Années: 2022 – 2023
Laboratoire: ISIR
Équipe: PIRoS

Rapport de stage

Estimation de l'incertitude pour les réseaux de neurones profonds

Victor Guillard
September 2023
M2MO

Sous la supervision de Kévin Bailly et Arnaud Dapogny

Abstract

Les réseaux de neurones profonds ont rendu possible des avancées considérables dans de nombreux domaines applicatifs de l'intelligence artificielle tels que la vision par ordinateur, la reconnaissance vocale ou le traitement naturel du langage. Toutefois, bien que ces réseaux fournissent des performances impressionnantes, leur intégration dans des domaines où des erreurs pourraient être catastrophiques, comme la médecine ou les voitures autonomes, nécessite de nouveaux outils pour mesurer efficacement la confiance que l'on peut accorder à leur prédictions. Les domaines de la classification sélective et l'estimation de l'incertitude sont donc essentiels aujourd'hui pour assurer la sûreté des utilisateurs de ces systèmes. Dans ce cadre là, nous avons tout d'abord fait un tour d'horizon de ce domaines, puis nous avons étudié les métriques d'évaluation des méthodes d'estimation de l'incertitude et enfin nous avons ré-implémenté des méthodes état de l'art comme Monte Carlo Dropout ou ConfidNet et nous avons analysé leurs performances ainsi que leurs limitations. Nous avons ensuite exploré des méthodes nouvelles pour améliorer les performances de l'état de l'art.

Contents

1 Introduction

1.1 Contexte

Ce stage de recherche a été effectué dans le cadre du Master 2 Modélisation aléatoire de l'université Paris-Cité dans l'équipe PIRoS de l'Insitut Des Systèmes Intelligents et de Robotique (ISIR).

L'ISIR est sous la double tutelle de Sorbonne Université et du Centre National de la Recherche Scientifique (CNRS). L'ISIR est rattaché à l'UFR919 d'ingénierie relevant de la Faculté des Sciences et de l'Ingénierie à Sorbonne Univeristé et en rattachement principal à l'Institut des sciences de l'information et de leurs interactions (INS2I) et, en rattachement secondaire, à l'Institut des sciences biologiques (INSB) et à l'Institut des sciences de l'ingénierie et des systèmes (INSIS) pour le CNRS.

L'équipe Perception, Interaction, Robotiques Sociales (PIRoS) mène des recherches fondamentales et appliquées dans le domaine de la perception sociale, de l'interaction sociale et de la robotique sociale. Les recherches reposent sur des méthodologies, d'une part de l'interaction humain-machine, de l'apprentissage machine, du traitement du signal et de la vision par ordinateur, et d'autre part de la psychologie, des neurosciences, des sciences cognitives et de la psychiatrie.

1.2 Motivations et Objectifs

Dans le cadre de l'apprentissage automatique, la capacité de comprendre et de mesurer l'incertitude associée à une prédiction est cruciale. Cette compréhension s'ancre dans l'idée qu'il est aussi important de savoir ce que l'on ignore que ce que l'on sait. Le domaine de la prédiction sélective offre des outils permettant de répondre à cette problématique, l'idée étant de chercher à minimiser les erreurs en évitant de formuler des prédictions dans des situations d'incertitude.

L'estimation de l'incertitude joue un rôle central dans ce domaine et sert à mesurer la confiance qu'un modèle peut avoir dans ses propres prédictions. Cette estimation n'est pas une simple métrique : elle est le mécanisme principal permettant à un modèle de décider s'il doit s'abstenir de prédire ou non. Dans des contextes critiques, comme celui des véhicules autonomes ou de la médecine, la capacité d'un système à évaluer son propre degré d'incertitude peut permettre d'éviter des échecs potentiellement catastrophique.

Alors que nous nous orientons vers une dépendance accrue envers l'intelligence artificielle pour des tâches de plus en plus complexes, la reconnaissance et la gestion de l'incertitude deviennent fondamentales. Elles garantissent que les systèmes d'apprentissage automatique soient non seulement efficaces, mais aussi capables d'estimer la qualité de leurs prédictions. Ce stage à pour objectifs d'explorer l'état de l'art des méthodes de l'estimation de l'incertitude, les différentes manières de mesurer la qualité de ces méthodes ainsi que de proposer des méthodes nouvelles palliant les limitations des méthodes existantes.

1.3 Remerciements

Je tiens à remercier mes deux encadrants de stage Arnaud Dapogny et Kévin Bailly, ainsi que les doctorants Edouard Yvinec et Jules Bonnard, avec qui j'ai passé six mois très enrichissants, pour m'avoir beaucoup appris tout autant techniquement que théoriquement dans le domaine du Deep Learning, ainsi que pour leur accueil chaleureux et leur bonne humeur qui ont rendu mon travail a leurs côtés aussi agréable qu'épanouissant.

2 Préliminaires

Nous nous intéresserons dans ce rapport à l'estimation de l'incertitude dans le cadre supervisé de la classification multi-classe. Nous utiliserons les termes incertitude et confiance de manière interchangeable, l'un étant l'opposé de l'autre. Nous commencerons tout d'abord par rappeler quelques notions élémentaires du Deep Learning puis nous présenterons de le domaine de l'estimation de l'incertitude.

2.1 Définition du problème et notations

Nous utiliserons les notations et définitions suivantes : soit X un espace de features (par exemple, les pixels bruts d'une image) et $Y = \{1, 2, \ldots, k\}$ un ensemble de labels pour les classes. Considérons une distribution inconnue $P(X, Y)$ sur $X \times Y$.

Un classifieur f est défini par :

$$f : X \to Y$$

Son risque s'écrit $R(f|P) = \mathbb{E}_{P(X,Y)}[\ell(f(x), y)]$, où $\ell : Y \times Y \to \mathbb{R}^+$ est une fonction de coût donnée.

À partir d'un ensemble annoté $D_n = \{(x_i, y_i)\}_{i=1}^n \subseteq (X \times Y)$ échantillonné de manière i.i.d. à partir de $P(X, Y)$, le risque empirique du classifieur f est donné par :

$$\hat{r}(f|D_n) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

2.2 Réseaux de neurones profonds

Nous nous intéresserons spécifiquement dans ce rapport au classfieurs f de type réseaux de neurones. Un réseau de neurones profond (DNN, Deep Neural Network) est une architecture algorithmique inspirée de la manière dont les neurones biologiques fonctionnent. Les DNN sont utilisés pour modéliser et traiter des structures de données complexes grâce à leurs capacités d'apprentissage profond.

2.2.1 Architecture et fonctionnement

Un réseau de neurones est une composition de fonctions linéaires et non-linéaires réparties en plusieurs couches. Le fonctionnement d'une couche densement connectée s'écrit mathématiquement comme suit :

$$v_k = a_k \left(\sum_{j=1}^{n} w_{kj} z_j + b_k \right)$$

où :

- v_k est la sortie de la couche k.
- a_k est la fonction d'activation de la couche k.

- w_{kj} sont les poids associés à la connexion entre la couche k et la couche j.
- z_j est la sortie de la couche précédente j.
- b_k est le biais de la couche k.

Une fonction d'activation détermine la sortie d'un neurone. Elle ajoute une non-linéarité qui permet au réseau de modéliser des relations complexes. Plusieurs fonctions d'activation sont couramment utilisées, telles que:

- Fonction sigmoïde: Utilisée traditionnellement, elle présente l'inconvénient de pouvoir saturer et ralentir l'apprentissage. $f(z) = \frac{1}{1+e^{-z}}$
- Fonction ReLU (Rectified Linear Unit): Devenue populaire pour sa simplicité et son efficacité dans les couches cachées des réseaux profonds. $f(z) = \max(0, z)$
- Fonction tanh: Une variante du sigmoïde qui retourne des valeurs entre -1 et 1. $f(z) = \tanh(z)$
- Fonction GELU (Gaussian Error Linear Unit): Une fonction d'activation $f(z) = 0.5z(1 + \tanh(\sqrt{\frac{2}{\pi}}(z + 0.044715z^3)))$

2.2.2 Réseaux de neurones convolutifs (CNN)

Les CNN sont une catégorie spécialisée de réseaux de neurones conçue pour traiter des données structurées en grilles, comme des images. En utilisant des convolutions discrètes, ils peuvent détecter des caractéristiques locales, comme des bords ou des textures, dans différentes parties d'une image.

Les CNN utilisent principalement trois types de couches:

- **Couches de convolution:** Appliquent des filtres pour détecter des caractéristiques à différents emplacements de l'image.

$$(I * K)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j)K(x - i, y - j)$$

- **Couches de pooling:** Simplifient l'information en réduisant la dimensionnalité, ce qui rend le réseau plus robuste et réduit le risque de surapprentissage.
- **Couches densement connectée:** Traitent les caractéristiques extraites pour produire une sortie finale, comme une classification d'image.

2.2.3 Classification multi-classes

Pour les tâches de classification multiclasse, la couche de sortie est souvent une couche softmax. Cette fonction convertit un vecteur de valeurs réelles en un vecteur de probabilités, où chaque entrée correspond à la probabilité qu'une entrée appartienne à une classe particulière.

La fonction softmax est définie comme :

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Pour une entrée $x \in X$, le vecteur $f(x) = (f(x)_1, \ldots, f(x)_k) \in \mathbb{R}^k$ représente les activations softmax de la dernière couche, où chaque composante du vecteur représente la probabilité que x appartienne à la classe correspondante.

2.3 Prédiction sélective

Nous allons maintenant présenter le paradigme de la prédiction sélective [4](aussi appelée classification sélective ou classification avec option de rejet). Ce domaine du deep learning a pour principale motivation de réduire le taux d'erreur en s'abstenant de prédire en cas de doute, tout en conservant une couverture aussi large que possible. Le but idéal de la classification sélective serait d'avoir un classificateur équipé d'un 'bouton' permettant de contrôler précisément le taux d'erreur réel souhaité (qui devrait être garanti avec une grande probabilité), tout en maintenant la couverture du classificateur aussi élevée que possible.

L'objectif est d'apprendre un classificateur sélectif (f, g), où f est un classificateur standard et g est une fonction de rejet. Le classificateur sélectif doit permettre un contrôle complet et sûr sur le risque réel. La méthode idéale devrait être capable de classifier des échantillons avec n'importe quel niveau de risque souhaité tout en conservant le taux de couverture optimal. Il est raisonnable de supposer que cette performance optimale ne peut être obtenue que si la paire (f, g) est formée ensemble. Cependant, nous considérons un cadre plus simple où un classificateur de type DNN f est déjà donné, et notre objectif est d'apprendre une fonction de rejet g qui garantira avec une grande probabilité un taux d'erreur souhaité. En d'autres termes, pour un classificateur f donné, un niveau de confiance δ, et un risque souhaité r^*, le but est d'obtenir un classificateur sélectif (f, g) dont l'erreur de test ne sera pas supérieure à r^* avec une probabilité d'au moins $1 - \delta$.

Pour pouvoir construire cette fonction de rejet g, il est courant d'utiliser des techniques liées à l'estimation de l'incertitude et la prédiction de succès ou d'erreurs, notamment une fonction de score de confiance, que l'on notera $\kappa(x, i, |f)$, où $x \in X$ et $i \in Y$. La fonction κ devrait quantifier la confiance dans la prédiction que x appartienne à la classe i, en fonction d'informations extraites de f. Une fonction de score κ devrait induire un ordre partiel sur les points dans X, et n'est donc pas obligée de distinguer entre les points ayant le même score. Par exemple, pour n'importe quel classifieur softmax f, la fonction de score de confiance utilisée traditionnellement comme baseline est la MCP (Maximum Class Probability) :

$$\kappa(x, i|f) = f(x)_i$$

Ceci peut être expliqué par l'interprétation probabiliste naturelle de la fonction softmax (toutes les valeurs sont non négatives et somment à 1). Un κ optimal pour un f donné devrait refléter une monotonie par rapport à la fonction de perte théorique, c'est-à-dire que pour tout échantillons $(x_1, y_1) \sim P(X, Y)$, et $(x_2, y_2) \sim P(X, Y)$,

$$\kappa(x_1, \hat{y}_f(x)|f) \leq \kappa(x_2, \hat{y}_f(x)|f) \Leftrightarrow \mathbf{Pr}_P[\hat{y}_f(x_1) \neq y_1] \geq \mathbf{Pr}_P[\hat{y}_f(x_2) \neq y_2].$$

2.4 Estimation de l'incertitude

2.4.1 Définitions

Nous allons maintenant présenter les notions d'estimation de l'incertitude dans le cadre du Deep Learning. On appelle estimation de l'incertitude le calcul d'un score κ associé à chaque prédiction qui mesure à quel point l'on peut se fier ou non à cette prédiction.

