamment distribués suivant une source de données. Mais que signifie vraiment une indépendance de jeux de données ? Ce n'est pas une notion définie. Dans la suite, nous ne prétendons pas avoir de réponse à cette question, cependant, nous trouvons le paradigme exploration/exploitation utile pour cristalliser notre intuition. En effet, dans la suite, on considérera que l'exploration de nouvelles augmentations de données correspond à obtenir des procédures indépendantes, et exploiter une augmentation de données correspond à des procédures non-indépendantes. Ainsi, nous en venons à notre question de recherche.

Question de recherche L'augmentation de données admet t-elle un compromis exploitation/exploration pour l'estimation de l'erreur de déploiement ?

8.5.1 Matériel et méthodes

Nous considérons que l'exploitation correspond à utiliser plusieurs niveaux de corruptions provenant d'une seule augmentation de données. Respectivement, nous considérons que l'exploration consiste à utiliser plusieurs augmentations de données différentes chacune prise avec un seul niveau de corruption. Ainsi, l'expérience précédente a été conduite dans le paradigme d'exploration. Pour contraster les deux phénomènes, dans cette expérience, nous

sélectionnons au hasard 5 augmentations parmi un groupe de 13 augmentations pouvant fournir au moins 5 niveaux de corruptions. Ensuite, ces 5 augmentations de données et leurs 5 niveaux de corruptions fournissent 25 domaines synthétiques. Ainsi, en notant aug_i^j l'augmentation i et le niveau de corruption j, nous calculons 26 distances GDE appliquées aux paires suivantes : $(S, S), (S, \text{aug}_1^1(S)), \cdots, (S, \text{aug}_1^5(S)), \cdots, (S, \text{aug}_5^1(S)), \cdots, (S, \text{aug}_5^5(S))$. Ensuite, pour confirmer l'effet du rééquilibrage, nous utilisons ces distances pour estimer l'erreur de déploiement sur les 4 hôpitaux cibles $(T_1, \cdots, T_4)$ et les 4 hôpitaux cibles rééquilibrés $(\bar{T}_1, \cdots, \bar{T}_4)$. Enfin, pour éviter que l'échantillonnage des 5 augmentations n'affecte l'estimation de l'erreur de déploiement, nous conduisons l'expérience 3 fois pour obtenir des statistiques descriptives.

8.5.2 Résultats

Nous présentons les résultats au tableau 8.3. L'effet du rééquilibrage reste le même qu'on explore plusieurs augmentations de données ou qu'on exploite quelques augmentations de données. En effet, le rééquilibrage réduit l'erreur de déploiement moyenne passant de 0.22% à 0.17.% Par ailleurs, le rééquilibrage réduit la fiabilité de l'incertitude avec un PICP passant de 0.43% à 0.38%, dû à une variance réduite, en effet, le MPIW passe de 0.39% à 0.27% pour les jeux rééquilibrés.

Nous observons que la stratégie d'exploitation, bien que plus facile à opérationaliser, est moins performante que l'exploration pour l'estimation de l'erreur de déploiement. En effet, les hôpitaux cibles voient leur MAE passer de 0.21% (tableau 8.2) à 0.22% (tableau 8.3). Le phénomène est identique sur les hôpitaux cibles rééquilibrés avec une MAE passant de 0.15% à 0.17%. Le désavantage de l'exploitation s'affirme davantage sur la fiabilité de l'incertitude avec un PICP passant de 0.6% à 0.43% sur les hôpitaux cibles et de 0.53% à 0.38% sur les hôpitaux rééquilibrés.

TABLE 8.3: Résultats en situation d'exploitation d'augmentation de données.

Résultats aggrégés	MAE	PICP	MPIW
hopitaux cibles	0.22 ± 0.01	**0.43** ± 0.24	0.39 ± 0.2
hopitaux cibles rééquilibrés	**0.17** ± 0.03	0.38 ± 0.06	0.27 ± 0.14

8.5.3 Conclusion

Il est définitivement plus avantageux d'utiliser des augmentations de données indépendantes pour favoriser l'estimation de l'erreur de déploiement. Car cela entraîne une estimation de l'erreur de déploiement plus précise, et une meilleure fiabilité de l'incertitude.

8.6 Conclusion

Notre première expérience nous a appris que GDE était la distance distributionnelle la plus performante pour estimer l'erreur de déploiement point. OTDD aussi était une

bonne distance, cependant cette dernière méthode requérait de connaître les étiquettes des données cible. Donc, dans un contexte où nous n'avons pas accès aux étiquettes des données de déploiement, GDE sort du lot. Ensuite, nous avons mis à l'épreuve GDE dans une situation réelle avec la base de données de mammographies Camelyon17 pour classifier le cancer du sein. Cette base de données permet de simuler une situation réelle, car un hôpital peut recevoir un modèle entraîné à partir de données sources issues d'un autre hôpital. L'hôpital recevant le modèle veut le déployer sur ses propres données, mais avec un doute sur la performance affichée par le modèle en question. Ainsi, dans notre seconde expérience, nous avons évalué si la connaissance de la distribution des classes des données sources procurait un avantage quelconque. Cela est important en pratique, car aujourd'hui, il n'y a pas encore de bonnes pratiques pour le partage de modèle. En outre, un hôpital va typiquement recevoir un modèle sans forcément avoir accès aux caractéristiques des données d'apprentissage. Nous voulons savoir dans quelle mesure la connaissance de ces caractéristiques peut servir dans le déploiement du modèle. Le résultat de cette expérience était mitigé, en effet connaître la distribution des classes des données sources permet d'obtenir une meilleure estimation de l'erreur de déploiement au détriment de la fiabilité de cette estimation de l'erreur. Nous savons que de manière générale, collecter plus de données pour toute tâche d'estimation permet d'obtenir une meilleure estimation. C'est la loi des grands nombres. Cependant, cette loi est vraie quand les données sont identiquement et indépendamment distribuées suivant une loi de probabilité. Cependant, dans notre cas, chaque donnée représente l'information d'un jeu de données. Une manière d'aborder le problème est de considérer que les jeux de données proviennent d'une méta distribution. Ainsi, pour améliorer la qualité d'estimation de l'erreur de déploiement, de nombreux auteurs ont exploité plusieurs sévérités d'un petit ensemble d'augmentation de données. Bien que l'idée d'indépendance entre jeux de données ne soit pas bien définie, dans la suite, nous avons utilisé le paradigme exploration/exploitation pour refléter deux approches allant vers plus ou moins d'indépendance entre des jeux de données augmentés. Ainsi, l'exploration de nouvelles augmentations de données permet d'obtenir des glissements de données plus indépendants les uns des autres que l'exploitation de plusieurs niveaux de sévérité d'une augmentation de données. Ainsi, dans notre troisième expérience, nous avons voulu comparer l'exploration et l'exploitation d'augmentations de données pour l'estimation de l'erreur de déploiement. Les résultats nous montrent que l'exploration de données est plus avantageuse pour l'estimation de l'erreur de déploiement. Nous pensons donc que les glissements dus à une exploration de plusieurs augmentations de données seront plus indépendants.

Chapitre Neuf

Conclusion et perspectives

Dans cette thèse, nous nous sommes intéressés au problème de robustesse. Ce problème se pose dès lors que nous avons un jeu d'apprentissage différent des données de déploiement. C'est un problème, car dans cette situation, nous ne pouvons pas garantir la fiabilité d'un modèle lors du déploiement. C'est ce problème qui a marqué le début de notre questionnement scientifique lors de notre travail sur les données acoustiques présenté au chapitre 6. En effet, contre toute attente, nous avions obtenu de meilleures performances sur le jeu de déploiement que sur les données de test. C'est cette profonde incompréhension qui a dirigé notre travail de thèse. Nous avons cherché à expliquer ce phénomène.

Notre revue de littérature nous a encouragés à considérer l'augmentation de données comme technique principale d'investigation. Augmenter un jeu de données permet de rendre un modèle invariant aux facteurs de variation encodés dans l'augmentation de données. Par exemple appliquer une légère rotation à une base de données d'images de chiens et chats permet à un modèle de devenir invariant aux rotations lorsqu'il sera déployé. Nous avons cherché au chapitre 7 à mieux cerner l'effet de l'augmentation de données sur la généralisation classique, et sur la généralisation hors distribution. Nos expériences nous ont appris que le meilleur usage de l'augmentation de données n'est pas de chercher à rendre un modèle plus robuste. En effet, sans connaissance de la loi de probabilité générant les données de déploiement, il est difficile à priori de savoir quelles augmentations de données seront utiles. Ainsi, il est possible d'appliquer une augmentation de données faisant chuter les performances du modèle lors du déploiement.

