
RAPPORT DE STAGE DE FIN D'ÉTUDES

**Prédiction conforme séquentielle pour les problèmes
d'images**

Manorathan Jeevakan

15 avril au 19 juillet 2024

Tuteur de stage : Monsieur Mohamed HEBIRI

Enseignant référent : Monsieur Cyril NICAUD

Établissement de formation : Université Gustave Eiffel

Entreprise d'accueil : Laboratoire d'Analyse et de Mathématiques
Appliquées - 5, boulevard Descartes, 77420 Champs-sur-Marne

1 Remerciements

Je tiens à remercier toutes les personnes qui ont contribué à la réalisation de mon stage de fin d'études et à la rédaction de mon rapport de stage.

Mes remerciements vont d'abord au LAMA et mon tuteur de stage Monsieur Hebiri pour leur accueil au sein du laboratoire. Je remercie Monsieur Hebiri également pour son implication à la fois dans la volonté de m'offrir un espace de travail confortable, mais aussi lors de la rédaction du rapport en étant à mes côtés tout au long.

Je souhaite remercier l'université pour avoir mis à disposition des stagiaires une salle de travail. Cette dernière offre un environnement propice à la productivité, la collaboration, l'autonomie, et permet d'échanger avec des stagiaires d'autres domaines.

Table des matières

1	Remerciements	2
2	Introduction	4
3	Définitions	6
4	Prédiction conforme – Étude théorique	13
4.1	Contexte	13
4.2	Procédure du vote majoritaire	14
4.3	Autres seuils et borne supérieure	15
4.4	Combiner des ensembles mutuellement indépendants	17
5	Prédiction conforme – Comparaison des différentes méthodes d'agrégation	19
5.1	Union et intersection	20
5.2	Vote majoritaire classique	21
5.3	Vote majoritaire généralisé	23
5.4	Seuil égal au quantile d'ordre α	27
5.5	Moyenne des probabilités issues de softmax	29
6	Conclusion	31
	Références	32

2 Introduction

Dans ce rapport, il est question principalement d'intelligence artificielle (IA), une technologie nouvelle qui ne cesse de faire parler d'elle, de par ses applications dans des domaines transversaux et les prouesses dont elle est capable de nos jours.

Parmi la panoplie d'applications, il y a l'IA générative : elle peut générer des images à partir de texte, générer des textes, étendre artificiellement une image sur les côtés, combler un vide dans une image en s'aidant de ses alentours (ces deux fonctions sont couramment utilisées en retouche photographique). Une application particulière est la capacité à faire des prédictions, plus précisément à faire de la classification d'images, c'est-à-dire donner une étiquette à une image donnée sachant qu'il y eu un apprentissage en amont. Jusque-là l'Homme aussi en est capable. Ce qui change ici, c'est que l'IA peut classifier une image en s'aidant d'une base de données contenant plusieurs milliers voire millions d'images, qu'elles soient des objets de la vie courante ou l'ensemble des espèces de plantes et fleurs existantes sur Terre, par exemple.

Dans notre cas, on s'intéresse à la base d'images ImageNet [1] constituée de 1000 classes contenant chacune près de 1300 images. Dans la Figure 1 se trouve un aperçu de la précision top 1 de ImageNet. Dans ce classement [3], on peut voir que la plus grande précision jamais atteinte pour cette base est de 88,3%.