Il existe deux principales sources d'incertitude : l'incertitude aléatoire (également connue sous le nom de "data uncertainty") et l'incertitude épistémique (aussi appelée "model uncertainty").

L'incertitude aléatoire survient en raison de la complexité inhérente aux données, telle que du bruit ou une superposition des distributions des classes. Il est important de noter que l'incertitude aléatoire est irréductible car inhérente à la distribution des données.

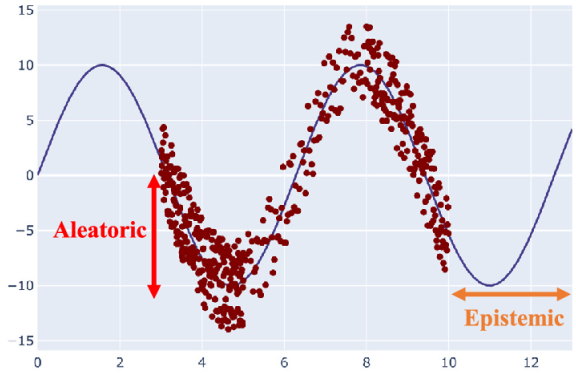

Figure 1: Incertitudes aléatoire et épistémique
[2]

De ces incertitudes découlent deux problématiques différentes dans le domaine de l'incertitude : la calibration et la prédiction d'échecs ou de succès de prédiction.

La calibration d'un DNN concerne la correspondance entre la confiance que le modèle affiche dans ses prédictions et la précision réelle de ces prédictions. Un modèle bien calibré présente une concordance entre la probabilité prédite et la précision empirique. Prenons un exemple : si un modèle de classification affirme qu'une image donnée a une probabilité de 90% d'être un chat, alors si nous avons 100 images avec une probabilité de 90%, environ 90 d'entre elles devraient réellement être des chats. Un modèle non calibré pourrait être trop confiant ou pas assez. Cela peut poser des problèmes, en particulier dans les applications critiques comme la médecine, où une fausse confiance dans une prédiction pourrait avoir de graves conséquences. Il existe plusieurs méthodes pour améliorer la calibration d'un modèle, dont certaines impliquent des ajustements post-entraînement (comme la recalibration Platt ou l'ajout d'un paramètre de température) où les sorties du modèle sont ajustées pour mieux refléter la réalité.

La problématique de prédiction d'erreurs et de succès se concentre davantage sur le calcul d'un score associé à chaque prédiction et à l'ordre de ces prédictions quant au score. Ainsi la situation idéale serait de réussir à séparer parfaitement les

échecs des succès de prédiction grâce au score calculé. Dans cette problématique, la valeur de l'incertitude associée à une prédiction n'est pas importante par elle même mais par rapport aux valeurs d'incertitude des autres prédictions et donc de son classement dans l'ensemble de test étudié.

Nous nous intéresserons principalement dans ce rapport à l'étude de la prédiction d'erreurs et de succès qui est beaucoup plus pertinente dans le cadre de la prédiction sélective.

2.4.2 Sources d'incertitude

Nous considérons quatre étapes différentes allant de l'information brute à une prédiction par un réseau de neurones avec des incertitudes quantifiées, à savoir :

- la collecte des données d'entraînement
- le processus de sélection et d'entraînement du DNN
- le choix du modèle d'inférence
- la modélisation des incertitudes dues au modèle et aux données

En pratique, ces quatre étapes contiennent plusieurs sources potentielles d'incertitude et d'erreurs, qui affectent à leur tour la prédiction finale d'un réseau de neurones. Les cinq facteurs que nous considérons comme les plus essentiels pour la cause de l'incertitude dans les prédictions d'un DNN sont :

- la variabilité dans les situations du monde réel
- les erreurs inhérentes aux systèmes de mesure
- les erreurs dans la spécification de l'architecture du DNN
- les erreurs dans la procédure d'entraînement du DNN
- les erreurs causées par des données inconnues

Acquisition de Données

Dans le contexte de l'apprentissage supervisé, l'acquisition de données décrit le processus où les mesures x et les variables cibles y sont générées pour représenter une situation ω du monde réel appartenant à un espace Ω. Dans le monde réel, une réalisation de ω pourrait par exemple être un oiseau, x une photo de cet oiseau, et y une label indiquant "oiseau". Pendant la mesure, du bruit aléatoire peut survenir et de l'information peut être perdue. Nous modélisons cette aléa dans x par :

$$x|\omega \sim p_{x|\omega}.$$

La variable cible correspondante y est obtenue de manière similaire, où la description est soit basée sur une autre mesure, soit le résultat d'un processus d'étiquetage. Dans les deux cas, la description peut être affectée par du bruit et des erreurs, ce que nous énonçons comme :

$$y|\omega \sim p_{y|\omega}.$$

Un réseau de neurones est entraîné sur un ensemble de données fini de réalisations de $x|\omega_i$ et $y|\omega_i$ basé sur n situations du monde réel $\omega_1, \ldots, \omega_N$,

$$D_n = \{x_i, y_i\}_{i=1}^n.$$

Lors de la collecte des données d'entraînement, deux facteurs peuvent causer de l'incertitude dans un réseau de neurones entraîné sur ces données. Tout d'abord, l'espace d'échantillonnage Ω devrait être suffisamment couvert par les données d'entraînement $x_1, \ldots, x_N$ pour $\omega_1, \ldots, \omega_N$. On supposera donc qu'en général pour un nouvel échantillon x^* :

$$x^* \neq x_i \text{ pour toutes les exemples d'entraînement } x_i.$$

Par conséquent, la cible doit être estimée sur la base du modèle de réseau de neurones entraîné, ce qui conduit directement au premier facteur d'incertitude.

Facteur I : Variabilité dans les situations du monde réel La plupart des environnements du monde réel sont hautement variables et presque constamment affectés par des changements. Ces changements affectent des paramètres comme par exemple la température, l'éclairage, l'encombrement, la taille et la forme des objets physiques. Les changements dans l'environnement peuvent également affecter l'expression des objets, comme par exemple les plantes après la pluie semblent très différentes des plantes après une sécheresse. Lorsque les situations du monde réel changent par rapport à l'ensemble d'entraînement, cela s'appelle un shift de distribution. Les réseaux de neurones sont sensibles aux shifts de distribution, ce qui peut entraîner des changements significatifs dans leur performance.

La deuxième source d'incertitude vient du système de mesure, qui a un effet direct sur la corrélation entre les échantillons et les cibles correspondantes. Le système de mesure génère des informations x_i et y_i qui décrivent ω_i mais peuvent ne pas contenir suffisamment d'informations pour apprendre une correspondance directe de x_i à y_i. Cela signifie qu'il pourrait y avoir des informations du monde réel très différentes ω_i et ω_j (par exemple, ville et forêt) donnant lieu à des mesures correspondantes très similaires x_i et x_j (par exemple, température) ou des cibles correspondantes similaires y_i et y_j (par exemple une erreur sur les labels les indiquant tout deux comme forêt).

Facteur II : Erreur et bruit dans les systèmes de mesure
Les mesures elles-mêmes peuvent être une source d'incertitude sur la prédiction du réseau de neurones. Cela peut être causé par des informations limitées dans les mesures, comme par exemple la résolution de l'image, ou par le non-mesurage d'informations fausses ou insuffisamment disponibles. De plus, cela peut être causé par du bruit, par exemple le bruit du capteur, par le mouvement ou par le stress mécanique conduisant à des mesures imprécises. De plus, une étiquetage erroné est également une source d'incertitude qui peut être considérée comme une erreur et un bruit dans le système de mesure. Il est considéré comme bruit sur les labels et affecte le modèle en réduisant la confiance sur la vraie prédiction de classe pendant l'entraînement.

Conception et entraînement du réseau de neurones profond

La conception d'un DNN couvre la modélisation explicite du réseau de neurones et son processus d'entraînement. Les hypothèses sur la structure du problème induites par la conception et l'entraînement du réseau de neurones sont appelées biais inductif. Nous résumons toutes les décisions du modélisateur sur la structure du réseau (par exemple, le nombre de paramètres, les couches, les fonctions d'activation, etc.) et le processus d'entraînement (par exemple, algorithme d'optimisation, régularisation, augmentation, etc.) dans une configuration de structure que l'on notera s. La structure du réseau représente notre troisième facteur d'incertitude.

Facteur III : Erreurs dans la structure du modèle
La structure d'un réseau de neurones a un effet direct sur sa performance et donc également sur l'incertitude de sa prédiction. Par exemple, le nombre de paramètres affecte la capacité de mémorisation, ce qui peut conduire à une sous- ou sur-adaptation sur les données d'entraînement. En ce qui concerne l'incertitude dans les réseaux de neurones, on sait que les réseaux profonds ont tendance à être trop confiants dans leur sortie softmax, ce qui se traduit par une probabilité surestimée pour la classe dont la probabilité softmax de sortie est la plus grande.

Pour une structure de réseau donnée s et un ensemble de données d'entraînement D_n, l'entraînement d'un réseau de neurones est un processus stochastique et donc le réseau de neurones résultant f_θ est basé sur une variable aléatoire,

$$\theta | D_n, s \sim p_{\theta|D_n,s}.$$

Le processus est stochastique en raison de décisions aléatoires comme l'ordre des données, l'initialisation aléatoire ou la régularisation aléatoire comme l'augmentation ou le dropout. Le paysage de la fonction de perte d'un réseau de neurones est hautement non linéaire et l'aléa dans le processus d'entraînement conduit en général à différents optima locaux θ^* résultant en différents modèles. De plus, des paramètres

tels que la taille du batch, le taux d'apprentissage et le nombre d'epochs affectent l'entraînement et donnent lieu à différents modèles. Selon la tâche sous-jacente, ces modèles peuvent différer de manière significative dans leurs prédictions pour des échantillons uniques, voire conduire à une différence dans la performance globale du modèle. Cette sensibilité au processus d'entraînement conduit directement au quatrième facteur d'incertitudes dans les prédictions du réseau de neurones.

Facteur IV : Erreurs dans la procédure d'entraînement
Le processus d'entraînement d'un réseau de neurones comprend de nombreux paramètres qui doivent être définis (taille du batch, optimiseur, taux d'apprentissage, critères d'arrêt, régularisation, etc.) et des décisions stochastiques à l'intérieur du processus d'entraînement (génération de lots et initialisation des poids) ont lieu. Toutes ces décisions affectent l'optimum local et il est donc très improbable que deux processus d'entraînement donnent la même paramétrisation du modèle. Un ensemble de données d'entraînement qui souffre de déséquilibre ou de faible couverture de régions individuelles dans la distribution des données introduit également des incertitudes sur les paramètres appris du réseau, comme déjà décrit dans l'acquisition des données. Cela peut être atténué en appliquant une augmentation pour augmenter la variété ou en équilibrant l'impact de classes ou de régions individuelles sur la fonction de perte. Comme le processus d'entraînement est basé sur l'ensemble de données d'entraînement donné D_n, des erreurs dans le processus d'acquisition des données (par exemple, bruit d'étiquetage) peuvent entraîner des erreurs dans le processus d'entraînement.

Inférence

L'inférence décrit la prédiction d'une sortie y^* pour un nouvel échantillon de données x^* par le réseau de neurones. À ce stade, le réseau est formé pour une tâche spécifique. Ainsi, les échantillons qui ne sont pas tirés du domaine de $P(X, Y)$ causent des erreurs et sont donc également une source d'incertitude.