Nous illustrons cela en figure 9.1. Ainsi en figure 9.1 a, on se place dans un cas, ou on découvre un ensemble de facteurs de variation $\mathcal{T}$ existant dans les données cibles. Ainsi en augmentant la source de données $\mathbb{P}^S_{XY}$ avec $\mathcal{T}$, nous rendront un modèle de base invariant à ces facteurs de variation. Et selon sa perspective les données sources $\mathbb{P}^S_{XY}$ vont se rapprocher des données cibles $\mathbb{P}^T_{XY}$.

Cette réduction de distance est une mesure globale entre jeux de données. L'illustration locale de ce principe revient à pousser un exemple dans la direction du facteur de variation. Cela ne fera pas bouger le coût du modèle. Ainsi, si un exemple est au centre de la cuvette, appliquer une force (une corruption) ne le déplacera pas trop, car il restera dans un équilibre stable (On a discuté de cette notion au chapitre 4.). Ainsi, cette invariance va améliorer la généralisation sur les données cibles.

Maintenant, en figure 9.1 b, on se place dans le cas inverse, où on applique un ensemble de corruptions $\mathcal{T}$ non-pertinentes sur la distribution source $\mathbb{P}^S_{XY}$. C'est-à-dire que l'on suppose que ces facteurs de variation n'existent pas dans la distribution cible $\mathbb{P}^T_{XY}$. Alors,

appliquer ces perturbations $\mathcal{T}$ sur la distribution source $\mathbb{P}^S_{XY}$ va éloigner la distribution cible selon la perspective du modèle.

Cela est probablement dû au fait qu'en cherchant un modèle invariant à $\mathcal{T}$, on sélectionne des représentations qui n'ont aucune utilité pour classifier les données provenant de $\mathbb{P}^T_{XY}$. Et cela rend le modèle fragile aux facteurs de variation présents naturellement dans $\mathbb{P}^T_{XY}$. Ainsi la deuxième vignette sur la figure 9.1 b, illustre l'effet d'obtenir des invariances à des facteurs de variations inutiles. Le modèle a plus de chance de devenir fragile face aux facteurs de variation utiles. Ce qui signifie qu'une faible force appliquée dans la direction du facteur de variation utile fera passer une donnée dans une autre région de décision et entraînera une baisse de performance de classification.

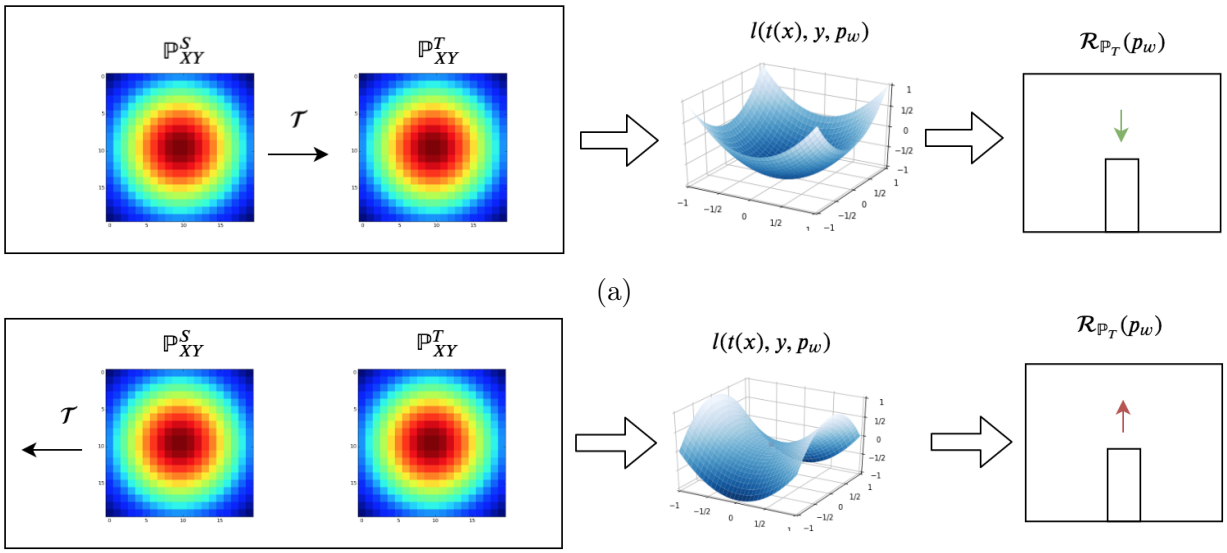


FIGURE 9.1: (a) On applique un ensemble de corruptions $\mathcal{T}$ pertinentes sur la distribution source $\mathbb{P}_S(x, y)$, cela a pour effet de rapprocher cette source de données corrompue de la distribution cible $\mathbb{P}_T(x, y)$. Le modèle devenant de plus en plus invariant n'aura pas de forte variation lorsque les données x seront transformées $t(x)$, le coût avant et après changera peu (équilibre stable). Ce qui entraînera une amélioration de la généralisation sur la distribution cible. (b) L'inverse se produit lorsque les corruptions $\mathcal{T}$ appliquées sur la distribution source $\mathbb{P}_S(x, y)$ ne sont pas pertinentes pour la distribution cible $\mathbb{P}_T(x, y)$. Cela a pour effet de sélectionner des représentations qui sont fragiles face aux facteurs de variations présents dans la distribution cible, et ainsi, cela crée un équilibre instable. Le coût peut beaucoup changer lorsque la données x est poussée dans la direction d'un facteur de variation des données cible $t(x)$. Car cette force, appliquée à la donnée, peut lui faire changer de région de décision. Au final, des corruptions non-pertinentes pour la distribution cible font baisser la généralisation sur cette dernière distribution.

Cette réalisation nous pousse à penser que la meilleure façon d'utiliser un ensemble d'augmentation de données est d'estimer l'erreur de déploiement. En effet, cela nous permet de renverser la malédiction de la dimension. La malédiction de la dimension réfère à la difficulté croissante à distinguer des objets lorsque la dimension augmente [Gorban et al., 2020]. Cette malédiction est due au phénomène de concentration, selon

lequel la différence entre la distance d'un point et de son plus proche voisin et la distance entre ce point et son plus lointain voisin tend vers 0 à mesure que la dimension augmente [Chih-Ming Hsu and Ming-Syan Chen, 2009]. Autrement dit, dans un espace de dimension élevée, les éléments peuvent être très différents en raison d'une petite variation dans une seule dimension, alors qu'elle reste constante dans toutes les autres dimensions. Deux objets peuvent être différents pour d'infimes variations dans chaque dimension. Pour faire court, en haute dimension, il est facile de déplacer un objet, et très difficile de percevoir la différence. Cela a des conséquences sur la tâche de classification en haute dimension, c'est pourquoi les premiers algorithmes d'apprentissage supervisé étaient axés sur l'élaboration de représentation pour réduire la dimension dans un espace où la distance pouvait être significative.

Nous disons que l'augmentation de données pour l'estimation de l'erreur de déploiement permet de renverser la malédiction de la dimension. En effet qu'une augmentation de données ait un facteur de variation inclus ou pas dans la distribution cible, nous pouvons toujours gagner de l'information sur la variance qu'elle entraînera sur la performance du modèle. Ainsi, la collecte d'augmentations de données indépendantes couplée à des distances distributionnelle permet d'estimer cette variance et d'anticiper l'erreur de déploiement.

Cela est très utile en pratique, en effet, l'estimation de l'erreur de déploiement nous indique la chute de performance attendue lorsqu'un modèle est déployé sur un domaine inconnu. Cependant, un mauvais niveau de certitude autour de cette estimation pourrait être dommageable pour certaines applications ayant une porté critique comme en médecine personnalisée par exemple. En effet, si un modélisateur observe seulement une faible erreur de déploiement, alors il est susceptible de déployer son modèle. Mais que se passerait-il si la variance autour de l'erreur de déploiement est importante en réalité, mais n'est pas estimée correctement. Cela signifierait que le modèle déployé pourrait avoir des comportements inattendus dans le monde réel. Ainsi, cette méthodologie :

1. pose l'importance de la collecte d'augmentations de données indépendantes pour une estimation de l'erreur de déploiement plus précise et plus fiable,
2. permet de comparer systématiquement divers modèles sur un nouveau domaine même en l'absence d'étiquettes et de sélectionner le plus robuste,
3. permet d'avancer vers un paradigme ou un modèle sera livré pour déploiement avec un estimateur d'erreur de déploiement. Nous faisons l'analogie avec la notice d'utilisation dans n'importe quelle boite de médicament. Ainsi, si la notice permet à l'utilisateur de savoir dans quelle mesure il peut faire confiance à un composé actif, en fonction de sa situation particulière et des effets secondaires. Alors l'estimateur de l'erreur de déploiement permettra à l'utilisateur de savoir dans quelle mesure il peut faire confiance au modèle livré pour ses données. Et décider s'il est nécessaire d'étiqueter ses données ou non.