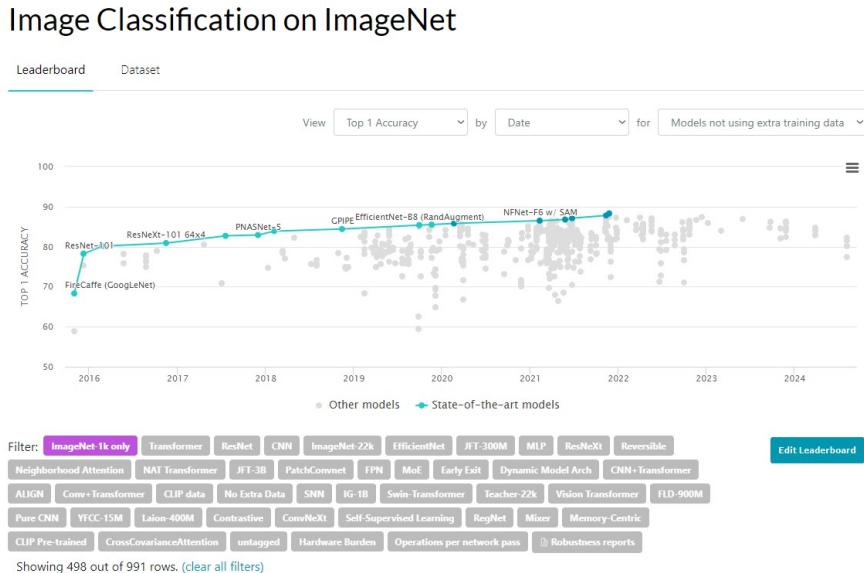


FIGURE 1 – Classification des modèles selon la précision Top 1 sur ImageNet

Dans un premier temps, j'ai importé le modèle ResNet-50 pour en mesurer la performance sur cette base de données, afin de mettre en évidence le classement des classes selon la précision du modèle sur chacune d'entre elles. La pire classe est constituée d'images de maillot, que ce soient des maillots sportifs ou des maillots de bain (voir Figure 2). Étant donné les différences entre ces deux types de maillot, on peut ne pas s'étonner que ce soit la pire classe en précision, avec un score de 30,5% seulement, ceci signifie que, le modèle prédira correctement dans 30,5% des cas si on lui donne une image de maillot.



FIGURE 2 – Quelques images de la classe "maillots", justifiant sa dernière place dans le classement

Pour améliorer la précision Top 1 de nos modèles, nous utiliserons la **prédition conforme**, une méthode initialement développée en 2005 par Vladimir Vovk, Alex Gammerman et Glenn Shafer dans le livre intitulé *Algorithmic Learning in a Random World* [7], qui consiste à utiliser les modèles existants pour agréger les prédictions de plusieurs d'entre eux pour en produire une avec un certain niveau de confiance. Ce stage a pour but de construire des ensembles de prédition et d'étudier différentes manières de les agréger de sorte à obtenir un ensemble final de taille minimale avec un minimum d'ensembles et une probabilité maximale.

Dans toute la suite, nous nous inspirerons principalement d'un article en particulier : celui de Matteo Gasparin et Aaditya Ramdas, intitulé *Merging uncertainty sets via majority vote* [5]. Dans l'article en question, on tente de fusionner différents ensembles de prédition de plusieurs agents en utilisant le vote majoritaire et d'autres méthodes. Ils explorent les nombreuses manières d'améliorer cette méthode avec des poids ou de l'aléatoire en étudiant leur aspect théorique et les accompagnent d'expériences réalisées sur des données réelles.

3 Définitions

Pour aborder le sujet en question convenablement, nous allons avoir besoin de définir quelques notions élémentaires indispensables pour comprendre ce qui va suivre tout au long du rapport.

Définition 3.1. Le **machine learning** consiste à laisser les algorithmes apprendre à reconnaître de nouveaux motifs à partir de données, en vue d'être performant dans une tâche spécifique. Ces données peuvent être de divers formats : images, nombres, mots ou texte, etc. Ainsi en leur donnant de nouvelles données, ces algorithmes pourront faire des prédictions. Les tâches principales qui leur sont confiées sont la régression (prédire une donnée) et la classification (prédir la classe d'appartenance).

Il existe 2 types d'apprentissage automatique :

- l'apprentissage supervisé : on donne un jeu de données d'entraînement où à un certain nombre de catégories pour une observation donnée, qu'on appelle **features**, on associe une catégorie cible, qu'on appelle **label**. Ensuite, si on lui donne de nouvelles données, l'algorithme doit pouvoir prédire une donnée faisant partie de la même catégorie que les labels.

Voici un exemple de problème supervisé de régression : étant donné le prix de l'huile durant ces 10 derniers mois, prédire son prix le mois prochain.

Voici un exemple de problème supervisé de classification : étant donné la longueur et la largeur des pétales d'une fleur, déterminer l'espèce de fleur.

- l'apprentissage non-supervisé : ici, on ne lui donne que des features, il n'y a pas de label. L'algorithme est chargé d'effectuer une tâche ne nécessitant pas la connaissance de labels, comme par exemple le fait de former des groupes à partir d'un nuage de points.

On peut citer quelques exemples de modèles de machine learning : la régression linéaire, la régression logistique, kNN (k voisins les plus proches), les forêts aléatoires, les arbres de décision, PCA, k-means clustering. Cependant il y a un type de modèles qui est de plus en plus utilisé de nos jours et ayant de nombreuses applications : les *réseaux neuronaux*.