Facteur V : Erreurs causées par des données inconnues
En particulier dans les tâches de classification, un réseau de neurones qui est formé sur des échantillons dérivés d'un monde W_1 peut également être capable de traiter des échantillons dérivés d'un monde complètement différent W_2. C'est par exemple le cas, lorsqu'un réseau formé sur des images de chats et de chiens reçoit un échantillon montrant un oiseau. Ici, la source d'incertitude ne réside pas dans le processus d'acquisition des données, puisque nous supposons qu'un monde contient uniquement des entrées réalisables pour une tâche de prédiction. Même si le résultat pratique peut être égal à trop de bruit sur un capteur ou à une défaillance totale d'un capteur, les données considérées ici représentent un échantillon valide, mais pour une tâche ou un domaine différent.

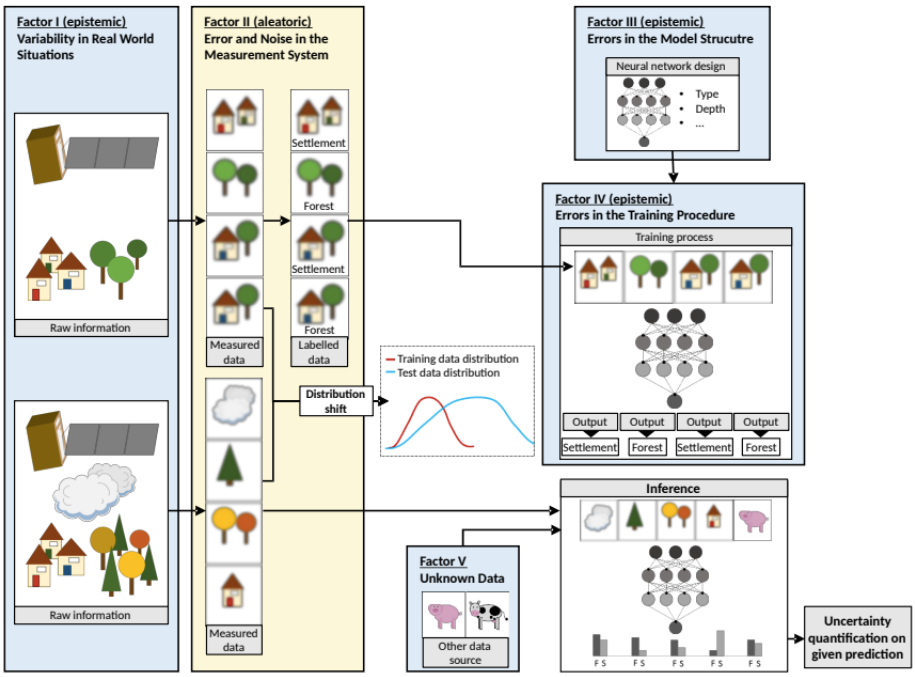

Figure 2: L'illustration montre les différentes étapes d'un pipeline de réseau de neurones, basé sur l'exemple d'observation de la terre pour la classification de la couverture terrestre (ici les zones habitées et la forêt) à partir d'images optiques. Les différents facteurs affectant l'incertitude prédictive sont mis en évidence dans les encadrés. Le facteur I est représenté par des changements d'environnements comme des arbres couverts de nuages, différents types et couleurs d'arbres. Le facteur II est illustré par des mesures insuffisantes, qui ne peuvent pas être utilisées directement pour différencier les zones habitées de la forêt, et par le bruit dans les labels. En pratique, la résolution de ces images peut être basse, ce qui relèverait également du facteur II. Les facteurs III et IV représentent les incertitudes causées par la structure du réseau et le processus d'entraînement stochastique, respectivement. En revanche, le facteur V est représenté par l'alimentation du réseau formé avec des types d'images inconnus, à savoir des vaches et des cochons.
[2]

3 Métriques

Nous nous intéresserons tout d'abord aux métriques usuelles utilisées pour évaluer les méthodes d'estimation de l'incertitude, la justesse et la fiabilité de ces prédictions d'incertitude n'étant en général pas automatiquement garanties.

Nous nous concentrerons sur l'étude des métriques mesurant la performance des méthodes sur un dataset de test dans le cadre de la classification multiclasse. Ces métriques sont utilisées pour évaluer la capacité d'une méthode donnée à prédire quels échantillons sont correctement classifiés ou non. Les approches les plus courantes sont l'AUROC (Area Under the Receiver Operating Curve), l'AUPR (Area Under the Precision Recall Curve), le FPR at 95% TPR et l'E-AURC (Excess Area Under the Risk Coverage Curve). Pour toutes ces métriques, le dataset est divisé en deux sous-ensembles, les échantillons correctement classifiés que nous appellerons succès de prédiction (ou prédictions réussies) et les échantillons incorrectement classifiés que nous appellerons erreurs de prédiction(ou prédictions erronées).

Nous considérerons les succès de prédiction comme la classe positive et les erreurs de prédiction comme la classe négative (sauf exception dans le cas de l'AUPR Error où ce sera l'inverse). On peut construire un classifieur prédisant le succès ou l'échec d'une prédiction en choisissant un seuil tel que tous les échantillons ayant un score d'incertitude supérieur à ce seuil sont classifiés comme des erreurs de prédiction et tous les échantillons dont le score est inférieur au seuil sont classifiés comme succès de prédiction.

Nous commencerons par quelques définitions :

- **Vrai positif (TP)** : Un échantillon positif et classifié comme positif.
- **Vrai négatif (TN)** : Un échantillon négatif et classifié comme négatif.
- **Faux positif (FP), Erreur de type I** : Un échantillon négatif et classifié comme positif.
- **Faux négatif (FN), Erreur de type II** : Un échantillon positif et classifié comme négatif.
- **Sensibilité, Recall, ou taux de vrai positif (TPR)** :

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = 1 - \mathrm{FNR} \tag{1}$$

- **Spécificité, sélectivité ou taux de vrai négatif (TNR)** :

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{FPR} \tag{2}$$

- **Précision ou valeur prédictive positive (VPP)** :

$$\mathrm{VPP} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = 1 - \mathrm{FDR} \tag{3}$$

3.1 AUROC

La courbe ROC illustre le taux de vrais positifs (TPR) en fonction du taux de faux positifs (FPR) pour différents seuils. Le TPR quantifie le nombre correct de résultats positifs par rapport à tous les échantillons positifs, tandis que le FPR indique le nombre incorrect de résultats positifs par rapport à tous les échantillons négatifs.

L'espace ROC, présenté dans la figure 3, est défini avec FPR et TPR comme axes x et y, illustrant les compromis entre les vrais positifs et les faux positifs. Une prédiction parfaite serait représentée par le point (0,1) dans cet espace, signifiant 100% de sensibilité et 100% de spécificité. Un classifieur aléatoire, comme un tirage au sort, produirait un point le long de la diagonale. Les points au-dessus de cette diagonale sont de bons classifieurs, tandis que ceux en dessous sont moins performants.

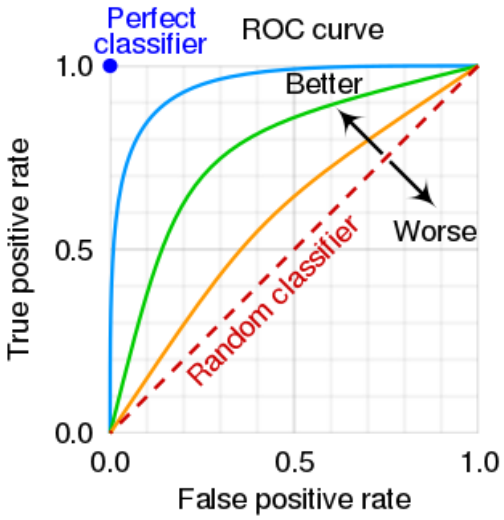

Figure 3: L'espace ROC pour différentes qualités de classifieur [7]

L'AUROC est l'aire sous la courbe ROC. Elle fournit une mesure agrégée de la performance du modèle sur tous les seuils possibles. De manière intuitive, l'AUROC représente la probabilité qu'une prédiction positive ait un score plus élevé qu'une prédiction négative.

L'une des limitations de l'AUROC est que lorsque l'on a affaire à des données très déséquilibrées, la classe majoritaire domine fortement la mesure. L'AUROC, en évaluant la performance sur tous les seuils possibles, prend en compte les taux de vrais positifs et de faux positifs. Cependant, dans un contexte déséquilibré, même un grand nombre de faux négatifs peut avoir un impact relativement faible sur le taux global de faux négatifs, du fait de la grande quantité de vrais positifs. Cela peut conduire à une valeur élevée de l'AUROC même si la prédiction pour la classe minoritaire est médiocre.

3.2 AUPR

La courbe Precision-Recall est une mesure utile du succès de la prédiction lorsque les classes sont très déséquilibrées. La précision est une mesure de la pertinence des résultats, tandis que le recall est une mesure du nombre de résultats réellement pertinents qui sont renvoyés, comme illustré dans la figure 4.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

La courbe PR montre le compromis entre la précision et le recall pour différents seuils. L'intuition derrière l'aire sous la courbe PR (AUPR) est qu'elle capture la capacité d'un modèle à distinguer entre les classes positives et négatives dans un contexte déséquilibré. Plus cette aire est proche de 1, plus le modèle est performant.

Une grande surface sous la courbe représente à la fois un recall élevé et une précision élevée. Des scores élevés pour les deux indiquent que le classifieur renvoie des résultats précis (grande précision), tout en renvoyant la majorité de tous les résultats positifs (grand recall).

L'AUPR est particulièrement utile lorsque les données sont déséquilibrées car, contrairement à l'AUROC qui considère à la fois les vrais positifs et les vrais négatifs, l'AUPR se concentre principalement sur les vrais positifs.

Nous distinguerons deux types de courbes PR : la courbe PR Success qui traite les succès de prédiction comme la classe positive et le courbe PR Error qui considère les erreurs de prédictions comme la classe positive. Nous nous intéresserons principalement à la métrique d'AUPR Error. Cela est pertinent car, par exemple, les réseaux de neurones de classification d'image atteignent souvent des taux de précision bien supérieurs à 50%. Dans une telle situation, les erreurs de prédiction deviennent la classe minoritaire.

Dans ce contexte déséquilibré, une métrique telle que l'AUROC est souvent trompeuse, car fortement influencée par la classe majoritaire (dans notre cas, les prédictions réussies). En revanche, l'AUPR met davantage l'accent sur la perfor-

mance du modèle sur la classe minoritaire, qui est précisément celle qui nous intéresse ici : les échecs de prédiction.

Ainsi, l'AUPR Error permet une évaluation plus ciblée de la capacité du modèle à quantifier son incertitude. Si la méthode de calcul du score d'incertitude d'un modèle est de bonne qualité, cela signifie que le modèle est capable de reconnaître de manière fiable quand il est susceptible de faire une erreur. Un AUPR Error élevé indique que le score d'incertitude du modèle est un bon indicateur des instances où il est susceptible de se tromper. Cela peut s'interpréter par le fait que le modèle a plus souvent tendance à faire des erreurs de prédiction lorsque le score d'incertitude κ est élevé.

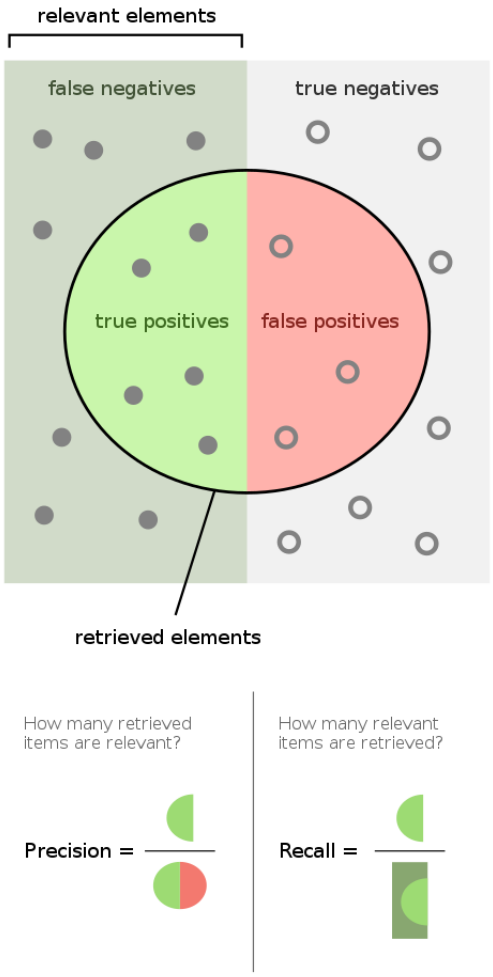

Figure 4: Illustration du rapport Precision-Recall
[8]

3.3 FPR at 95% TPR

Le FPR (taux de faux positifs) à 95% de TPR (taux de vrais positifs) correspond au taux de faux positifs lorsque le seuil de classification du score κ est choisi tel

que 95% des exemples dont dont la prédiction est un succès sont classifiés comme certains.