Aujourd'hui, les compagnies pharmaceutiques doivent passer par plusieurs études cliniques avant de parfaitement comprendre les effets secondaires de leurs produits. Nous pensons que dans les décennies à venir, les modèles subiront des stress test extensifs et leur comportement sera documenté avant d'être livrés. Nous pensons que cela permettra à la société de tirer profit de la capacité des réseaux de neurones profonds avec plus d'équité et de justice.

Bibliographie

[Abadi et al., 2016] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., and Devin, M. (2016). Tensorflow : Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv :1603.04467*.

[Alvarez-Melis and Fusi, 2020] Alvarez-Melis, D. and Fusi, N. (2020). Geometric dataset distances via optimal transport. *Advances in Neural Information Processing Systems*, 33 :21428–21439.

[Anselmi et al., 2016] Anselmi, F., Leibo, J. Z., Rosasco, L., Mutch, J., Tacchetti, A., and Poggio, T. (2016). Unsupervised learning of invariant representations. *Theoretical Computer Science*, 633 :112–121.

[Arévalo et al., 2016] Arévalo, A., Niño, J., Hernández, G., and Sandoval, J. (2016). High-frequency trading strategy based on deep neural networks. In *International Conference on Intelligent Computing*, pages 424–436. Springer.

[Arjovsky, 2020] Arjovsky, M. (2020). *Out of Distribution Generalization in Machine Learning*. PhD thesis, New York University.

[Arjovsky et al., 2020] Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2020). Invariant Risk Minimization. *arXiv :1907.02893 [cs, stat]*.

[Balan et al., 2015] Balan, A. K., Rathod, V., Murphy, K. P., and Welling, M. (2015). Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3438–3446.

[Baldi et al., 2014] Baldi, P., Sadowski, P., and Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1) :1–9.

[Barbu et al., 2019] Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J., and Katz, B. (2019). Objectnet : A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32.

[Bartholomä, 2006] Bartholomä, A. (2006). Acoustic bottom detection and seabed classification in the German Bight, southern North Sea. *Geo-Marine Letters*, 26(3) :177.

[belgianchocolate-FlickR, 2017] belgianchocolate-FlickR (2017). Cow farm in Qatari desert struggles amid Arab boycott.

[Ben-David et al., 2006] Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2006). Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19.

[Ben-Tal et al., 2009] Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust Optimization*, volume 28. Princeton university press.

[Bengio et al., 2013a] Bengio, Y., Courville, A., and Vincent, P. (2013a). Representation learning : A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1798–1828.

[Bengio et al., 2013b] Bengio, Y., Courville, A., and Vincent, P. (2013b). Representation learning : A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1798–1828.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2) :157–166.

[Bishop, 1994] Bishop, C. M. (1994). Mixture density networks. *Neural Computing Research Group Report : NCRG/94/004*.

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv :1505.05424*.

[Bousquet and Elisseeff, 2002] Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. *Journal of machine learning research*, 2(Mar) :499–526.

[Brautaset et al., 2020] Brautaset, O., Waldeland, A. U., Johnsen, E., Malde, K., Eikvil, L., Salberg, A.-B., and Handegard, N. O. (2020). Acoustic classification in multifrequency echosounder data using deep convolutional neural networks. *ICES Journal of Marine Science*.

[Brehmer, 2006] Brehmer, P. (2006). Fisheries Acoustics : Theory and Practice, 2nd edn. *Fish and Fisheries*, 7.

[Brehmer et al., 2019] Brehmer, P., Sancho, G., Trygonis, V., Itano, D., Dalen, J., Fuchs, A., Faraj, A., and Taquet, M. (2019). Towards an autonomous pelagic observatory : Experiences from monitoring fish communities around drifting FADs. *Thalassas : An International Journal of Marine Sciences*, 35(1) :177–189.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., and Askell, A. (2020). Language models are few-shot learners. *arXiv preprint arXiv :2005.14165*.

[Buolamwini and Gebru, 2018] Buolamwini, J. and Gebru, T. (2018). Gender shades : Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*, pages 77–91. PMLR.

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1) :41–75.

[Chen et al., 2021] Chen, M., Goel, K., Sohoni, N. S., Poms, F., Fatahalian, K., and Ré, C. (2021). Mandoline : Model Evaluation under Distribution Shift. In *International Conference on Machine Learning*, pages 1617–1629. PMLR.

[Chen et al., 2014] Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient hamiltonian monte carlo. In *International Conference on Machine Learning*, pages 1683–1691.

[Chih-Ming Hsu and Ming-Syan Chen, 2009] Chih-Ming Hsu and Ming-Syan Chen (2009). On the Design and Applicability of Distance Functions in High-Dimensional Data Space. *IEEE Transactions on Knowledge and Data Engineering*, 21(4) :523–536.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation : Encoder-decoder approaches. *arXiv preprint arXiv :1409.1259*.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv :1412.3555*.

[Cubuk et al., 2019] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment : Learning augmentation strategies from data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 113–123.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4) :303–314.

[D'Amour et al., 2020] D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., and Hoffman, M. D. (2020). Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv :2011.03395*.

[Dao et al., 2019] Dao, T., Gu, A., Ratner, A. J., Smith, V., De Sa, C., and Ré, C. (2019). A kernel theory of modern data augmentation. *Proceedings of machine learning research*, 97 :1528.

[Dawid et al., 2016] Dawid, A. P., Musio, M., and Ventura, L. (2016). Minimum scoring rule inference. *Scandinavian Journal of Statistics*, 43(1) :123–138.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet : A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee.

[Deng, 2012] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6) :141–142.

[Deng et al., 2021] Deng, W., Gould, S., and Zheng, L. (2021). What Does Rotation Prediction Tell Us about Classifier Accuracy under Varying Testing Environments ? In *International Conference on Machine Learning*, pages 2579–2589. PMLR.

[Deng and Zheng, 2021] Deng, W. and Zheng, L. (2021). Are labels always necessary for classifier accuracy evaluation ? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15069–15078.

[Denos et al., 2017] Denos, K., Ravaut, M., Fagette, A., and Lim, H. (2017). Deep learning applied to underwater mine warfare. In *OCEANS 2017 - Aberdeen*, pages 1–7.

[DeVries and Taylor, 2017] DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv :1708.04552*.

[Elsahar and Gallé, 2019] Elsahar, H. and Gallé, M. (2019). To Annotate or Not ? Predicting Performance Drop under Domain Shift. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2163–2173, Hong Kong, China. Association for Computational Linguistics.

[Erhan et al., 2010] Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning ? *Journal of Machine Learning Research*, 11(Feb) :625–660.

[Federici et al., 2017] Federici, M., Ullrich, K., and Welling, M. (2017). Improved bayesian compression. *arXiv preprint arXiv :1711.06494*.

[Filos et al., 2019] Filos, A., Farquhar, S., Gomez, A. N., Rudner, T. G., Kenton, Z., Smith, L., Alizadeh, M., de Kroon, A., and Gal, Y. (2019). A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks. *arXiv preprint arXiv :1912.10481*.

[Foote et al., 1991] Foote, K. G., Knudsen, H. P., Korneliussen, R. J., Nordbo/, P. E., and Ro/ang, K. (1991). Postprocessing system for echo sounder data. *The Journal of the Acoustical Society of America*, 90(1) :37–47.

[Gal and Ghahramani, 2016] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation : Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059.

[Gal et al., 2017] Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR.

[Ganin et al., 2016] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1) :2096–2030.

[Gao et al., 2017] Gao, R., Chen, X., and Kleywegt, A. J. (2017). Distributional robustness and regularization in statistical learning. *arXiv preprint arXiv :1712.06050*.

[Garg et al., 2022] Garg, S., Balakrishnan, S., Lipton, Z. C., Neyshabur, B., and Sedghi, H. (2022). Leveraging Unlabeled Data to Predict Out-of-Distribution Performance. *International Conference on Learning Representations*.