Définition 3.2. Un **réseau neuronal** est un ensemble de couches connectées en série, chaque couche contenant un certain nombre de neurones. On passe d'une couche à l'autre via une transformation affine. Le réseau peut contenir autant de couches intermédiaires que nécessaires (par exemple dans la Figure 3, il n'y en a qu'une). Cette construction s'appuie sur le fonctionnement d'un cerveau humain et constitue un modèle de machine learning non négligeable.

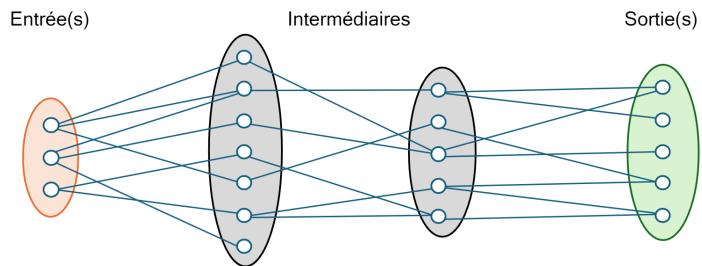


FIGURE 3 – Schéma d'un réseau neuronal

Dans le cadre de la classification d'images, il est nécessaire de savoir tirer des informations telles que le contraste, les bordures, les formes, etc de l'image afin de détecter le type d'objet observé dans un premier temps, pour ensuite affiner la recherche. C'est là que la *convolution* entre en jeu.

Définition 3.3. La **convolution** est une opération qui, à deux fonctions données, associe une fonction. En informatique, notamment en traitement d'images, on utilise la convolution entre deux matrices, la 1ère représentant une image et la 2e représentant un filtre, pour extraire des informations pertinentes sur l'image (voir exemple ci-après).

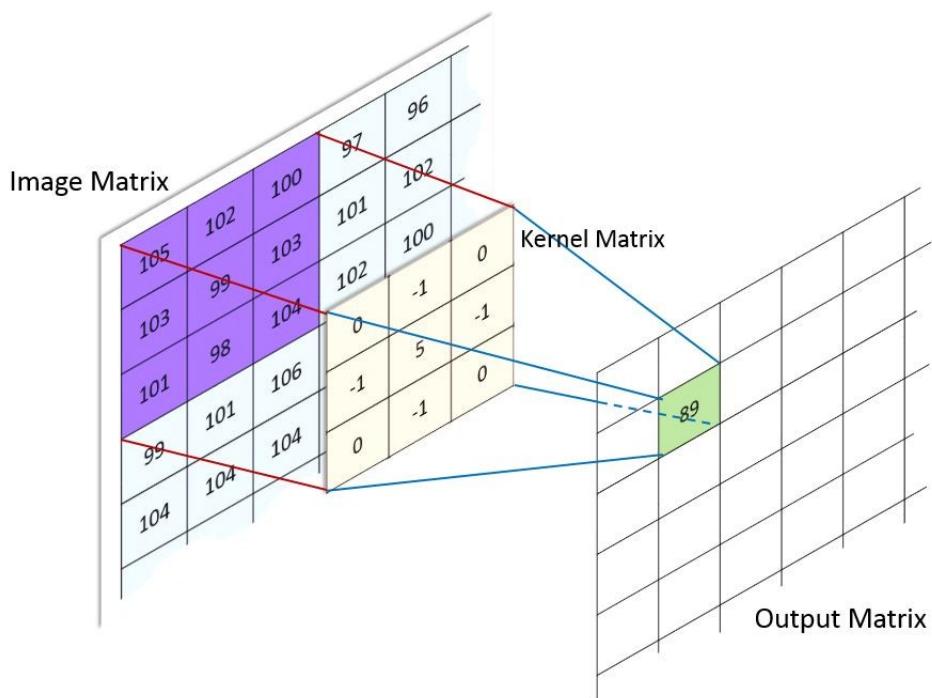


FIGURE 4 – Matrice convolée à une autre

Si on prend l'exemple de la Figure 4, pour obtenir 89, on fait :

$$105 \times 0 + 102 \times (-1) + 100 \times 0 + 103 \times (-1) + 99 \times 5 + \dots + 104 \times 0.$$

La matrice intitulée "Kernel Matrix" est la matrice de ce qu'on appelle un filtre. On l'applique à la matrice "Image Matrix" qui représente une image. La position du coefficient 89 est la même que celle du centre de la matrice du filtre (99). Si le filtre ne couvre pas lui-même entièrement une partie de l'image, alors on ne prendra en compte que les coefficients couverts par le filtre.