Lorsque nous évaluons la performance d'un modèle, en particulier dans le contexte de l'estimation de l'incertitude, nous voulons souvent comprendre comment le modèle se comporte dans des situations limites. Le FPR à 95% de TPR est une métrique nous permettant d'appréhender cela en nous donnant une perspective sur les performances du modèle lorsqu'il est à un niveau très élevé de TPR.

Cependant, obtenir un TPR élevé peut souvent s'accompagner d'un coût, à savoir l'acceptation d'un taux plus élevé de faux positifs. Par conséquent, en fixant le TPR à 95%, nous voulons voir combien d'exemples négatifs le modèle classifie incorrectement comme positifs. C'est précisément ce que mesure le FPR.

Cette métrique est particulièrement pertinente pour l'estimation de l'incertitude car elle nous montre comment le modèle gère le compromis entre capturer autant de vrais positifs que possible (c'est-à-dire avoir une grande certitude sur ses prédictions correctes) tout en évitant de mal classer les négatifs. Si le FPR est élevé à 95% de TPR, cela peut indiquer que le score d'incertitude κ du modèle n'est pas fiable dans les situations limites.

3.4 AURC et E-AURC

Les métriques d'AURC et d'E-AURC [3] s'inscrivent directement dans le cadre de la prédiction sélective, en utilisant des concepts propres à ce domaine comme les notions de classifieur sélectif, de risque sélectif (selective risk) et de couverture (coverage). Nous allons tout d'abord définir ces termes puis nous introduirons la courbe RC (Risk Coverage). La métrique AURC d'un score d'incertitude κ est définie comme l'aire sous la courbe RC d'un classifieur sélectif induit par κ. La métrique E-AURC est une normalisation de l'AURC où nous soustrayons l'AURC du meilleur κ a posteriori. L'avantage de cette normalisation est qu'elle permet des comparaisons pertinentes entre différents problèmes.

La performance d'un classifieur sélectif est quantifiée en utilisant la couverture et le risque. La couverture, définie par $\phi(f, g) = \mathbb{E}_P[g(x)]$, est la masse de probabilité de la région non rejetée dans X. Le risque sélectif de (f, g) est

$$R(f, g) = \frac{\mathbb{E}_P[l(f(x), y)g(x)]}{\phi(f, g)}$$

Ces deux mesures peuvent être évaluées empiriquement sur n'importe quel ensemble annoté fini D_n (pas nécessairement l'ensemble d'entraînement) de manière directe. Ainsi, le risque sélectif empirique est défini par :

$$\hat{r}(f, g|D_n) = \frac{\sum_{i=1}^{n} l(f(x_i), y_i)g(x_i)}{\hat{\phi}(f, g|D_n)}$$

où $\hat{\phi}$ est la couverture empirique définie $\hat{\phi}(f, g|D_n) = \frac{1}{n}\sum_{i=1}^{n} g(x_i)$. Le profil de performance global d'une famille de classifieurs sélectifs (optimisés pour différents taux de couverture) peut être mesuré en utilisant la courbe risque-couverture (RC-curve), définie comme le risque sélectif en fonction de la couverture.

Étant donné un classifieur f et une fonction de score de confiance κ définie pour f, nous définissons une mesure de performance empirique pour κ en utilisant un ensemble indépendant V_m de m points annotés (par exemple l'ensemble de test). La mesure de performance est définie en termes du classifieur sélectif suivant (f, g) (où f est notre classifieur donné), et les fonctions de sélection g sont définies comme un seuil sur les valeurs κ, $g_\theta(x|\kappa, f) = 1[\kappa(x, \hat{y}_f(x)|f) > \theta]$.

Soit Θ l'ensemble de toutes les valeurs κ des points dans V_m, $\Theta = \{\kappa(x, \hat{y}_f(x)|f) : (x, y) \in V_m\}$. La performance de κ est définie comme l'aire sous l'aire sous la courbe risque-couverture (AURC) de la paire (f, g) calculée sur V_m,

$$AURC(\kappa, f|V_m) = \frac{1}{m} \sum_{\theta \in \Theta} \hat{r}(f, g_\theta|V_m)$$

Intuitivement, un meilleur κ induira un meilleur classifieur sélectif qui aura tendance à rejeter en premier les points qui sont mal classifiés par f. En conséquence, la courbe RC associée diminuera plus rapidement (avec une couverture décroissante) et l'AURC sera plus petit.

Une fonction de score de confiance optimale a posteriori pour f, notée κ^*, donnera la courbe de risque de couverture optimale. Cette fonction optimale classe tous les points mal classifiés (par f) plus bas que tous les points correctement classifiés. Le risque sélectif associé à κ^* est donc nul à tous les taux de couverture inférieurs à $1 - \hat{r}(f|V_m)$. La raison est que la fonction optimale rejette tous les points mal classifiés à de tels taux.

Puisque l'AURC de toutes les courbes RC pour f induites par n'importe quelle fonction de score de confiance sera plus grand que l'AURC de κ^*, nous normalisons par AURC(κ^*) pour obtenir une mesure de performance sans unité. Pour calculer l'AURC de κ^*, nous calculons l'intégrale discrète de $\hat{r}$ (par rapport à κ^*) du niveau de couverture de $1 - \hat{r}(f|V_m)$ (0 erreurs) à 1 ($m\hat{r}$ erreurs). Ainsi,

$$\text{AURC}(\kappa^*, f|V_m) = \frac{1}{m} \sum_{i=1}^{rm} \frac{i}{m(1-\hat{r}) + i}$$

Nous approximons ensuite cette quantité en utilisant l'intégrale suivante :

$$\text{AURC}(\kappa^*, f|V_m) \approx \int_0^{\hat{r}} \frac{x}{1-\hat{r}+x}\, dx = x - (1-\hat{r}) \ln(1-\hat{r}+x) \Big|_0^{\hat{r}} = \hat{r} + (1-\hat{r}) \ln(1-\hat{r})$$

Pour conclure cette section, nous définissons l'Excess-AURC (E-AURC) comme E-AURC($\kappa, f|V_m$) = AURC($\kappa, f|V_m$) − AURC($\kappa^*, f|V_m$). E-AURC est une mesure sans unité dans [0, 1], et le κ optimal aura E-AURC = 0.

4 Méthodes d'estimation de l'incertitude

4.1 Présentation générale des différente stratégies

Comme décrit dans la section II, plusieurs facteurs peuvent causer de l'incertitude dans le modèle et les données, et affecter la prédiction d'un DNN. Cette variété de sources d'incertitude rend l'exclusion complète des incertitudes dans un réseau de neurones impossible pour presque toutes les applications. En particulier dans les applications pratiques utilisant des données du monde réel, les données d'entraînement ne sont qu'un sous-ensemble de toutes les données possibles, ce qui signifie qu'un décalage entre le domaine du DNN et le domaine de données réel inconnu est souvent inévitable. De plus, une représentation exacte de l'incertitude d'une prédiction DNN n'est également pas possible à calculer, car les différentes incertitudes ne peuvent en général pas être modélisées avec précision et sont le plus souvent même inconnues.

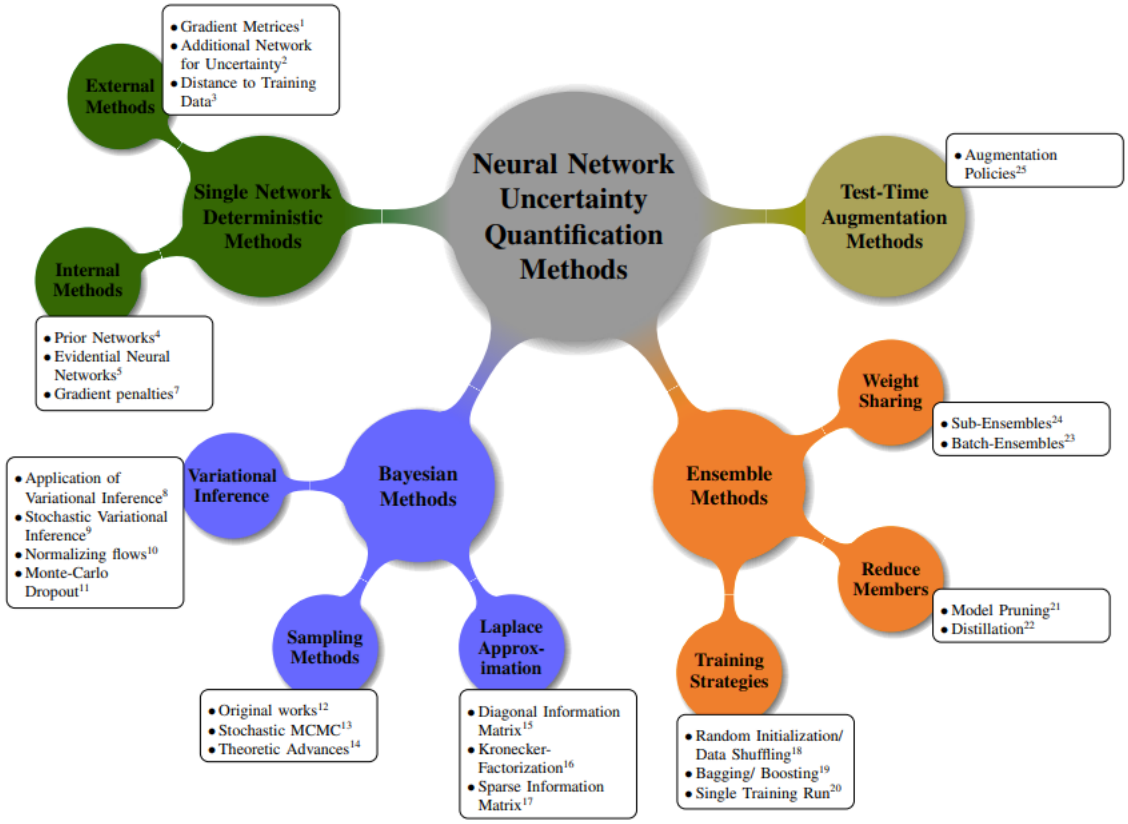

Figure 5: Présentation des familles de méthodes d'estimation de l'incertitude [2]

En général, les méthodes d'estimation de l'incertitude peuvent être divisées en quatre types différents basés sur le nombre (unique ou multiple) et la nature (déterministe ou stochastique) des DNN utilisés.

- Les méthodes de réseau déterministe unique produisent la prédiction à partir d'une seule forward pass dans un réseau déterministe. La quantification de

l'incertitude est soit calculée en utilisant des méthodes supplémentaires, soit directement prédite par le réseau.

- Les méthodes bayésiennes couvrent tous les types de DNN stochastiques, c'est-à-dire des DNN où deux forward pass du même échantillon conduisent généralement à des résultats différents.
- Les méthodes d'ensemble combinent les prédictions de plusieurs réseaux déterministes différents lors de l'inférence.
- Les méthodes d'augmentation au moment du test donnent la prédiction basée sur un seul réseau déterministe mais augmentent les données d'entrée au moment du test afin de générer plusieurs prédictions qui sont utilisées pour évaluer la certitude de la prédiction.