[Geirhos et al., 2019] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. (2019). ImageNet-trained CNNs are biased towards texture ; increasing shape bias improves accuracy and robustness. *arXiv :1811.12231 [cs, q-bio, stat]*.

[Geirhos et al., 2018] Geirhos, R., Temme, C. R. M., Rauber, J., Schütt, H. H., Bethge, M., and Wichmann, F. A. (2018). Generalisation in humans and deep neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7549–7561.

[Gneiting and Raftery, 2007] Gneiting, T. and Raftery, A. E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477) :359–378.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press.

[Gorban et al., 2020] Gorban, A. N., Makarov, V. A., and Tyukin, I. Y. (2020). High-Dimensional Brain in a High-Dimensional World : Blessing of Dimensionality. *Entropy*, 22(1) :82.

[Graves, 2011] Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356.

[Graves, 2013] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv :1308.0850*.

[Guillory et al., 2021] Guillory, D., Shankar, V., Ebrahimi, S., Darrell, T., and Schmidt, L. (2021). Predicting with Confidence on Unseen Distributions. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1114–1124, Montreal, QC, Canada. IEEE.

[Gulrajani and Lopez-Paz, 2020a] Gulrajani, I. and Lopez-Paz, D. (2020a). In Search of Lost Domain Generalization. In *International Conference on Learning Representations*.

[Gulrajani and Lopez-Paz, 2020b] Gulrajani, I. and Lopez-Paz, D. (2020b). In search of lost domain generalization. *International Conference on Learning Representations*.

[Ha and Eck, 2018] Ha, D. and Eck, D. (2018). A Neural Representation of Sketch Drawings. In *International Conference on Learning Representations*.

[Hampel, 2005] Hampel, F. R., editor (2005). *Robust Statistics : The Approach Based on Influence Functions*. Wiley Series in Probability and Mathematical Statistics. Wiley, New York, digital print edition.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv :1512.03385 [cs]*.

[Hendrycks et al., 2021a] Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., and Guo, M. (2021a). The many faces of robustness : A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349.

[Hendrycks and Dietterich, 2019] Hendrycks, D. and Dietterich, T. (2019). Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*.

[Hendrycks et al., 2019] Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. (2019). Augmix : A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv :1912.02781*.

[Hendrycks et al., 2021b] Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. (2021b). Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271.

[Hinton et al., 2012] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., and Sainath, T. N. (2012). Deep neural networks for acoustic modeling in speech recognition : The shared views of four research groups. *IEEE Signal processing magazine*, 29(6) :82–97.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8) :1735–1780.

[Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708.

[Huang and Belongie, 2017] Huang, X. and Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510.

[Huber, 1964] Huber, P. J. (1964). Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1) :73–101.

[Huber and Ronchetti, 2009] Huber, P. J. and Ronchetti, E. M. (2009). Robust statistics. 2nd john wiley & sons. *Hoboken, NJ*, 2.

[Jewson et al., 2018] Jewson, J., Smith, J. Q., and Holmes, C. (2018). Principles of Bayesian inference using general divergence criteria. *Entropy*, 20(6) :442.

[Jiang et al., 2022] Jiang, Y., Nagarajan, V., Baek, C., and Kolter, J. Z. (2022). Assessing Generalization of SGD via Disagreement.

[Jumper et al., 2021] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., and Potapenko, A. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873) :583–589.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv :1312.6114 [cs, stat]*.

[Kirilenko et al., 2017] Kirilenko, A., Kyle, A. S., Samadi, M., and Tuzun, T. (2017). The flash crash : High-frequency trading in an electronic market. *The Journal of Finance*, 72(3) :967–998.

[Klambauer et al., 2017] Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 971–980.

[Ko et al., 2015] Ko, T., Peddinti, V., Povey, D., and Khudanpur, S. (2015). Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

[Koh et al., 2021] Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., and Gao, I. (2021). Wilds : A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR.

[Korneliussen, 2004] Korneliussen, R. J. (2004). The Bergen echo integrator post-processing system, with focus on recent improvements. *Fisheries Research*, 68(1-3) :159–169.

[Krizhevsky and Hinton, 2009] Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

[Krueger et al., 2021] Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Le Priol, R., and Courville, A. (2021). Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR.

[Kukačka et al., 2017] Kukačka, J., Golkov, V., and Cremers, D. (2017). Regularization for Deep Learning : A Taxonomy.

[Lakshminarayanan et al., 2017] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413.

[LeCun and Bengio, 1995] LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10) :1995.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553) :436–444.

[Lemley et al., 2017] Lemley, J., Bazrafkan, S., and Corcoran, P. (2017). Smart augmentation learning an optimal data augmentation strategy. *Ieee Access*, 5 :5858–5869.

[Li et al., 2017] Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. (2017). Deeper, broader and artier domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5542–5550.

[Li et al., 2019] Li, Y., Huang, C., Ding, L., Li, Z., Pan, Y., and Gao, X. (2019). Deep learning in bioinformatics : Introduction, application, and perspective in the big data era. *Methods*, 166 :4–21.

[Lim et al., 2019] Lim, S., Kim, I., Kim, T., Kim, C., and Kim, S. (2019). Fast autoaugment. In *Advances in Neural Information Processing Systems*, pages 6662–6672.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco : Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer.

[Llorens et al., 2008] Llorens, D., Prat, F., Marzal, A., Vilar, J. M., Castro, M. J., Amengual, J.-C., Barrachina, S., Castellanos, A., Boquera, S. E., and Gómez, J. A. (2008). The UJIpenchars Database : A Pen-Based Database of Isolated Handwritten Characters. In *LREC*.

[Louizos et al., 2017] Louizos, C., Ullrich, K., and Welling, M. (2017). Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298.

[Louizos and Welling, 2016] Louizos, C. and Welling, M. (2016). Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716.

[Louizos and Welling, 2017] Louizos, C. and Welling, M. (2017). Multiplicative normalizing flows for variational bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org.

[MacLennan et al., 2004] MacLennan, D. N., Copland, P. J., Armstrong, E., and Simmonds, E. J. (2004). Experiments on the discrimination of fish and seabed echoes. *ICES Journal of Marine Science*, 61(2) :201–210.

[Mescheder et al., 2017] Mescheder, L., Nowozin, S., and Geiger, A. (2017). Adversarial variational bayes : Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2391–2400. JMLR. org.

[Miller et al., 2021] Miller, J. P., Taori, R., Raghunathan, A., Sagawa, S., Koh, P. W., Shankar, V., Liang, P., Carmon, Y., and Schmidt, L. (2021). Accuracy on the line : On the strong correlation between out-of-distribution and in-distribution generalization. In *International Conference on Machine Learning*, pages 7721–7735. PMLR.

[Mitchell, 1997] Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL : McGraw Hill*, 45(37) :870–877.

[Molchanov et al., 2017] Molchanov, D., Ashukha, A., and Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org.

[Moreno-Torres et al., 2012] Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., and Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern recognition*, 45(1) :521–530.

[Mu and Gilmer, 2019] Mu, N. and Gilmer, J. (2019). MNIST-C : A Robustness Benchmark for Computer Vision. *arXiv preprint arXiv :1906.02337*.

[Nalisnick and Smyth, 2018] Nalisnick, E. and Smyth, P. (2018). Learning priors for invariance. In *International Conference on Artificial Intelligence and Statistics*, pages 366–375.

[Neklyudov et al., 2017] Neklyudov, K., Molchanov, D., Ashukha, A., and Vetrov, D. P. (2017). Structured bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems*, pages 6775–6784.

[Nguyen et al., 2015] Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep Neural Networks are Easily Fooled : High Confidence Predictions for Unrecognizable Images. *arXiv :1412.1897 [cs]*.

[Nichol, 2016] Nichol, K. (2016). Painter by numbers, Wikiart, 2016. *URL https ://www. kaggle. com/c/painter-by-numbers/overview*.

[Niu et al., 2019] Niu, H., Gong, Z., Ozanich, E., Gerstoft, P., Wang, H., and Li, Z. (2019). Deep learning for ocean acoustic source localization using one sensor. *arXiv preprint arXiv :1903.12319*.

[Niu et al., 2017] Niu, H., Reeves, E., and Gerstoft, P. (2017). Source localization in an ocean waveguide using supervised machine learning. *The Journal of the Acoustical Society of America*, 142(3) :1176–1188.

[Nix and Weigend, 1994] Nix, D. A. and Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 Ieee International Conference on Neural Networks (ICNN'94)*, volume 1, pages 55–60. IEEE.