Par exemple, pour calculer le coefficient en (1, 2) (1ère ligne, 2e colonne), on procède comme suit :

$$105 \times (-1) + 102 \times 5 + 100 \times (-1) + 103 \times 0 + 99 \times (-1) + 103 \times 0 = 206.$$

L'image ci-dessus étant en 2D, on n'applique qu'un filtre. Si nous avons une image en couleurs, donc en 3D, nous aurons, pour la 3ème dimension, les couches correspondant au Rouge, Vert, Bleu. Donc soit on applique le même filtre pour les 3 couches, soit on applique un filtre différent par couche, autrement dit un filtre en 3D.

En guise d'illustration, prenons une image et appliquons-y différents filtres de flou, un filtre de détection de contours et un filtre de renforcement des contours, puis voyons le résultat.

1er filtre : Filtre de flou 3×3

$$\text{Matrice du filtre : } \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Image Originale



Filtre de Flou 3×3



FIGURE 5 – Filtre de flou 3×3

2e filtre : Filtre de flou 7×7

$$\text{Matrice du filtre : } \frac{1}{49} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$



FIGURE 6 – Filtre de flou 7×7

Remarquez que, dans les Figures 5 et 6, plus la taille de la matrice est grande, plus le flou est important. L'explication est que dans le 2ème cas, nous faisons la moyenne des coefficients sur un plus grand rayon. Donc le coefficient qui en résulte est plus proche de la moyenne globale que celui du flou 3×3 .

3e filtre : Filtre de détection des contours

$$\text{Matrice du filtre : } \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

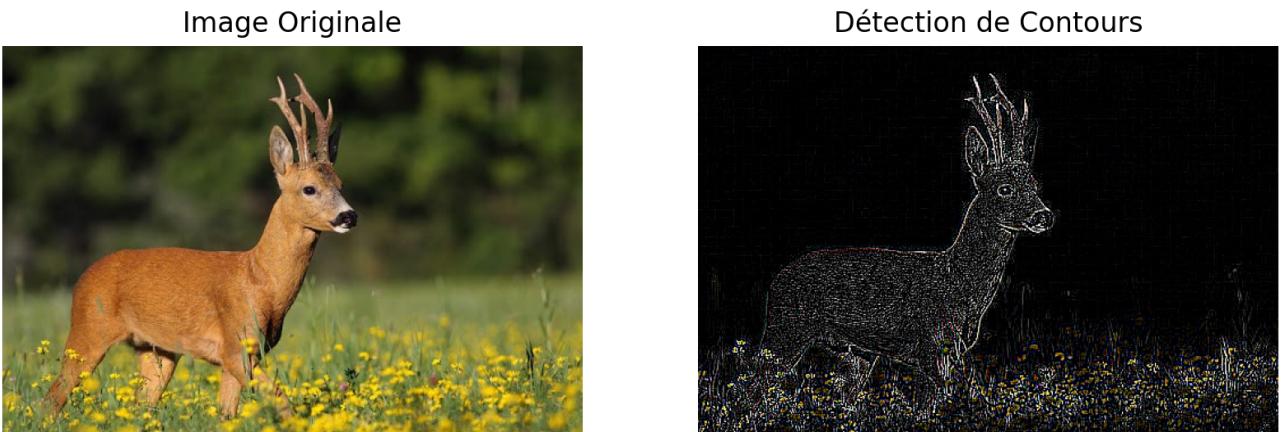


FIGURE 7 – Filtre de détection des contours

Dans la Figure 7, notez que la somme des coefficients de la matrice du filtre est égale à 0. Alors si une zone de taille au moins 3×3 de l'image a une couleur quasiment identique, en convolant cette zone avec le filtre, il est tout à fait normal qu'on obtienne 0. Donc dans le résultat final, si un coefficient est non nul, c'est que la zone autour de celui-ci ne contenait pas qu'une seule nuance de couleur, ce qui veut dire qu'il y a eu nécessairement un contour à cet endroit-là. Ainsi la matrice du filtre a véritablement du sens dans ce contexte.