	Single Deterministic Networks	**Bayesian Methods**	**Ensemble Methods**	**Test-Time Data Augmentation**
Description	Approaches that receive an uncertainty quantification on a prediction of a deterministic neural network.	Model parameters are explicitly modeled as random variables. For a single forward pass the parameters are sampled from this distribution. Therefore, the prediction is stochastic and each prediction is based on different model weights.	The predictions of several models are combined into one prediction. A variety among the single models is crucial.	The prediction and uncertainty quantification at inference is based on several predictions resulting from different augmentations of the original input sample.
Description of Model Uncertainties	No	Yes	No	No
Need changes on existing networks	Depends on method	Yes	Yes (retrain several times)	No
Sensitivity to initialization and parameters of training process	High (in general)	Low (Usage of uninformative priors possible)	Low	Low
Number of networks trained	1	1	Several	1
Computational effort during training	Low	High	High	Low
Memory consumption during training	Low	Low	High	Low
Number of inputs per prediction	1	1	1	Several
Forward passes per prediction	1	Several	Several	Several
Evaluated modes	Single	Single	Multiple	Single
Computational effort during inference	Low (One forward pass, possibly some minor additional effort for uncertainty quantification)	High (sampling is either needed for explicit approach or for the approximation of intractable formulas)	High (Several models need to be evaluated)	High (Several augmentations and forward passes are performed)
Memory Consumption - Inference	Low	Low	High	Low

Figure 6: Tableau récapitulatif des familles de méthodes
[2]

Nous allons étudier dans cette section quatre méthodes état de l'art : Softmax (notre baseline qui est une méthode de réseau déterministe unique), MCDropout (la méthode bayésienne de référence), Deep Ensemble (une méthode ensembliste) et ConfidNet (une méthode de réseau déterministe unique qui utilise un DNN auxiliaire pour prédire l'incertitude).

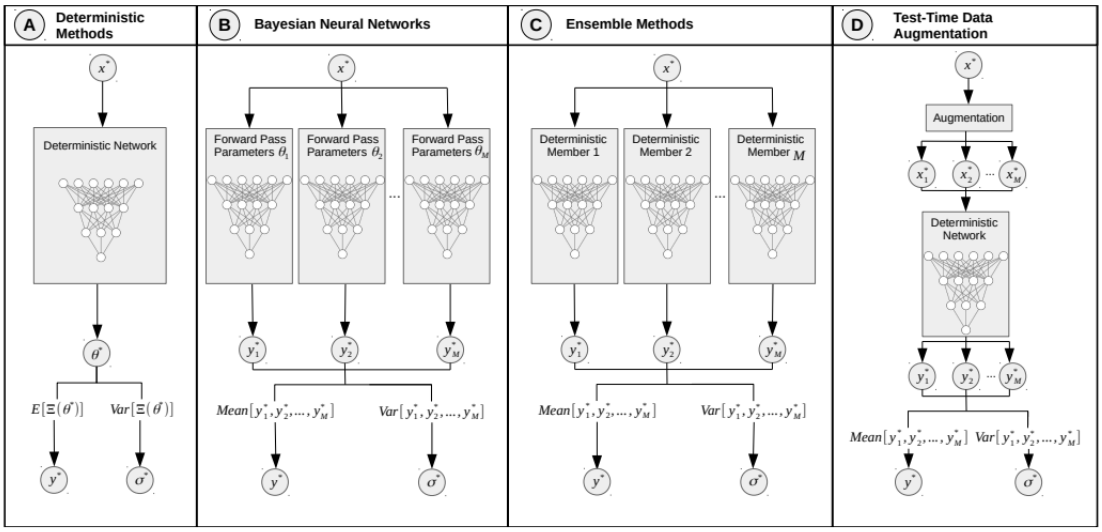

Figure 7: Schéma explicatif des familles de méthodes d'estimation de l'incertitude [2]

4.2 MCP (Baseline)

Nous présenterons tout d'abord la méthode utilisée traditionellement comme baseline dans l'estimation de l'incertitude : la MCP (Maximum Class Probability) qui est la probabilité softamx de sortie du réseau de la classe prédite. Un problème bien connu d'utilisation de la MCP comme méthode d'estimation de l'incertitude est la sur-confiance systématique dans la confiance accordée aux prédictions même dans les cas où le réseau se trompe complètement. Ce problème étant surtout dommageable dans un contexte d'étude de la calibration du réseau, nous le laisserons de côté et étudierons la capacité de la MCP à prédire les erreurs de prédiction. L'article de Dan Henrycks et Kevin Gimpel "A baseline for detecting misclassified and out-of-distribution examples in neural networks" présente des résultats qui justifie l'utilisation de cette méthode de score comme baseline [5].

Leurs expérimentations se basent sur trois datasets : MNIST, CIFAR-10 et CIFAR-100. MNIST est un ensemble de données de chiffres manuscrits, composé de 60 000 exemples d'entraînement et de 10 000 exemples de test. Pendant ce temps, CIFAR-10 dispose d'images en couleur appartenant à 10 classes différentes, avec 50 000 exemples d'entraînement et 10 000 exemples de test. CIFAR-100 est plus difficile, car il comporte 100 classes différentes avec 50 000 exemples d'entraînement et 10 000 exemples de test. Dans le tableau ci dessous, nous voyons que les exemples correctement classifiés et incorrectement classifiés sont suffisamment distincts et permettent donc une discrimination fiable. On peut noter que les métriques se dégradent lorsque l'erreur de test augmente.

Dataset	AUROC /Base	AUPR Succ/Base	AUPR Err/Base	Pred. Prob Wrong(mean)	Test Set Error
MNIST	97/50	100/98	48/1.7	86	1.69
CIFAR-10	93/50	100/95	43/5	80	4.96
CIFAR-100	87/50	96/79	62/21	66	20.7

Figure 8: La MCP permet de discriminer entre les exemples du jeu de test correctement et incorrectement classifiés. "Pred. Prob Wrong (mean)" est la probabilité des MCP pour les exemples mal classifiés, mettant en évidence sa limite en tant que mesure directe de confiance. La MCP est comparée à la méthode de score aléatoire appelée ici Base.

[5]

4.3 Deep Ensemble

La méthode Deep Ensemble [6] est une méthode relativement simple composée de trois étapes:

1. Utiliser une *proper scoring rule* comme critère d'entraînement.
2. Utiliser l'entraînement adversarial pour lisser les distributions prédictives.
3. Former un ensemble.

Nous utiliserons comme notations dans cette section M pour le nombre de réseaux de neurones dans l'ensemble et $\{\theta_m\}_{m=1}^{M}$ pour les paramètres de l'ensemble.

Règles de scoring appropriées : Les règles de scoring évaluent la qualité de l'incertitude prédictive. Une *scoring rule*, attribue un score numérique à une distribution prédictive $p(y|x, D_n)$. L'objectif est de récompenser les prédictions mieux calibrées par rapport aux moins bonnes. Une règle de scoring est dite appropriée (*proper scoring rule*) si le score pour une prédiction correcte est toujours supérieur ou égal au score pour une prédiction incorrecte, avec égalité seulement lorsque la prédiction est parfaite. Il s'avère que de nombreuses fonctions de loss utilisée pour entrîner les réseaux de neurones sont des *proper scoring rule*. Par exemple dans le cas de la classification multi-classes, la loss d'entropie croisée softmax couramment utilisée est équivalente à la vraisemblance logarithmique et constitue une *proper scoring rule*.

Entraînement adversarial pour lisser les distributions prédictives : Un exemple adversarial est un échantillon d'entrée qui est très proche d'un échantillon d'entraînement original, mais qui trompe le réseau pour donner une prédiction incorrecte. L'entraînement adversarial vise à rendre le réseau de neurones robuste à de tels exemples en ajoutant des perturbations lors de l'entraînement. Ce faisant, il aide également à lisser la distribution prédictive du réseau, rendant les prédictions plus fiables sur des échantillons inconnus. La méthode utilisée pour générer des exemples adversarials est la *fast gradient sign method* : étant donné une entrée x

avec pour cible y, et une fonction de perte $l(f(x), y)$, la *fast gradient sign method* génère un exemple adversarial par

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x l(\theta, x, y))$$

où ϵ est une petite valeur telle que la norme maximale de la perturbation est limitée. Intuitivement, la perturbation adversarial crée un nouvel exemple d'entraînement en ajoutant une perturbation dans une direction où la fonction de loss augmente. En supposant que ϵ soit suffisamment petit, ces exemples adversariaux peuvent être utilisés pour augmenter l'ensemble d'entraînement original en traitant (x', y) comme des exemples d'entraînement supplémentaires. Cette procédure, appelée entraînement adversarial, améliore la robustesse du classifieur.

Ensembles (entraînement et prédiction) : Un ensemble est une combinaison de plusieurs modèles d'apprentissage automatique (dans ce cas, des réseaux de neurones) qui sont utilisés ensemble pour produire une prédiction. La méthode d'ensemble permet d'améliorer la performance globale et de réduire le risque d'overfitting. Dans cette méthode, l'approches utilisée est la randomisation plutôt que le bagging. La méthode Deep Ensemble utilise toutes les données d'entraînement pour entraîner chaque réseau car les DNN ont tendance à mieux fonctionner avec beaucoup de données. En pratique, l'initialisation aléatoire des paramètres du NN, combinée à un mélange aléatoire des points de données, est, d'après les auteurs, suffisante pour obtenir une bonne diversité de réseaux.

Algorithm 1 Pseudocode de la procédure d'entraînement

Laisser chaque réseau de neurones paramétrer une distribution sur les sorties, i.e., $p_\theta(y|x)$.
Initialiser $\theta_1, \theta_2, ..., \theta_M$ aléatoirement.
for $m = 1$ à M **do**
  Échantillonner le point de données n_m aléatoirement pour chaque réseau.
  Générer un exemple adversarial utilisant $x'_{nm} = x_{nm} + \epsilon \text{sign} \nabla_{x_{nm}} l(\theta_m, x_{nm}, y_{nm})$.
  Minimiser $l(\theta_m, x_{nm}, y_{nm}) + l(\theta_m, x'_{nm}, y_{nm})$ par rapport à θ_m.
end for

Dans l'article de recherche présentant cette méthode, les auteurs n'évaluent pas leur méthode dans le cadre de la prédiction d'échecs ou de succès et donc n'utilisent pas les métriques présentées à la section précédente. De plus les résultats présentés dans l'article sont de notre point de vue peu intéressants : les auteurs comparent leurs résultats à MC Dropout en termes de nombre de réseaux et non en terme de coût computationnel et mémoriel alors qu'il est beaucoup moins coûteux de faire une forward pass pour MC Dropout que d'entraîner un réseau supplémentaire pour Deep Ensemble. Nous ne présenterons donc que nos résultats à partir de notre implémentation de cette méthode à la section suivante.

4.4 Monte Carlo Dropout

Nous allons dans cette section présenter la méthode bayésienne Monte Carlo Dropout [5]. Nous commencerons par introduire la notion de distribution prédictive puis nous présenterons le fonctionnement de la méthode MCDropout.

Distributions prédictives : Considérons une distribution prédictive, notée par $p(y \mid x, D_n)$, où x est la cible, y est l'entrée, et D_n est l'ensemble d'exemples d'entraînement, défini comme $D_n = (x_i, y_i)_{i=1}^n$. Une fois cette distribution obtenue, il est possible d'analyser sa variance pour déceler l'incertitude.

MC Dropout : L'une des approches pour apprendre une distribution prédictive nécessite d'apprendre une distribution sur les fonctions ou, de manière équivalente, une distribution sur les paramètres (c'est-à-dire la distribution postérieure paramétrique $p(\Theta \mid D_n)$). La technique de Monte Carlo Dropout, proposée par Gal et Ghahramani en 2016, offre une méthode évolutive pour apprendre cette distribution.

MC Dropout fonctionne en désactivant aléatoirement des neurones dans un réseau de neurones, ce qui régularise le réseau. Chaque configuration de dropout correspond à un échantillon différent de la distribution postérieure approximée $q(\Theta \mid D_n)$. Cette équivalence peut être exprimée comme $\Theta_t \sim q(\Theta \mid D_n)$, où Θ_t est une configuration de dropout ou, de manière équivalente, une simulation échantillonnée depuis la distribution postérieure approximée.

En échantillonnant depuis cette distribution postérieure approximée, on peut réaliser une intégration de Monte Carlo de la vraisemblance du modèle, révélant ainsi la distribution prédictive comme suit :

$$p(y \mid x) \approx \int_\Omega p(y \mid x, \Theta) q(\Theta \mid D) d\Theta \tag{4}$$

$$\overset{\text{MC}}{\approx} \frac{1}{T} \sum_{t=1}^{T} p\left(y \mid x, \Theta_t\right), \text{ où } \Theta_t \sim q(\Theta \mid D) \tag{5}$$

Pour simplifier, on peut supposer que la vraisemblance est distribuée selon une gaussienne, exprimée comme :

$$p(y \mid x, \Theta) = \mathcal{N}\left(f(x, \Theta), s^2(x, \Theta)\right)$$

où la fonction gaussienne est spécifiée par les paramètres de moyenne $f(x, \Theta)$ et de variance $s^2(x, \Theta)$ issus des simulations de Monte Carlo Dropout.