[Ona and Mitson, 1996] Ona, E. and Mitson, R. B. (1996). Acoustic sampling and signal processing near the seabed : The deadzone revisited. *ICES Journal of Marine Science*, 53(4) :677–690.

[Ovadia et al., 2019] Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. (2019). Can you trust your model's uncertainty ? Evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems*, 32.

[Pan and Yang, 2010] Pan, S. J. and Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10) :1345–1359.

[Park et al., 2019] Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. (2019). Specaugment : A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv :1904.08779*.

[Pawlowski et al., 2017] Pawlowski, N., Brock, A., Lee, M. C., Rajchl, M., and Glocker, B. (2017). Implicit weight uncertainty in neural networks. *arXiv preprint arXiv :1711.01297*.

[Pearce et al., 2018] Pearce, T., Brintrup, A., Zaki, M., and Neely, A. (2018). High-quality prediction intervals for deep learning : A distribution-free, ensembled approach. In *International Conference on Machine Learning*, pages 4075–4084. PMLR.

[Perrot et al., 2018] Perrot, Y., Brehmer, P., Habasque, J., Roudaut, G., Behagle, N., Sarré, A., and Lebourges-Dhaussy, A. (2018). Matecho : An Open-Source Tool for Processing Fisheries Acoustics Data. *Acoustics Australia*, pages 1–8.

[Quinonero-Candela et al., 2008] Quinonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2008). *Dataset Shift in Machine Learning*. Mit Press.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[Samuels, 2016] Samuels, G. (2016). Denmark develops 'super grass' to cut cow burp emissions. https ://www.independent.co.uk/news/science/denmark-scientists-develop-super-grass-to-cut-cow-burp-methane-emissions-a7358156.html.

[Sarr et al., 2021] Sarr, J.-M. A., Brochier, T., Brehmer, P., Perrot, Y., Bah, A., Sarré, A., Jeyid, M. A., Sidibeh, M., and El Ayoubi, S. (2021). Complex data labeling with deep learning methods : Lessons from fisheries acoustics. *Isa Transactions*, 109 :113–125.

[Sarr et al., 2020] Sarr, J. M. A., Yannakakis, G. N., Liapis, A., Bah, A., and Cambier, C. (2020). Djehuty : A Mixed-Initiative Handwriting Game for Preschoolers. In *International Conference on the Foundations of Digital Games*, pages 1–4.

[Sarré et al., 2018] Sarré, A., Krakstad, J.-O., Brehmer, P., and Mbye, E. M. (2018). Spatial distribution of main clupeid species in relation to acoustic assessment surveys in the continental shelves of Senegal and The Gambia. *Aquatic Living Resources*, 31 :9.

[Savage, 1971] Savage, L. J. (1971). Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 66(336) :783–801.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11) :2673–2681.

[Settles, 2010] Settles, B. (2010). Active learning literature survey. Technical report.

[Shalev-Shwartz and Ben-David, 2014] Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning : From Theory to Algorithms*. Cambridge university press.

[Shi et al., 2017] Shi, J., Sun, S., and Zhu, J. (2017). Kernel implicit variational inference. *arXiv preprint arXiv :1705.10119*.

[Shimodaira, 2000] Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2) :227–244.

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1) :60.

[Shrivastava et al., 2017] Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. (2017). Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2107–2116.

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., and Lanctot, M. (2016). Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587) :484–489.

[Simard et al., 1991] Simard, P., Victorri, B., LeCun, Y., and Denker, J. S. (1991). Tangent prop-a formalism for specifying selected invariances in an adaptive network. In *NIPS*, volume 91, pages 895–903. Citeseer.

[Simmonds and MacLennan, 2008] Simmonds, J. and MacLennan, D. N. (2008). *Fisheries Acoustics : Theory and Practice*. John Wiley & Sons.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*.

[Socha et al., 1996] Socha, D. G., Watkins, J. L., and Brierley, A. S. (1996). A visualization-based post-processing system for analysis of acoustic data. *ICES Journal of Marine Science*, 53(2) :335–338.

[Sokolic et al., 2017a] Sokolic, J., Giryes, R., Sapiro, G., and Rodrigues, M. (2017a). Generalization error of invariant classifiers. In *Artificial Intelligence and Statistics*, pages 1094–1103. PMLR.

[Sokolic et al., 2017b] Sokolic, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. D. (2017b). Robust Large Margin Deep Neural Networks. *IEEE Transactions on Signal Processing*, 65(16) :4265–4280.

[Sun et al., 2017] Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

[Swendsen and Wang, 1987] Swendsen, R. H. and Wang, J.-S. (1987). Nonuniversal critical dynamics in Monte Carlo simulations. *Physical review letters*, 58(2) :86.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[Tanner and Wong, 1987] Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, 82(398) :528–540.

[Taori et al., 2020] Taori, R., Dave, A., Shankar, V., Carlini, N., Recht, B., and Schmidt, L. (2020). Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33 :18583–18599.

[Torralba and Efros, 2011] Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE.

[Tran et al., 2017] Tran, T., Pham, T., Carneiro, G., Palmer, L., and Reid, I. (2017). A bayesian data augmentation approach for learning deep models. In *Advances in Neural Information Processing Systems*, pages 2797–2806.

[Ullrich et al., 2017] Ullrich, K., Meeds, E., and Welling, M. (2017). Soft weight-sharing for neural network compression. *arXiv preprint arXiv :1702.04008*.

[Van Dyk and Meng, 2001] Van Dyk, D. A. and Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1) :1–50.

[Vapnik, 1992] Vapnik, V. (1992). Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*, pages 831–838.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, \., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[Vilalta and Drissi, 2002] Vilalta, R. and Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2) :77–95.

[Villalobos et al., 2013] Villalobos, H., Martínez, M. O. N., Santos-Molina, J. P., González-Máynez, V. E., de los Ángeles Martínez-Zavala, M., Hermand, J.-P., and Brehmer, P. (2013). Acoustic estimation of Pacific sardine biomass in the Gulf of California during the spring 2008-2012. In *2013 IEEE/OES Acoustics in Underwater Geosciences Symposium*, pages 1–4. IEEE.

[Volpi et al., 2018] Volpi, R., Namkoong, H., Sener, O., Duchi, J. C., Murino, V., and Savarese, S. (2018). Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems*, 31.

[Wang et al., 2019a] Wang, H., Ge, S., Lipton, Z., and Xing, E. P. (2019a). Learning robust global representations by penalizing local predictive power. *Advances in Neural Information Processing Systems*, 32.

[Wang et al., 2021] Wang, J., Lan, C., Liu, C., Ouyang, Y., Zeng, W., and Qin, T. (2021). Generalizing to unseen domains : A survey on domain generalization. *arXiv preprint arXiv :2103.03097*.

[Wang and Deng, 2018] Wang, M. and Deng, W. (2018). Deep Visual Domain Adaptation : A Survey. *Neurocomputing*, 312 :135–153.

[Wang et al., 2019b] Wang, W., Zheng, V. W., Yu, H., and Miao, C. (2019b). A survey of zero-shot learning : Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2) :1–37.

[Wang et al., 2019c] Wang, Y., Pan, X., Song, S., Zhang, H., Huang, G., and Wu, C. (2019c). Implicit semantic data augmentation for deep networks. *Advances in Neural Information Processing Systems*, 32.

[Wen et al., 2018] Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. (2018). Flipout : Efficient Pseudo-Independent Weight Perturbations on Mini-Batches. In *International Conference on Learning Representations*.

[Williams, 2016] Williams, D. P. (2016). Underwater target classification in synthetic aperture sonar imagery using deep convolutional neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2497–2502.

[Wilson et al., 2019] Wilson, B., Hoffman, J., and Morgenstern, J. (2019). Predictive inequity in object detection. *arXiv preprint arXiv :1902.11097*.

[Winkler et al., 1996] Winkler, R. L., Munoz, J., Cervera, J. L., Bernardo, J. M., Blattenberger, G., Kadane, J. B., Lindley, D. V., Murphy, A. H., Oliver, R. M., and Ríos-Insua, D. (1996). Scoring rules and the evaluation of probabilities. *Test*, 5(1) :1–60.

[Wong et al., 2019] Wong, E., Rice, L., and Kolter, J. Z. (2019). Fast is better than free : Revisiting adversarial training. In *International Conference on Learning Representations*.

[Xiao et al., 2017] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist : A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv :1708.07747*.

[Xie et al., 2019] Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2019). Unsupervised data augmentation. *arXiv preprint arXiv :1904.12848*.

[Xu and Mannor, 2012] Xu, H. and Mannor, S. (2012). Robustness and generalization. *Machine learning*, 86(3) :391–423.