4e filtre : Filtre de renforcement des contours

$$\text{Matrice du filtre : } \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

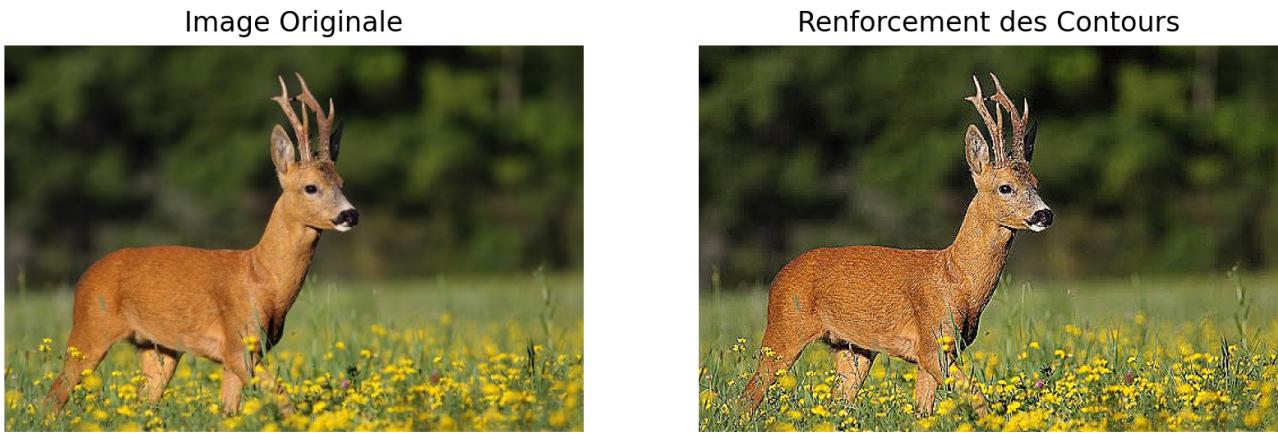


FIGURE 8 – Filtre de renforcement des contours

Dans la Figure 8, on peut voir qu'en prêtant attention à la matrice du filtre en question, en remplaçant le coefficient 5 par un 4, nous avons à nouveau la somme des coefficients nulle. Cette nouvelle matrice peut en effet faire office de matrice de détection de contours, c'est ce qu'on appelle le filtre Laplacien. Ici nous avons un 5. Cela signifie que dans une zone de couleur identique, là où le filtre Laplacien renverrai 0, parce qu'on a une différence de 1 entre 4 et 5, ce filtre renvoie l'identité. Donc là où il y a un contour, ce filtre renvoie alors le contour **en plus** du pixel lui-même, ce qui a pour effet de le mettre en valeur. Pour cette raison, le filtre dans le Figure 8 est bien un filtre de renforcement des contours.

Comme dit précédemment, les couches de convolution permettent d'extraire des informations pratiques utiles pour déterminer avec précision ce qu'on observe actuellement. Pour cela, on génère une image par caractéristique étudiée. Or si l'image initiale a une taille non raisonnable (ce qui est désormais possible avec des appareils photo de haute résolution), on se retrouve à manipuler beaucoup d'images de haute résolution, générant un problème de mémoire. Une façon de résoudre ceci est le *max-pooling*.

Définition 3.4. Le **max-pooling** est une des couches d'un réseau neuronal convolutif, c'est un type de pooling (opération d'agrégation). Il permet de réduire les dimensions d'une image sans altérer les caractéristiques importantes de celle-ci.

Pour expliquer son fonctionnement, on choisit une fenêtre de taille donnée, par exemple 2x2. On la fait passer sur l'image et, à chaque pas, on effectue l'agrégation voulue (prendre le maximum dans notre cas) avec les 4 valeurs présentes dans la fenêtre. En faisant ceci sur l'image, on obtient en sortie une matrice de plus petite taille mais sans information superflue.

Par exemple, si on prend une image 2D de taille 5x5 et qu'on choisit une fenêtre de taille 3x3, on aura à la fin une matrice de taille 3x3 ($3 = 5-3+1$) (voir Figure 9).