La figure suivante illustre le MCDropout. Chaque configuration de dropout produit une sortie différente en éteignant aléatoirement des neurones (cercles gris) et en les allumant (cercles noirs) à chaque propagation avant. De multiples forward pass avec différentes configurations de dropout fournissent une distribution prédictive sur la moyenne $p(f(x, \theta))$. Le nombre de forward pass à travers les données devrait être évalué quantitativement, mais une plage de 30 à 100 est appropriée à considérer d'après les auteurs.

De même que pour la méthode Deep Ensemble, les auteurs n'évaluent pas leur méthode dans le cadre de la prédiction d'échecs ou de succès et donc n'utilisent pas les métriques présentées à la section précédente. Nous ne présenterons donc que nos résultats à partir de notre implémentation de cette méthode à la section suivante.

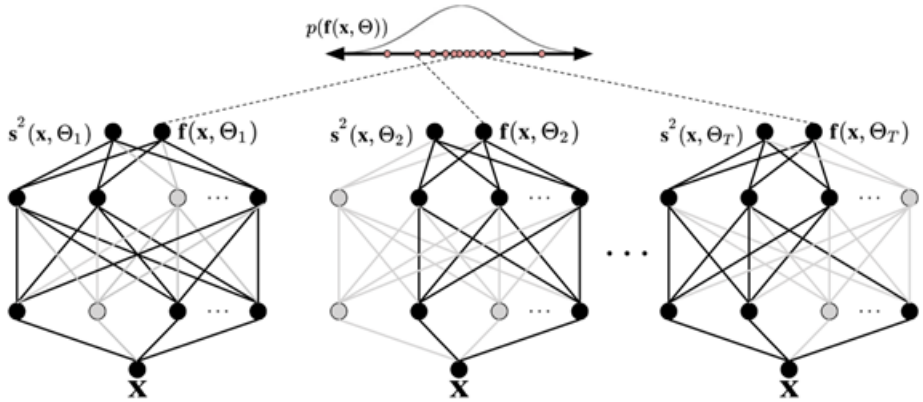

Figure 9: Schéma du fonctionnement de MCDropout

4.5 ConfidNet

La méthode ConfidNet [1] est une méthode spécifiquement développée pour la prédiction d'erreurs et de succès par C. Corbière et al.

La première contribution de cette méthode est de de considérer la True Class Probability (TCP) comme critère de confiance adapté à la prédiction des erreurs :

$$\text{TCP} : R^d \times Y \to R \quad (x, y^*) \to P(Y = y^*|w, x)$$

Avec la TCP, les propriétés suivantes sont vérifiées :

- $\text{TCP}(x, y^*) > \frac{1}{2} \Rightarrow \hat{y} = y^*$: l'exemple est correctement classifié par le modèle,
- $\text{TCP}(x, y^*) < \frac{1}{K} \Rightarrow \hat{y} \neq y^*$: l'exemple est mal classifié par le modèle.

Au sein de l'intervalle $\left[\frac{1}{K}, \frac{1}{2}\right]$, il n'y a pas de garantie théorique que les prédictions correctes et incorrectes ne se chevauchent pas en termes de TCP. Cependant, avec les réseaux de neurones profonds, nous observons que la zone de chevauchement réelle est extrêmement petite en pratique.

Les auteurs introduisent également une variante normalisée du critère de confiance TCP, qui consiste à calculer le rapport entre TCP et MCP :

$$\text{TCPr}(x, y^*) = \frac{P(Y = y^*|w, x)}{P(Y = \hat{y}|w, x)}$$

Le critère TCPr présente des garanties théoriques plus fortes que TCP, puisque, par conception, les prédictions correctes recevront la valeur 1, tandis que les erreurs seront dans l'intervalle $[0, 1[$. D'un autre côté, l'apprentissage de ce critère peut être plus difficile car toutes les prédictions correctes doivent correspondre à une unique valeur scalaire.

Apprentissage de la TCP avec un réseau de neurones auxiliaire: Utiliser la TCP comme score de confiance serait très utile pour prédire les erreurs. Cependant, la vraie classe y^* n'est évidemment pas disponible lors de l'estimation de la confiance sur les échantillons de test. Ainsi, nous proposons d'utiliser un réseau

auxiliaire pour apprendre la TCP : $c^*(x, y^*) = P(Y = y^*|w, x)$. Ce réseau appelé ConfidNet, a pour paramètres θ et produit une prédiction de confiance $\hat{c}(x, \theta)$. L'objectif lors de l'entraînement est d'obtenir θ tel que $\hat{c}(x, \theta)$ soit proche de $c^*(x, y^*)$.

Durant l'entraînement initial de classification, le modèle M apprend à extraire des caractéristiques de plus en plus complexes. Pour bénéficier de ces représentations riches, le réseau ConfidNet utilisera ces couches intermédiaires comme inputs. La loss L^2 est utilisée pour entraîner ConfidNet, car nous souhaitons régresser un score entre 0 et 1.

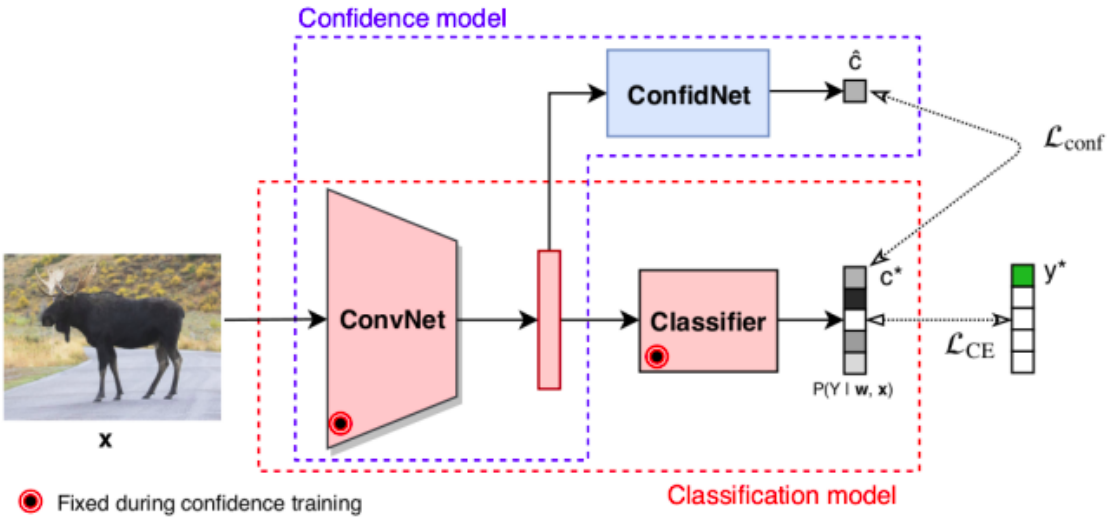

Figure 10: Schéma du fonctionnement de ConfidNet
[1]

Les expérimentations dans l'article utilisent plusieurs datasets de complexité différentes : MNIST et SVHN offrent des images simples et de petite taille (28×28) représentant des chiffres (10 classes), CIFAR-10 et CIFAR-100 présentent des tâches de reconnaissance d'objets plus complexes sur des images de faible résolution. Les architectures des réseaux de classification varient de petits réseaux convolutionnels pour MNIST et SVHN à la plus grande architecture VGG-16 pour les jeux CIFAR. Un perceptron multicouche (MLP) avec une couche cachée a également été ajouté pour MNIST. Le réseau de prédiction de confiance, ConfidNet, est relié à l'avant-dernière couche du réseau de classification et se compose de 5 couches denses.

La méthode ConfidNet est comparée aux méthodes : MCP comme baseline, Trust Score, et Monte-Carlo Dropout. Nous constatons que cette approche surpasse la méthode baseline dans tous les scénarios, notamment sur les petits modèles/jeux de données, et obtient en moyenne de meilleurs résultats que les méthodes MC-Dropout et TrustScore. Cela confirme que la TCP est un critère de confiance adéquat pour la prédiction des échecs et que cette méthode est capable de l'apprendre.

Dataset	Model	FPR-95%-TPR	AUPR-Error	AUPR-Success	AUC
MNIST MLP	Baseline (MCP) [17]	14.87	37.70	99.94	97.13
	MCDropout [10]	15.15	38.22	99.94	97.15
	TrustScore [20]	12.31	52.18	99.95	97.52
	ConfidNet (Ours)	**11.79**	**57.37**	**99.95**	**97.83**
MNIST Small ConvNet	Baseline (MCP) [17]	5.56	35.05	99.99	98.63
	MCDropout [10]	5.26	38.50	99.99	98.65
	TrustScore [20]	10.00	35.88	99.98	98.20
	ConfidNet (Ours)	**3.33**	**45.89**	**99.99**	**98.82**
SVHN Small ConvNet	Baseline (MCP) [17]	31.28	48.18	99.54	93.20
	MCDropout [10]	36.60	43.87	99.52	92.85
	TrustScore [20]	34.74	43.32	99.48	92.16
	ConfidNet (Ours)	**28.58**	**50.72**	**99.55**	**93.44**
CIFAR-10 VGG16	Baseline (MCP) [17]	47.50	45.36	99.19	91.53
	MCDropout [10]	49.02	46.40	**99.27**	92.08
	TrustScore [20]	55.70	38.10	98.76	88.47
	ConfidNet (Ours)	**44.94**	**49.94**	99.24	**92.12**
CIFAR-100 VGG16	Baseline (MCP) [17]	67.86	71.99	92.49	85.67
	MCDropout [10]	64.68	72.59	**92.96**	86.09
	TrustScore [20]	71.74	66.82	91.58	84.17
	ConfidNet (Ours)	**62.96**	**73.68**	92.68	**86.28**
CamVid SegNet	Baseline (MCP) [17]	63.87	48.53	96.37	84.42
	MCDropout [10]	62.95	49.35	96.40	84.58
	TrustScore [20]		20.42	92.72	68.33
	ConfidNet (Ours)	**61.52**	**50.51**	**96.58**	**85.02**

Figure 11: Résultats expérimentaux de ConfidNet [1]

5 Résultats

5.1 Méthodologie

Nous avons utilisé le language python et la biblithèque tensorflow/keras pour l'implémentation des méthodes d'estimation de l'incertitude. Nous avons essentiellement utilisé les GPUs A6000 du laboratoire ISIR pour toutes nos expérimentations.

Nous nous placerons dans un cadre de classification d'images et nos résultats se baseront sur le dataset CIFAR10 qui contient 60000 images 32×32 réparties en 10 classes, avec 50000 images dans l'ensemble d'entraînement et 10000 images dans l'ensemble de test.

Nous utiliserons un ResNet 44 comme architecture pour notre réseau de neurones, ce qui nous donne une accuracy de 92%.

5.2 Expérimentations sur les métriques

Nous avons observé que les métriques usuelles d'incertitudes, telles que celles présentées en Section 3 présentaient des variations lorsque que l'accuracy des modèles sous-jacent changeait. Afin d'illustrer ce phénomène, nous avons simulé des datasets jouets dans $\mathbb{R}^2$ avec des niveaux croissants de séparations des deux classes, puis nos avons entraîné sur chacun un classifieur binaire XGBoost.

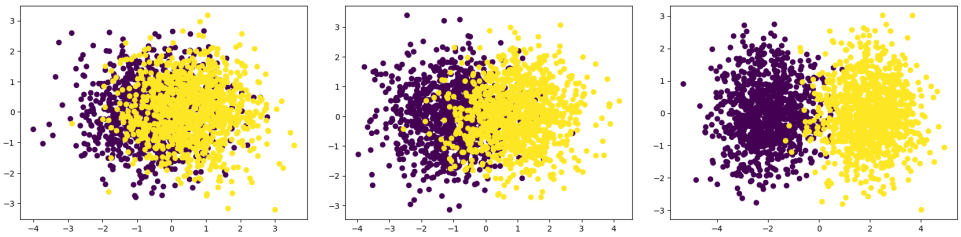

Figure 12: Datasets jouets dans $\mathbb{R}^2$

Ces modèles de classification binaire présentent donc une accuracy croissante par rapport au coefficient de séparation des classes. Nous avons utilisé la méthode baseline MCP (Section 4.2) ce qui nous a permis de mesurer l'incertitude du classifieur entraîné à mesure que les données sont de plus en plus faciles à classifier. Nous avons affiché dans la figure 13 ci-dessous nos résultats.