[Yang et al., 2020] Yang, S., Liu, L., and Xu, M. (2020). Free Lunch for Few-shot Learning : Distribution Calibration. In *International Conference on Learning Representations*.

[Yun et al., 2019] Yun, S., Han, D., Chun, S., Oh, S. J., Yoo, Y., and Choe, J. (2019). CutMix : Regularization Strategy to Train Strong Classifiers With Localizable Features. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6022–6031. IEEE Computer Society.

[Zagoruyko and Komodakis, 2016] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv :1605.07146*.

[Zhang et al., 2018] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). Mixup : Beyond Empirical Risk Minimization. In *International Conference on Learning Representations*.

[Zhang et al., 2019] Zhang, L., Wang, X., Yang, D., Sanford, T., Harmon, S., Turkbey, B., Roth, H., Myronenko, A., Xu, D., and Xu, Z. (2019). When unseen domain generalization is unnecessary ? rethinking data augmentation. *arXiv preprint arXiv :1906.03347*.

[Zhou et al., 2021] Zhou, K., Liu, Z., Qiao, Y., Xiang, T., and Loy, C. C. (2021). Domain generalization in vision : A survey. *arXiv preprint arXiv :2103.02503*.

Chapitre Dix

Annexes

10.1 Préparation des données

10.1.1 Prétraitement en mer

Le prétraitement des données a été réalisé à l'aide du logiciel Matecho [Perrot et al., 2018]. Le signal rétrodiffusé des 500 m supérieurs de la colonne d'eau a été extrait et l'écho à chaque profondeur a été interpolé sur une grille régulière avec une résolution verticale de 20 cm. Pourtant, lors du prétraitement, la profondeur de l'océan pour chaque ping a été estimée par un algorithme automatique qui recherchait le gradient maximal du signal acoustique, correspondant généralement au signal acoustique au fond, c'est ce que nous appellons "Fond brut" dans le tableau 10.1. Puis en post-traitement, l'échogramme et la détection automatique de la ligne de fond ont été visualisés par un expert qui a effectué manuellement la correction de la ligne de fond, cela correspond au "Fond corrigé" dans le tableau 10.1.

10.1.2 Préparation des données pour l'apprentissage

Mise à l'échelle des données de 2011 et 2015

Trois transformations ont été appliquées à partir des données brutes pour extraire un format standardisé pour 2011 et 2015.

La première transformation a été effectuée pour obtenir le même nombre de lignes pour nos jeux de données, qui variaient initialement légèrement (tableau 10.1). Pour ce faire, nous avons supprimé les 31 premières lignes de l'échogramme 2011 complet ($\sim$ 6 mètres), ainsi que les 17 premières lignes de l'échogramme 2015 ($\sim$ 3,3 mètres). Cela était nécessaire car il y avait fréquemment des signaux forts dans les premières rangées qui ont par la suite perturbé les premiers processus d'apprentissage. Une telle transformation était nécessaire pour permettre aux mêmes réseaux de neurones d'être entraînés et appliqués aux deux sous-échantillons d'ensembles de données. Enfin, il nous restait 2550 lignes dans chacun des ensembles de données prétraités.

La deuxième transformation visait à réduire la taille de l'ensemble de données en supprimant les pings pour lesquels aucun apprentissage ne pouvait être effectué, c'est-à-dire lorsque les pings dans l'échogramme n'atteignaient pas le fond. Cela se produit lorsque le navire est situé dans des zones de plus de 500 mètres de profondeur, car seuls les 500 premiers mètres ont été enregistrés lors de la campagne en mer. Ces pings ont été supprimés

TABLE 10.1: Résumé des variables présentes dans chaque dimension des jeux de données bruts d'enquêtes en mer de 2011 et 2015 et les étiquettes associées. Les principales variables utilisées pour traiter les données et en outre pour mettre en place le modèle d'apprentissage automatique sont décrites. S_v signifie la force de rétrodiffusion de volume (en dB).

	Profondeur	Échogramme	Fond brut	Fond corrigé
Dimensions du jeu de 2015 : [Nb. lignes, Nb. colonnes]	[2567, 1]	[2567, 2 000 000]	[1, 2 000 000]	[1, 2 000 000]
Dimensions du jeu de 2011 : [Nb lignes, Nb colonnes]	[2581, 1]	[2581, 2 661 003]	[1, 2 661 003]	[1, 2 661 003]
Unité	Mètre (m)	S_v in dB	Mètre (m)	Mètre (m)
Description de la matrice	Vecteur vertical : chaque cellule correspond à une valeur en mètre	Matrice : chaque colonne est un ping donné, et chaque ligne une valeur en dB correspondant à une profondeur	Vecteur horizontal : chaque colonne donne la profondeur du fond trouvée par le Procédure automatique	Vecteur horizontal : chaque colonne donne la profondeur du fond corrigée par l'expert

à l'aide d'un filtre basé sur des seuils. Nous avons distingué les échogrammes avec et sans fond, en fonction de la présence ou non d'une forte rétrodiffusion (> -32 dB), considérée comme une signature de fond.

Enfin, comme nous avions besoin que chaque ping ait les mêmes dimensions pour un calcul ultérieur et que les algorithmes numériques ne pouvaient pas traiter les valeurs NaN, nous avons remplacé toutes les valeurs NaN par -200 dB, une valeur qui correspond aux valeurs enregistrées les plus faibles dans les échogrammes (-199 dB pour 2011 et -198 dB pour 2015). Cette opération peut être vue comme le remplacement de valeurs non enregistrées par du bruit. En effet, les valeurs inférieures à ~ -90 dB ne sont jamais (ou rarement) prises en compte dans l'analyse acoustique des pêches. Ainsi, les valeurs NaN ont été traitées comme des zones situées sous le fond et non atteintes par le signal acoustique. Les tailles des ensembles de données formatés de 2011 et 2015 sont résumées dans le tableau 6.1).

Étiquetage des données

Le but de la procédure est de classer les données (ici, les pings acoustiques) en deux classes pour orienter l'intervention de l'expert lors de son travail de correction de l'échogramme brut. La première classe regroupe les pings ayant peu de chance de nécéssiter une correction et la seconde classe regroupe les pings ayant une forte probabilité de nécessiter une correction. Nous avons étiqueté chaque ping restant (après formatage ; voir la section précédente) comme appartenant à l'une de ces deux classes selon la distance entre la correction expert (variable " CleanBottom " , Tableau 1) et la profondeur de fond

initialement prédite (variable " Bottom " , Tableau 1). Ainsi, si |CleanBottom - Bottom| $<$ 3,31, les requêtes ping étaient étiquetées " correction faible " , c'est-à-dire qu'aucune correction d'expert n'était nécessaire, tandis que si |CleanBottom - Bottom| $>=$ 3,31, les pings étaient étiquetés " correction forte " , c'est-à-dire qu'une correction d'expert était probablement nécessaire. Des exemples de ces deux classes sont illustrés à la figure 6.2.

La valeur seuil de 3,31 a été choisie en comparant la précision obtenue après une itération d'apprentissage pour différentes valeurs de seuil allant de 1,00 à 5,00 avec un pas de 0,01, et nous avons sélectionné le seuil qui a donné nous la meilleure précision de classification. La distribution de ces classes a varié considérablement de 2011 à 2015. En effet, la classe de forte correction représentait 13% des pings en 2011, alors qu'elle ne représentait que 1% en 2015. Comme on peut le voir sur la figure 6.1, il n'y a pas de modèle clair des pings à corriger (voir couleur rouge sur la carte) . Ce qui nous à motiver à utiliser les méthodes d'apprentissage pour automatiser la recherche du ping, avec une forte probabilité de nécessiter une forte correction.

10.2 Étude comparative pour la classification de données d'acoustique active

Nous avons évalué plusieurs procédures d'apprentissage pour tirer parti des données étiquetées passées pour lesquelles un expert a déjà corrigé la prédiction de résultat, ainsi de grands ensembles de données d'entraînement sont disponibles. La tâche spécifique étudiée était d'identifier les parties de l'échogramme nécessitant que l'expert corrige la ligne du fond. Notre première contribution est de proposer une comparaison de différents algorithmes d'apprentissage automatique pour cette tâche. Quatre algorithmes d'apprentissage sont comparés : les forêts aléatoires, les réseaux de neurones convolutifs, les machines à vecteurs de support et les réseaux de neurones pleinement connectés. Leurs hyperparamètres ont été séléctionnés en utilisant un algorithme d'optimisation bayésienne. Cela a conduit à l'identification des CNN comme l'algorithme d'apprentissage le plus adapté.