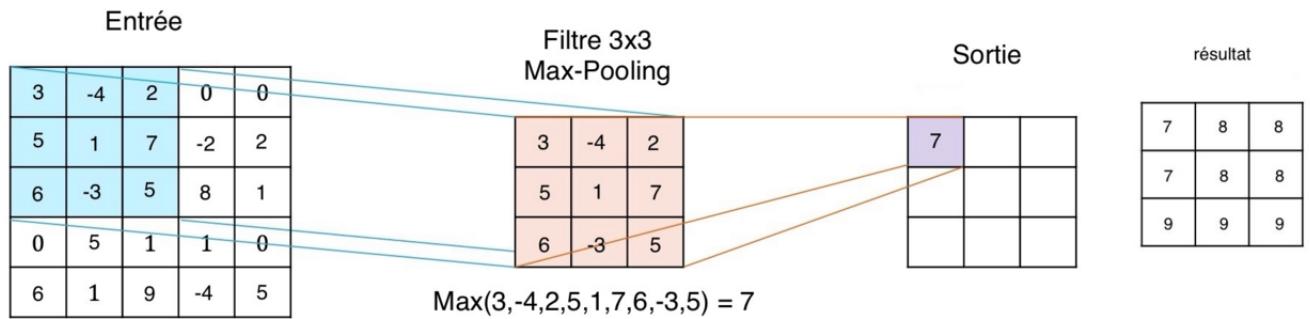


FIGURE 9 – Utilisation du filtre Max-pooling

Définition 3.5. La fonction **softmax** est une fonction qui, pour $N > 0$ un entier naturel, à une liste de N nombres quelconques, associe une liste de N nombres tous compris entre 0 et 1. Mathématiquement parlant, on a :

$$\text{softmax} : \begin{cases} \mathbb{R}^N \longrightarrow [0, 1]^N \\ (x_1, \dots, x_N) \longmapsto \left(\frac{e^{x_1}}{\sum_{i=1}^N e^{x_i}}, \dots, \frac{e^{x_N}}{\sum_{i=1}^N e^{x_i}} \right) \end{cases} .$$

Il faut noter que les coordonnées du vecteur en sortie de la fonction sont tous compris entre 0 et 1 et leur somme donne exactement 1. Cette fonction a donc la propriété de transformer toute liste de nombres en une distribution de probabilité. De ce fait, elle est très utilisée pour faire de la classification en multi-classes ; pour prédire une classe, il suffit de considérer celle ayant la plus grande probabilité.

Définition 3.6. Un **réseau neuronal convolutif** (qu'on notera par la suite CNN pour Convolutional Neural Network en anglais) est un réseau neuronal où les couches sont regroupées par paquets où chaque paquet contient des couches de convolution, de pooling. Dans le cas d'une classification, la dernière couche sera une couche dense avec la fonction d'activation softmax.

La Figure 10 est un exemple de CNN utilisé pour de la classification d'images, elle indique sa composition en couches et les étapes du processus de classification.

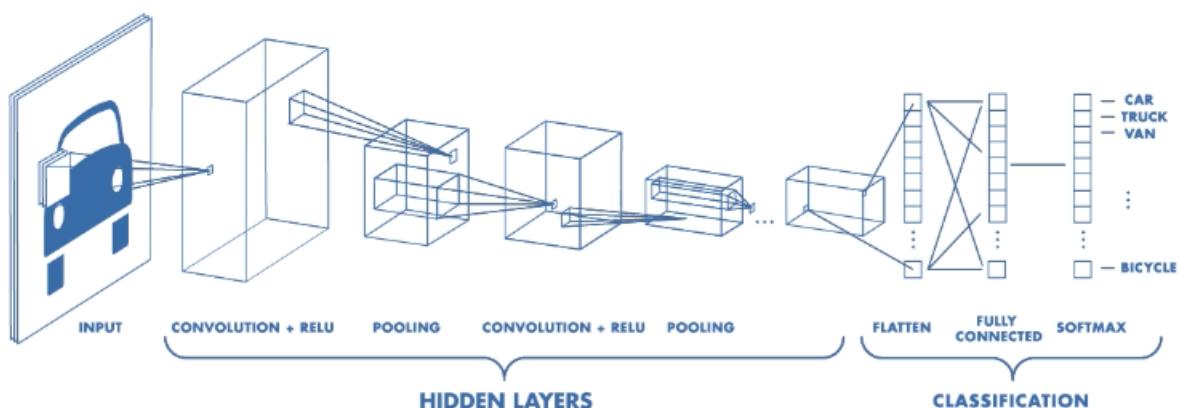


FIGURE 10 – Schéma montrant les étapes constituant un CNN, composées de convolutions, de poolings, pour finir avec la couche softmax

Les réseaux neuronaux **profonds** contiennent beaucoup plus de couches que les réseaux neuronaux classiques, on entre alors dans le domaine de l'*apprentissage profond*. Il a été observé que plus un réseau neuronal était profond, meilleures étaient ses performances.