Nous avons observé dans ce cadre très simplifié une forte corrélation des variations de ces métriques avec les variations en accuracy. Par rapport à notre objectif de comparaison des performances des méthodes d'estimation de l'incertitude, nous avons conclu que seules les méthodes ne changeant pas l'accuracy du modèle de base étaient pertinentes à utiliser car elles seules pouvaient être comparées entre elles. Par exemple, les méthodes bayésiennes et ensemblistes supposant par définition une modification du modèle et donc de l'accuracy, ne peuvent pas être comparées à d'autres méthodes, ce qui de notre point de vue les disqualifie. En revanche, nous considérons les méthodes de réseaux auxiliaires, qui ne modifie pas l'accuracy, comme les méthodes les plus pertinentes à étudier, car pouvant être objectivement évaluées.

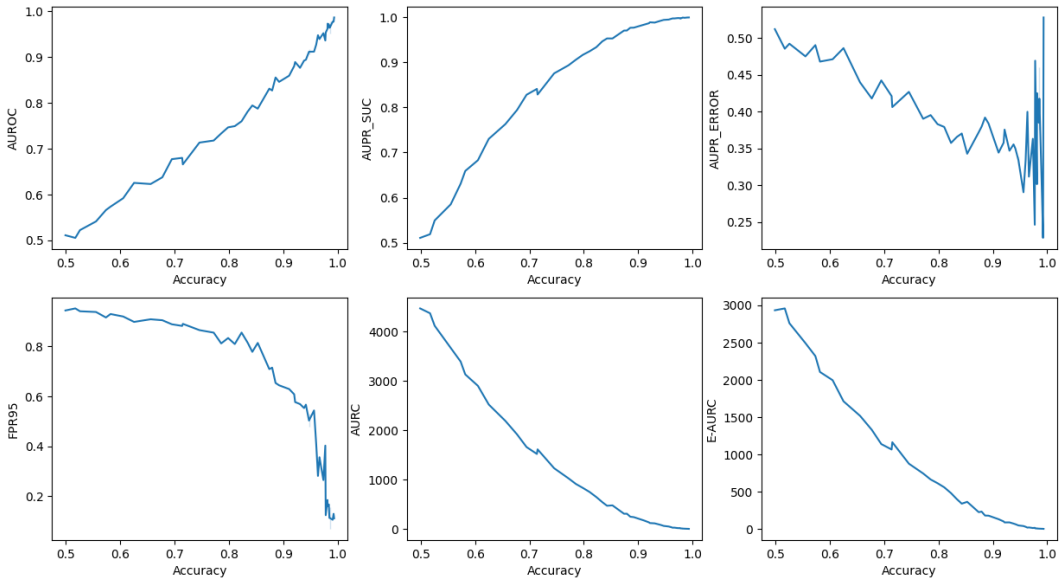

Figure 13: Variations des métriques usuelles en fonction de l'accuracy

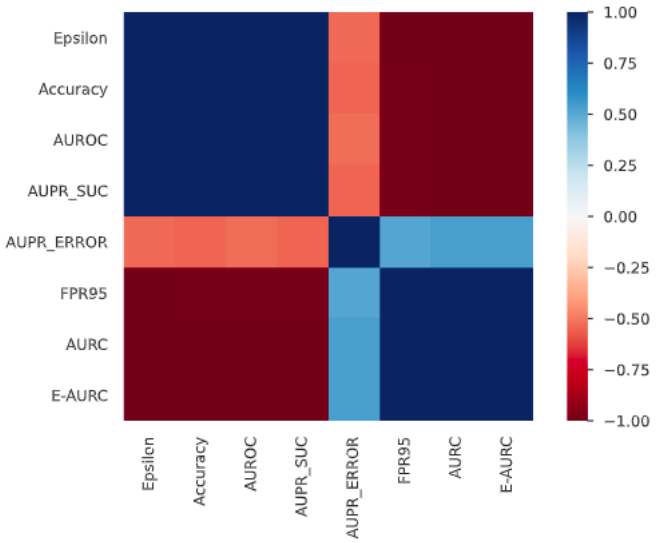

Figure 14: Heatmap des corrélations entre les métriques et l'accuracy du modèle

5.3 Expérimentations sur les méthodes

Nous mesurerons les performances des méthodes implémentées avec les métriques AUROC, AUPR-Sucess, AUPR-Error et FPR at 95% TPR. Nous avons recontré un problème dans l'implémentation des métriques AURC et E-AURC tels que décrits dans la papier que nous n'avons pas pu résoudre à temps pour le rapport.

5.3.1 MCP (Baseline)

Nous avons affiché ci-dessous les distributions des scores pour la méthode baseline (en vert les succès de prédiction et en rouge les échecs). On observe que les succès sont extrêmement majoritairement prédits avec une grande certitude alors qu'un grand nombre d'échecs ont un score élevés. Cela est dû au problème mentionné de sur-confiance de la probabilité softmax de sortie.

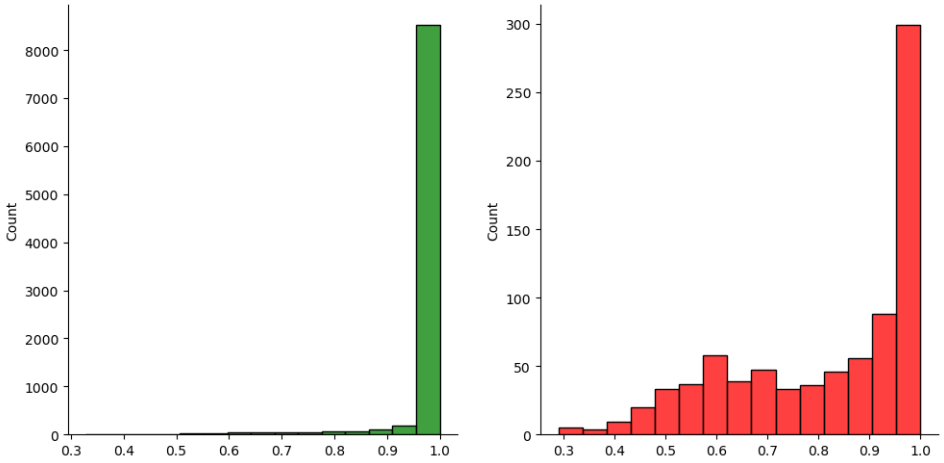

Figure 15: Résultats expérimentaux de MCP

Les résultats de la méthode MCP sont présentés dans la table ci-dessous.

Métrique	**Valeur**
AUROC	0.908
AUPR Succcess	0.989
AUPR Error	0.475
FPR at 95% TPR	0.488

Table 1: Métriques MCP

5.3.2 Deep Ensemble

Nous avons implémenté la méthode Deep Ensemble en suivant les recommandations données par les auteurs de la méthode, à savoir cinq réseaux avec des initialisations différentes et des données d'entraînement mélangées de manière différente à chaque fois. Il est toutefois à noter que cette méthode présente une accuracy proche de 94% et n'est donc pas comparable à la baseline MCP ne comprenant qu'un seul réseau.

Nous avons testé plusieurs manières d'obtenir un score de confiance à partir des prédictions d'un ensemble, notamment prendre la moyenne des MCP (Mean), la variance (Var) ou l'écart-type (STD) des MCP, la différence entre la MCP et la probabilité softmax de la classe ayant la seconde probabilité softmax de sortie la plus élevée (DIF_1), ou bien la différence entre la MCP et la probabilité softmax de la classe ayant la seconde probabilité softmax de sortie la plus élevée chacune normalisée par son écart type (DIF_2). Les résultats que nous avons obtenus sont présentés dans la table ci-dessous. Nous remarquons que la méthode qui moyenne les MCP des réseaux de l'ensemble est la plus performante.

Méthode	**AUROC**	**AUPR Success**	**AUPR Error**	**FPR at 95% TPR**
MEAN	0.931	0.995	0.436	0.432
VAR	0.923	0.994	0.392	0.513
STD	0.924	0.995	0.389	0.505
DIF_1	0.93	0.995	0.403	0.443
DIF_2	0.926	0.995	0.379	0.469

Table 2: Résultats des méthodes DeepEnsemble

5.3.3 Monte Carlo Dropout

Nous avons implémenté la méthode MCDropout en ajoutant un dropout à la fin de chaque connexion résiduelle de notre ResNet. Nous avons choisi de faire 100 forward pass comme le recommande les auteurs de la méthode. Nous avons de plus essayé plusieurs probabilité de dropout, nous présenterons ici les résultats pour une probabilité de 20% et de 50%. Il faut noter de même que pour la section précédente que la méthode MCDropout a légèrement augmenté les performances du réseau donnant une accuracy d'environ 93% ce qui empêche toute comparaison avec la méthode baseline MCP ou Deep Ensemble.

Méthode	**AUROC**	**AUPR Success**	**AUPR Error**	**FPR at 95% TPR**
MEAN	**0.933**	**0.994**	**0.497**	**0.378**
VAR	0.929	**0.994**	0.456	0.469
STD	0.932	**0.994**	0.469	0.386
DIF_1	**0.933**	**0.994**	0.491	0.379
DIF_2	**0.933**	**0.994**	0.483	0.379

Table 3: Résultats pour MCDropout à 20%

Méthode	AUROC	AUPR Success	AUPR Error	FPR at 95% TPR
MEAN	**0.926**	**0.994**	**0.473**	0.436
VAR	0.914	0.993	0.398	0.554
STD	0.922	0.993	0.439	0.501
DIF_1	0.925	**0.994**	0.456	0.433
DIF_2	0.924	0.993	0.437	**0.432**

Table 4: Résultats pour MCDropout à 50%

Nous remarquons encore une fois que la méthode de moyenne des MCP sur les forward pass dans le réseau est la plus efficace.

5.3.4 ConfidNet

Nous avons implémenté la méthode ConfidNet de manières légèrement différente de celle proposée par les auteurs. En effet, l'architecture du réseau auxiliaire et le processus d'entraînement qu'ils utilisent présentaient des résultats moins bon que ceux dans leur article de recherche lorsque nous les avons testé sur notre ResNet44. Nous avons donc faits différents essais et nous avons décidé d'utiliser un réseau un peu plus profond, avec un nombre de neurones décroissants par couche et nous avons utilisé l'optimiseur ADAM.

Nous avons de plus rencontré une difficulté lors l'implémentation de ConfidNet. Notre réseau de classification d'images a en effet une accuracy sur les données de train de plus de 99% et une accuracy de 92% en test. Ce shift de distribution pose problème car seuls les échecs de prédiction ont une TCP basse ce qui ne représente dans notre cas qu'environ 300 exemples sur les données d'entraînement.