Vision d'ensemble de la méthodologie d'échantillonnage

Nous décrivons ici des ensembles de données qui ont été utilisés pour (1) optimiser les hyperparamètres, (2) comparer les performances des algorithmes d'apprentissage et (3) identifier l'algorithme d'apprentissage le mieux adapté avec un ensemble de données mixte. Pour commencer, les 2 000 000 premiers pings de chaque campagne ont été sélectionnés pour constituer le pool de données. 2.000.000 était le maximum que nous pouvions garder en mémoire. Dans ce qui suit, pour faciliter les notation, tous les 1000 pings seront traités notés 1K.

Pour l'optimisation bayésienne, nous avons utilisé un ensemble de données de 100 000 pings extraits au hasard de l'ensemble de données complet du pool de 2011 (Fig 10.1 a). Pour comparer les performances des algorithmes d'apprentissage, des ensembles de données avec des tailles successives de 200K, 400K, 600K, 800K et 1000K pings ont également été extraits de l'ensemble de données du pool de 2011 (respectivement 2,0, 4, 6,0, 8,0 et 10,0 Go). Le plus grand ensemble de données utilisé pour l'apprentissage était constitué de 1 000 000 de pings, car il était fastidieux de télécharger de grands ensembles de données

sur des clusters GPU en ligne. De plus, pour comparer différents algorithmes d'apprentissage, chacun de ces ensembles de données a été divisé en 90% des pings pour l'ensemble d'apprentissage et 10% pour l'ensemble de test (Fig 10.1 b).

Pour évaluer l'effet de l'apprentissage en domaine mixte, 100K, 300K, 500K et 550K pings ont été échantillonnés à partir de l'ensemble de données du pool de 2011 pour l'apprentissage, ainsi que 100K pings se succédant du pool de 2015. Ces 100 000 pings ont ensuite été divisés au hasard en deux ensembles de données de 50 000 pings, l'un qui servirait d'ensemble de données de validation et l'autre qui servirait à l'entraînement croisé (Fig 10.1 c).

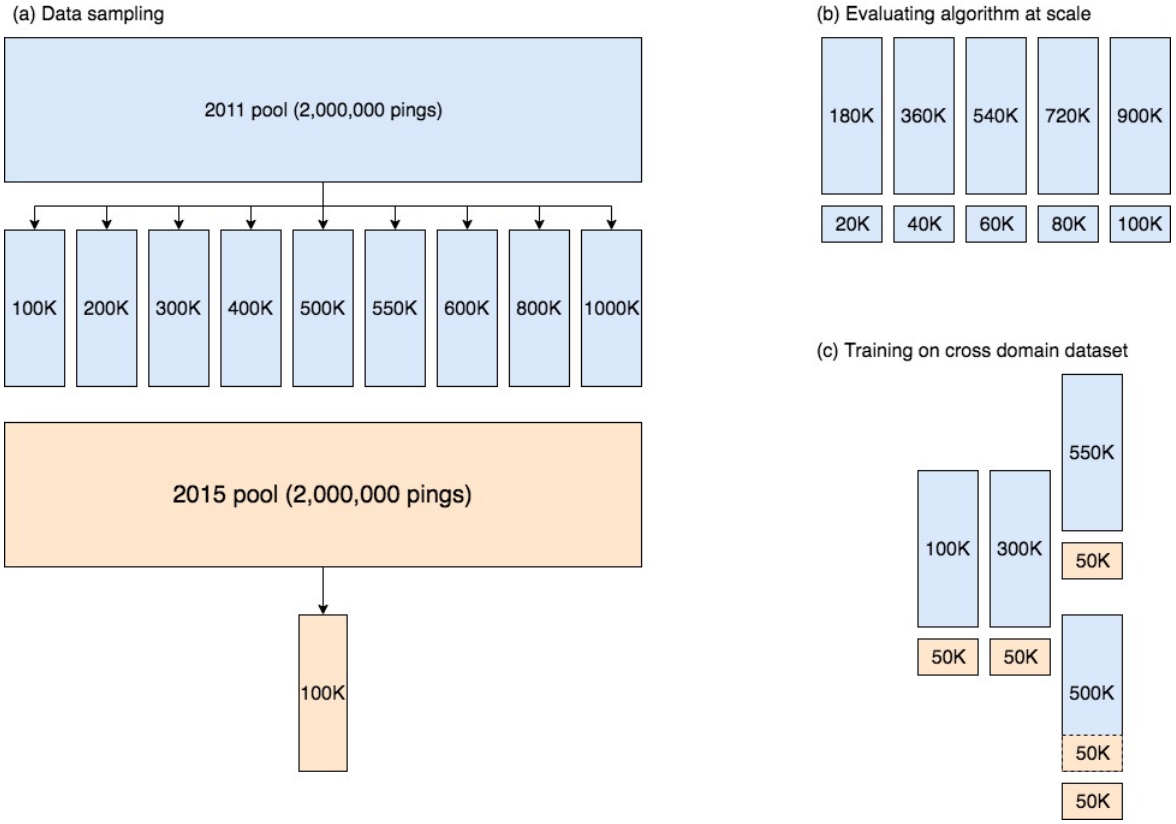


FIGURE 10.1: Méthodologie d'échantillonnage. La sous-figure (a) résume la stratégie d'échantillonnage des données. La sous-figure (b) montre les tailles des ensembles d'apprentissage et de test comme décrit dans l'expérience 2.2.1. La sous-figure (c) montre le réglage proposé pour l'expérience 2.2.3. La couleur bleue correspond aux données de 2011 et la couleur orange correspond aux données de 2015. Dans (b) et (c), les cases les plus longues correspondent à l'ensemble de données d'entraînement. En (b) les petites cases sont utilisées comme ensemble de test et en (c) comme ensemble de validation.

10.2.1 Comparaison des performances des algorithmes d'apprentissage

La meilleure précision de validation obtenue pour un ensemble d'hyperparamètres a été obtenue en moins de 10 itérations pour SVM et en environ 22 itérations pour RF. Cela est

cohérent, en effet RF a deux hyperparamètres et SVM un seul. Cependant les FFNN et CNN se sont améliorés mais n'ont pas atteint une performance maximale définitive même après 50 itérations (Fig 10.2). L'ensemble final d'hyperparamètres pour chaque algorithme d'apprentissage est affiché dans le tableau 10.2.

Les performances des algorithmes d'apprentissage ont ensuite été comparées pour des apprentissage avec des tailles de données croissantes. SVM n'a pas augmenté ses performances lors de l'augmentation de la taille de l'ensemble de données ; étonnamment, il semble même s'aggraver (Fig 10.3 a). Le réseau de neurones Feed-Forward (FFNN) présente la plus grande variabilité, pour chaque taille de l'ensemble d'entraînement, la pire valeur est presque approximativement la même et résulte d'un blocage à un minimum local lors de l'apprentissage. À part pour les jeux de données 800K et 100K ou l'algorithme a eu des performances relativement stable (Fig 10.3 b). Cependant la variabilité était plus forte pour le jeu 100K que 800K, cela impliquant une meilleure performance après entraînement sur le jeu 800K. On peut conclure à une moins bonne stabilité de l'apprentissage avec un FFNN, même si en moyenne l'augmenation de la taille des données améliore la généralisation. RF n'a affiché presque aucune variabilité et ses performances ont bénéficié de l'augmentation de la taille de l'ensemble de données.

Enfin, les réseaux de neurones convolutifs (CNN) ont obtenu le score de précision le plus élevé lors de l'entraînement à partir du jeu 400K et sur tous les jeux comportant plus de données (Fig 10.3). Il est en outre plus stable à entraîner car il affiche moins de variabilité que FFNN tout en ayant toujours le meilleur score de précision de test maximum. À l'exception des machines à vecteurs de support (SVM), nous observons que tous les algorithmes d'apprentissage présentent une amélioration des performances avec l'ajout de données. FFNN ayant la plus grande variabilité dans ses gains car il est souvent bloqué sur des minima locaux lorsque l'apprentissage commence. Random Forest (RF) augmente également régulièrement sa précision lors du passage à l'échelle. Les réseaux de neurones convolutifs (CNN), qui ne sont jamais restés dans les minima locaux, réalisent le gain le plus élevé.