On entraîna ainsi des réseaux de plus en plus profonds, jusqu'à atteindre une limite : les gradients (fonction réinitialisant les poids des couches intermédiaires lors de l'apprentissage) devenaient trop grands ou trop petits, ce qui rendait l'apprentissage instable. Pour y remédier, des "raccourcis" sont ajoutés entre l'entrée et la sortie d'une transformation. Plus précisément, si on considère la transformation F , au lieu d'avoir $\text{Sortie} = F(\text{Entrée})$, on aura $\text{Sortie} = F(\text{Entrée}) + \text{Entrée}$ (voir Figure 11).

Un bloc contenant une telle connexion est appelé **bloc résiduel**. Un réseau neuronal constitué de tels blocs est un **réseau neuronal résiduel**.

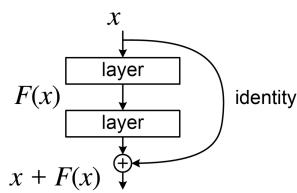


FIGURE 11 – Illustration d'un bloc résiduel

4 Prédiction conforme – Étude théorique

4.1 Contexte

La prédiction conforme est une méthode de classification utilisée en machine learning. Elle consiste à construire un sous-ensemble \mathcal{C} de l'ensemble des classes pour un niveau de confiance $\alpha \in]0, 1[$. C'est-à-dire que si Y est une variable aléatoire désignant le vrai label d'une observation qu'on tire aléatoirement, alors l'ensemble \mathcal{C} vérifie :

$$\mathbb{P}(Y \in \mathcal{C}) \geq \alpha.$$

Contrairement à la classification classique où nous n'avons qu'une seule prédiction, la prédiction conforme en donne plusieurs en sortie avec un score de conformité (la probabilité que celle-ci soit correcte) pour donner à l'utilisateur/trice le privilège de choisir parmi elles en optant pour le plus haut score ou bien pour une autre option qu'il/elle aura choisi en s'aidant de caractéristiques plus fines, en cas de doute entre plusieurs options. Les scores de conformité indiquent également le degré de certitude quant à la prédiction proposée, ce qui donne une confiance au procédé en question et au résultat final.

Une application pratique de la prédiction conforme est l'application mobile Pl@ntNet [2]. Pl@ntNet est une base de données constituée d'images de diverses plantes, fleurs, des fruits et autres. L'application mobile permet d'identifier une plante simplement en la photographiant. Son fonctionnement est simple : on prend en photo la plante ou la fleur qu'on veut identifier, l'application propose plusieurs résultats avec la probabilité d'être le bon résultat, puis on valide ce qu'on pense être la bonne réponse (12). Ainsi, on vient d'améliorer l'algorithme avec une image supplémentaire. Les utilisateurs permettent à la base de données de s'améliorer en y ajoutant de nouvelles espèces de plantes.