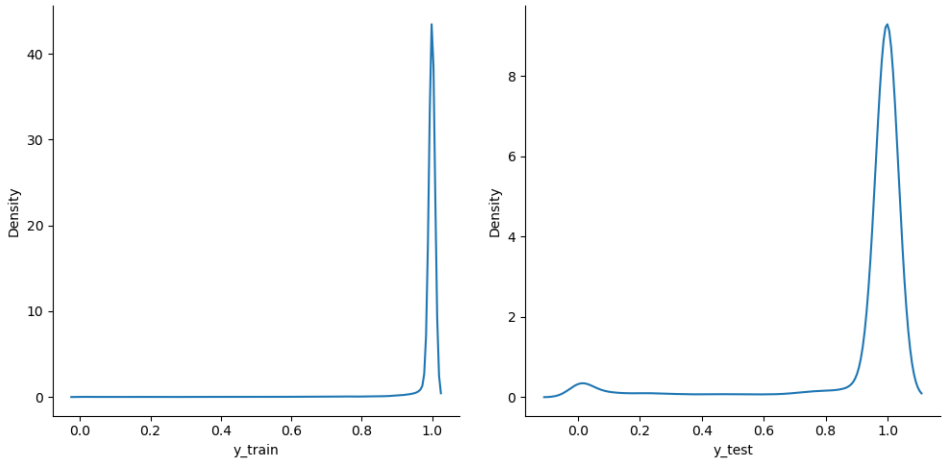

Figure 16: Distributions de la TCP entre ensemble d'entraînement à gauche et ensemble de test à droite

Comme nous pouvons le voir dans les graphiques ci-dessus représentant les distributions des TCP pour les données d'entraînement et de test, le shift de distribution est marqué par la présence dans les données de train d'une bosse vers 0 qui représente

les échecs de prédiction. Cela a pour conséquence dans notre cas que le réseau auxiliaire a du mal à apprendre à prédire les échecs. Nous avons tenté de nombreuses méthodes d'augmentation des données mais aucune ne présentaient de résultats intéressants, ce qui s'explique par le très faible nombre d'échecs de prédictions en entraînement ne permettant pas d'avoir une diversité suffisante d'information pour généraliser à l'ensemble de test.

Les résultats ci-dessous sont donc ceux de la méthode sans augmentation des données.

	AUROC	**AUPR Success**	**AUPR Error**	**FPR at 95% TPR**
MCP	**0.908**	**0.989**	**0.475**	**0.488**
ConfidNet	0.896	0.988	0.444	0.526

Table 5: Comparaison des métriques entre MCP et ConfidNet

Nous remarquons dans notre implémentation que la baseline MCP est meilleure que ConfidNet dans toutes les métriques. Cela peut s'expliquer par le réseau de classification sous-jacent utilisé qui nous donne une accuracy différente par rapport au réseau que les auteurs de la méthode ConfidNet ont utilisé dans leurs expérimentations.

5.4 Méthode Proposée

Nous avons choisi comme piste de recherche privilégiée les réseaux auxiliaires car ils présentent l'avantage de ne pas modifier la structure du classifieur sous-jacent et donc son accuracy ce qui le rend comparable à la baseline MCP ainsi qu'aux autres méthodes ne modifiant pas l'accuracy, comme vu dans la Section 5.2.

Nous proposons comme solution d'ajouter un paramètre de température sur la fonction softmax de sortie, ce qui lissera les distributions de la TCP rendant le dataset moins déséquilibré et donc plus facile à entraîner, et donc ce qui réduira le shift de distribution entre entraînement et test.

Le softmax est une fonction couramment utilisée en apprentissage automatique, en particulier dans les modèles de classification. Son rôle principal est de transformer un vecteur de scores, souvent appelé logits, en un vecteur de probabilités. Le paramètre de température vient moduler le niveau de confiance du softmax.

Formule générale La formule standard du softmax pour une classe i donnée parmi N classes est :

$$p(i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

où z_i est le logit pour la classe i.

Lorsqu'on introduit un paramètre de température T, cette formule devient :

$$p(i) = \frac{e^{\frac{z_i}{T}}}{\sum_{j=1}^{N} e^{\frac{z_j}{T}}}$$

Interprétation Le paramètre de température contrôle l'entropie du softmax:

- Lorsque $T \to 0^+$, la probabilité de la classe avec le score le plus élevé tend vers 1, tandis que toutes les autres tendent vers 0.
- À $T = 1$, le softmax est le standard sans modulation.
- Pour $T > 1$, les probabilités se rapprochent d'une distribution uniforme, rendant le modèle plus incertain.

Nous remarquons dans les graphiques ci-dessous que le shift de distribution se réduit lorsque nous augmentons le paramètre de température. Il faut cependant faire attention à ne pas trop l'augmenter car cela mènerait à une perte d'information dans le cadre de l'entraînement d'un modèle auxiliaire tel que ConfidNet.

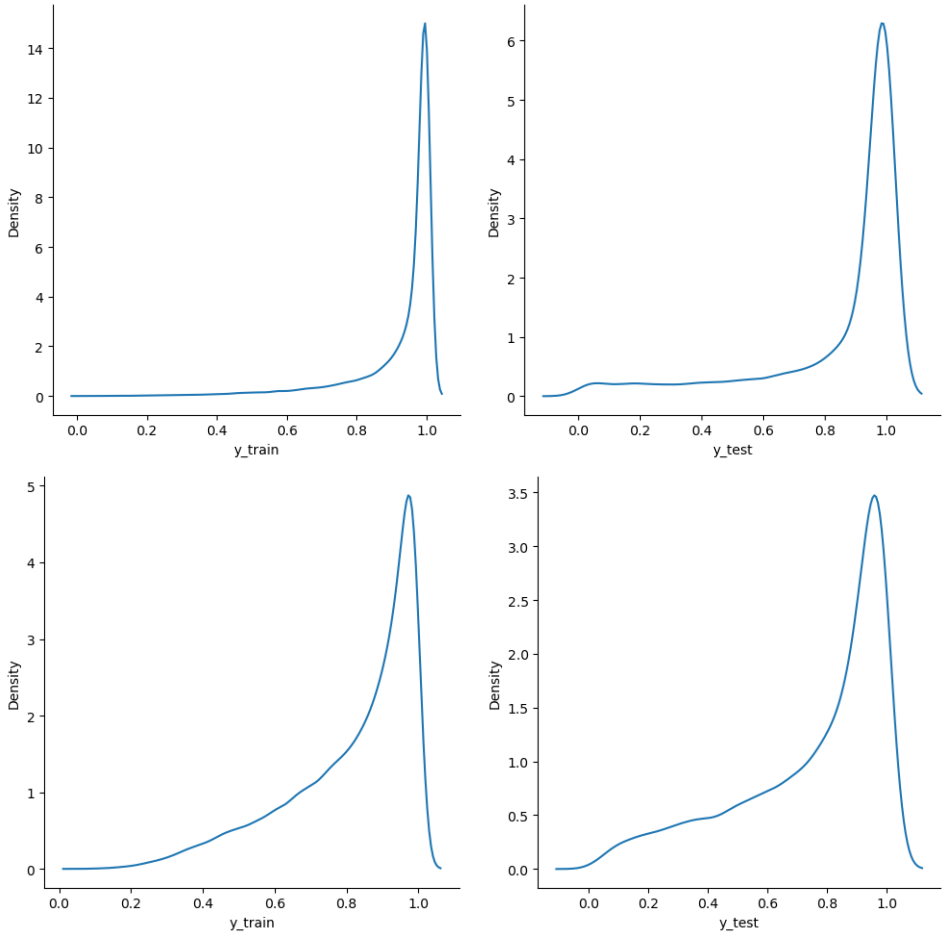

Figure 17: Distributions de la TCP avec un paramètre de température à 3 en haut et à 5 en bas

Ce paramètre de température devient donc un levier d'optimisation pour augmenter les performances d'un ConfidNet. Comme nous pouvons le voir dans la table 6, une l'intervalle de température entre 2 et 3 augmente sensiblement les performances de la méthode, avec un maximum à 2,5. Nous remarquons de plus que lorsque la température dépasse 3, les performances commencent à décliner ce qui était attendu.

T	AUROC	AUPR Success	AUPR Error	FPR
1	0.896	0.988	0.444	0.526
1.25	0.905	0.990	0.453	0.523
1.5	0.914	0.991	0.466	0.509
1.75	0.916	0.992	0.481	0.491
2	0.916	0.992	0.487	0.483
2.25	0.915	0.992	0.480	0.481
2.5	**0.919**	**0.992**	**0.507**	0.480
2.75	**0.919**	**0.992**	0.506	0.470
3	0.917	0.992	0.502	0.465
3.25	0.918	0.992	0.498	**0.464**
4	0.915	0.992	0.490	0.497
5	0.910	0.991	0.475	0.500

Table 6: Résultats en fonction de la valeur de T

Nous avons donc proposé une méthode nouvelle permettant de pallier certaines limitations de ConfidNet, ce qui augmente sensiblement ses performances et nous donne des résulats bien meilleurs que notre baseline MCP, comme nous pouvons le voir dans la table 7.

	AUROC	AUPR Success	AUPR Error	FPR at 95% TPR
MCP	0.908	0.989	0.475	0.488
T = 1	0.896	0.988	0.444	0.526
T = 2.5	**0.919**	**0.992**	**0.507**	**0.480**

Table 7: Comparaison des métriques entre MCP et ConfidNet

6 Conclusion

La motivation principale de l'étude de l'incertitude se trouve dans la classification sélective : l'objectif étant de réduire le taux d'erreur en s'abstenant de prédire en cas de doute, tout en conservant une couverture aussi large que possible. Nous nous sommes placé dans un cadre simplifié où un classificateur de type DNN f est déjà donné, et l'objectif est d'apprendre une fonction de rejet g qui garantira avec une grande probabilité un taux d'erreur souhaité. Nous nous sommes tout d'abord intéressé au paradigme de la classification sélective puis nous présenté les différentes sources d'incertitude qui pouvaient affecter un réseau de neurones. Nous avons ensuite étudié les métriques couramment utilisées pour mesurer la qualité d'un score de confiance, puis les méthodes état de l'art en estimation de l'incertitude : la MCP, Deep Ensemble, MCDropout et enfin ConfidNet. Finalement nous avons décrit les expérimentations que nous avons mené, premièrement sur les métriques, puis sur les méthodes. Les conclusions que nous apportons sur notre travail concernant les métriques sont les suivantes : toutes les métriques étudiées présentent même dans un cadre simplifié des variations en fonction de l'accuracy du classifieur sous-jacent, il n'est donc pas possible de comparer entre elles des méthodes qui modifient la structure et donc l'accuracy du réseau de classification sous-jacent, par exemple les méthodes bayésiennes et ensemblistes. Nous avons donc privilégié d'étudier les méthodes de réseaux auxiliaires comme ConfidNet qui ne souffre pas de ce problème. Dans notre implémentation de cette méthode, nous avons observé des performances inférieures à la baseline et aux résultats avancés par les auteurs, ce qui peut être attribué à une différence de réseau de classification utilisé. Nous avons en effet remarqué que dans notre cas, une très grande accuracy pendant l'entraînement ne permettait pas au réseau auxiliaire d'apprendre correctement sa cible, la TCP. Nous avons donc proposé d'introduire un paramètre de température pour lisser les distributions d'entraînement et de test de la TCP ce qui a produit des gains de performances significatifs, dépassant la méthode ConfidNet originale ainsi que notre baseline, la MCP.

Nous avons exploré beaucoup d'autres pistes de recherche durant ce stage que nous n'avons pas mentionné dans ce rapport, pour cause de résultats peu convaincants ou de manque de temps pour peaufiner certaines méthodes. Nous sommes par exemple en train de travailler sur une catégorie de méthodes nouvelles qui utilisent l'historique d'entraînement du réseau de neurones pour en extraire des statistiques pertinentes pour l'estimation de l'incertitude. Nous n'avons pas pu obtenir de résultat pour cet ensemble de méthodes à temps pour ce rapport mais nous considérons cet axe de recherche particulièrement pertinent et prometteur.

References

[1] Charles Corbière. "Addressing Failure Prediction by Learning Model Confidence". In: (2019).

[2] Jakob Gawlikowski. "A Survey of Uncertainty in Deep Neural Networks". In: (2021). DOI: https://doi.org/10.48550/arXiv.2107.03342.

[3] Yonatan Geifman. “Bias-Reduced Uncertainty Estimation for Deep Neural Classifiers”. In: (2018). DOI: https://doi.org/10.48550/arXiv.1805.08206.

[4] Yonatan Geifman. “Selective Classification for Deep Neural Networks”. In: (2017). DOI: https://doi.org/10.48550/arXiv.1705.08500.

[5] Dan Hendrycks. “A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks”. In: (2016). DOI: https://doi.org/10.48550/arXiv.1610.02136.

[6] Balaji Lakshminarayanan. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”. In: (2016). DOI: https://doi.org/10.48550/arXiv.1612.01474.

[7] *ROC Curve*. URL: https://en.wikipedia.org/wiki/Receiver_operating_characteristic#/media/File:Roc_curve.svg.

[8] *Schéma Precision Recall*. URL: https://en.wikipedia.org/wiki/Precision_and_recall#/media/File:Precisionrecall.svg.