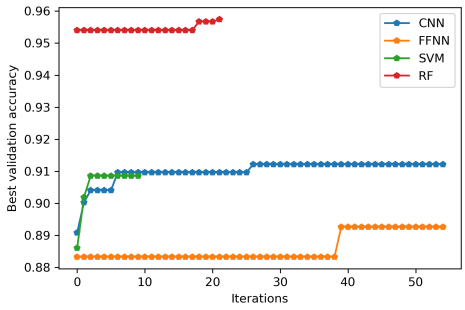


FIGURE 10.2: Illustration de la sélection des hyperparamètres avec la procédure d'optimisation bayésienne pour les forêts aléatoires (RF), les machines à vecteurs de support (SVM), les réseaux de neurones Feed-Forward (FFNN) et les réseaux de neurones convolutifs (CNN). L'optimisation se termine en moins de 20 itérations pour SVM et RF, mais n'est toujours pas terminé après 50 itérations avec FFNN et CNN.

Comme prévu avec l'apprentissage profond, augmenter la taille des données d'entraîne-

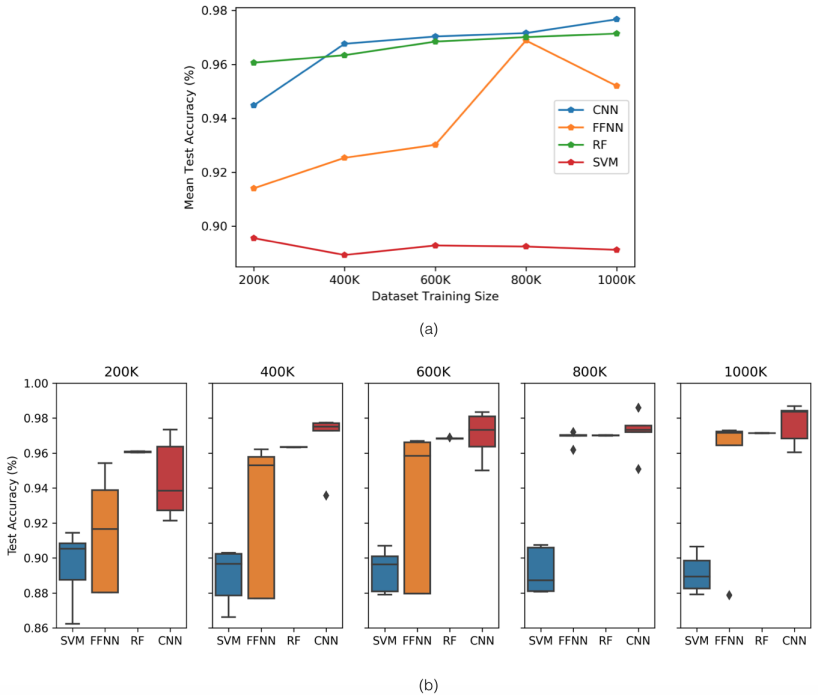


FIGURE 10.3: (a) Précision moyenne obtenue sur le jeu test pour les forêts aléatoires (RF), les machines à vecteurs de support (SVM), les réseaux de neurones Feed-Forward (FFNN) et les réseaux de neurones convolutifs (CNN) tout en faisant passer à l'échelle les jeux d'apprentissage de 200 000 à 1 000 000 pings sur les relevés en mer de 2011. (b) Statistiques récapitulatives de la précision obtenu par chaque algorithme d'apprentissage après 5 répétitions du processus d'apprentissage.

ment conduit presque toujours à de meilleures performances sur l'ensemble d'entraînement mais aussi un jeu test issue de la même source que le jeu d'apprentissage, mais aussi sur un jeu test provenant d'une autre source.

10.2.2 Comparaison d'algorithmes d'apprentissage : pourquoi CNN se démarque

Le SVM a mal fonctionné, en effet c'est parce que nous avons utilisé un noyau linéaire sur un problème de grande dimension, cela était nécessaire pour permettre au SVM d'apprendre sur de grands ensembles de données. Aussi comme SVM ne permet pas d'augmenter le nombre de paramètres utilisés pour apprendre, il a une asymptote plus faible que les autres modèles (Fig 10.3 a). Alors que RF a la meilleure précision lorsqu'il est entraîné avec un petit ensemble de données (200K), CNN s'est avéré avoir de meilleures performances dès que les ensembles de données sont devenus plus important. Pourtant, le RF est la deuxième meilleure option et suivait de près le CNN même si la précision obtenue avec 1000K pings semblait avoir fait décoller la précision de CNN. Une autre propriété intéressante des RF est qu'elles ne présentent pratiquement aucune variabilité lorsqu'elles sont comparées entre différentes versions d'apprentissage sur le même ensemble de données. Cette propriété permet au modélisateur d'être sûr qu'il a obtenu le meilleur résultat possible avec un RF à la

TABLE 10.2: Hyperparameters search space and value found using a Python Bayesian optimization library (GyOpt).

Random forest		
Hyperparameter	Search Space	Value
Number of tree range	[10, 10 000]	187
Min. samples leaf	[20, 50]	24
Support vector machines		
Hyperparameter	Search Space	Value
Alpha	[0.0001, 0.1]	0.077
Feed-forward neural network		
Hyperparameter	Search Space	Value
Number of neuron : fully connected layer 1	[5, 600]	75
Number of neuron : fully connected layer 2	[5, 320]	105
Number of neuron : fully connected layer 3	[5, 120]	95
Dropout rate : fully connected layer 3	[0, 1]	0.6
Convolutional neural network		
Hyperparameter	Search Space	Value
Kernel 1	[5, 60]	5
Kernel 2	[5, 60]	59
Kernel 3	[5, 60]	19
Number of neuron : fully connected layer 1	[5, 600]	260
Number of neuron : fully connected layer 2	[5, 320]	319
Number of neuron : fully connected layer 3	[5, 120]	101
Dropout rate : fully connected layer 3	[0, 1]	0.9

fin de l'apprentissage.

Le principal avantage des réseaux de neurones, qu'ils soient FFNN ou CNN, est la capacité qu'ils donnent au modélisateur d'augmenter le nombre de paramètres en fonction du problème d'apprentissage. En conséquence, ils sont modulaires et peuvent s'adapter à de nombreux types de problèmes. Le principal inconvénient du FFNN est sa grande variabilité d'un apprentissage à l'autre. En effet, le lecteur peut observer que le pire résultat obtenu pour chaque jeu de données est sensiblement le même autour de 87% (Fig 10.3.b). Au moins sur un apprentissage fait sur cinq le FFNN n'a est resté bloqué sur cette performance de 87%. Cela peut se produire lorsque le réseau est bloqué dans un minimum local et ne peut pas en sortir pendant l'apprentissage. En revanche, lors de l'entraînement, le CNN n'est jamais tombé dans un minimum local (Fig 10.3.b). En effet, les couches convolutives ont compressé les données d'origine dans une représentation de dimension inférieure qui est ensuite transmise à un réseau de neurones entièrement connecté. En conséquence, les caractéristiques importantes sont résumées par les couches convolutives, [LeCun and Bengio, 1995, Goodfellow et al., 2016, LeCun et al., 2015]. Et l'optimisation se fait dans un espace de dimension inférieure. La convolution s'est révélée très prometteuse pour la compréhension des images. La comparaison ci-dessus suggère que la convolution à une dimension est la représentation la plus adaptée pour traiter les données acoustiques actives. Enfin, le CNN présente les meilleures performances avec des ensembles de données d'entraînement de tailles 400K, jusqu'à 1000K. [Niu et al., 2017] ont trouvé des performances comparables pour chaque modèle : SVM, RF et FFNN dans la localisation de source sous-marine. Nos résultats différent de ceux de [Niu et al., 2017] car nous avons trouvé un net avantage à utiliser des réseaux de neurones et plus particulièrement des réseaux de neurones convolutifs. Cette différence peut s'expliquer par le fait qu'ils ont utilisé un ensemble de données relativement petit pour l'apprentissage (1 380 exemples pour l'apprentissage et 120 exemples pour les tests), et leur travail portait également sur des données acoustiques passives simulées. En revanche, notre travail a été effectué sur un ensemble de données réel et plus grand (200 000 exemples à 1 000 000 d'exemples) de données acoustiques actives. L'obtention de meilleurs résultats sur des ensembles de données plus volumineux avec l'apprentissage profond est abordée dans [Goodfellow et al., 2016] et [Sun et al., 2017]. De plus, des applications pratiques ont été trouvées dans la localisation de sources sous-marines par [Niu et al., 2019] qui a utilisé un réseau résiduel de 50 couches pour apprendre sur des dizaines de millions d'exemples.