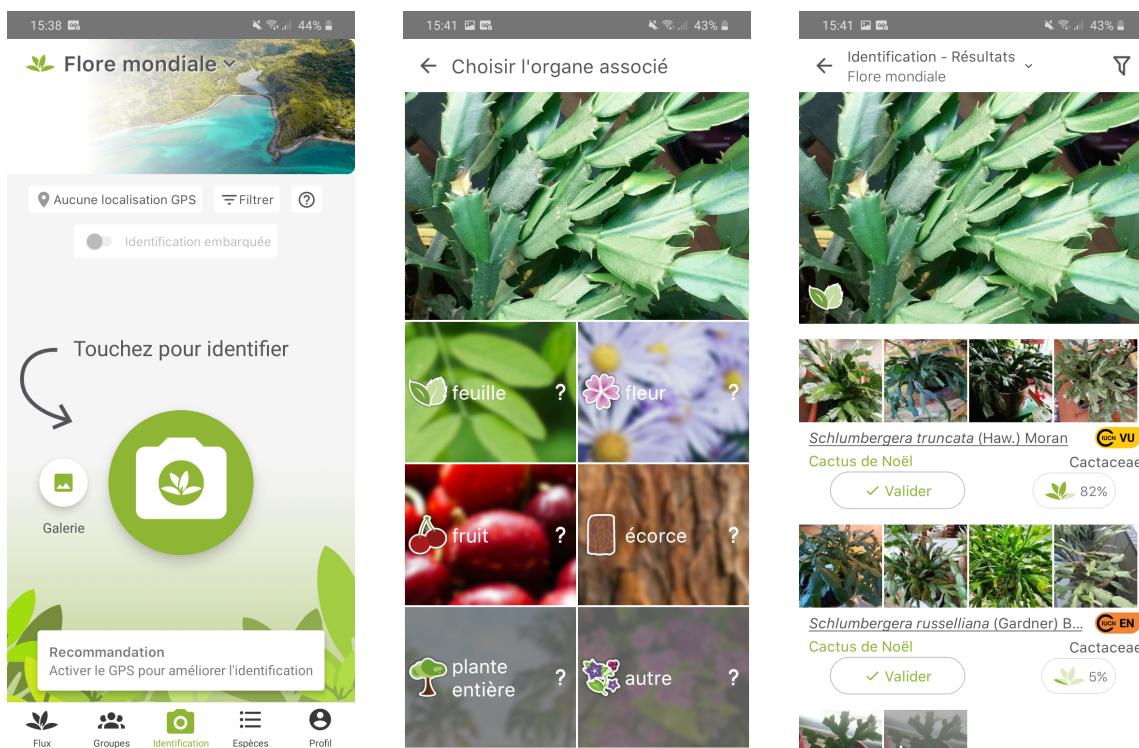


FIGURE 12 – Fonctionnement de Pl@ntNet

On pose $\mathcal{Z} = \{(x, y) \mid x \in \mathcal{X} \text{ et } y \in \mathcal{Y} \text{ est associé à } x\}$ l'espace de données. On définit la suite de variables aléatoires $\mathbf{Z} = (Z_1, \dots, Z_n)$ avec $Z_i = (X_i, Y_i)$ où $Y_i \in \mathcal{Y}$ est le label correspondant à la donnée $X_i \in \mathcal{X}$. On les suppose toutes indépendantes et identiquement distribuées selon une distribution de probabilité. On pose $z = (z_1, \dots, z_n)$ une réalisation de \mathbf{Z} , qu'on prendra comme données d'entraînement. Soit $(X, Y) \in \mathcal{Z}$ une nouvelle donnée choisie aléatoirement sur \mathcal{Z} .

On dispose de K agents. Pour $\alpha \in]0; 1[$, avec un niveau de confiance $1 - \alpha$, on définit $\mathcal{C}_k = \mathcal{C}_k(X, z)$ comme l'ensemble de prédiction donné par l'agent k , étant donné les données d'entraînement z . Alors, on a :

$$\forall k \in \{1, \dots, K\}, \mathbb{P}(Y \in \mathcal{C}_k) \geq 1 - \alpha \quad (1)$$

Le label Y étant choisi aléatoirement, il peut bien évidemment être une variable aléatoire. De ce fait, la distribution de la probabilité utilisée dans la propriété (1) est celle de (\mathbf{Z}, Y) . On dit que \mathcal{C}_k a une couverture *exacte* si $\mathbb{P}(Y \in \mathcal{C}_k) = 1 - \alpha$. Le but est d'agréger tous ces ensembles pour créer un nouvel ensemble de sorte à être optimal en couverture et en taille. Une première possibilité triviale est de définir \mathcal{C}^J comme l'union de tous les ensembles :

$$\mathcal{C}^J = \bigcap_{k=1}^K \mathcal{C}_k.$$

De toute évidence, \mathcal{C}^J vérifie la propriété définie en (1). Mais cet ensemble est trop grand pour être pertinent, on perd alors en précision. Quant à l'ensemble formé par l'intersection $\mathcal{C}^I = \bigcup_{k=1}^K \mathcal{C}_k$, il sera de plus en plus petit mais la couverture est inadéquate et l'ensemble est garanti d'avoir une couverture minimale de $1 - K\alpha$ par l'inégalité de Bonferroni ; cette garantie devient inutile lorsque K est grand.

4.2 Procédure du vote majoritaire

On définit un nouvel ensemble \mathcal{C}^M contenant toutes les classes $Y \in \mathcal{Y}$ votées par au moins la moitié des agents :

$$\mathcal{C}^M = \left\{ Y \in \mathcal{Y} : \frac{1}{K} \sum_{k=1}^K \mathbb{1}\{Y \in \mathcal{C}_k\} > \frac{1}{2} \right\}. \quad (2)$$

Théorème 4.1. *Si $\mathcal{C}_1, \dots, \mathcal{C}_K$ sont $K \geq 2$ ensembles de confiance différents basés sur les mêmes données, vérifiant la propriété (1), alors \mathcal{C}^M admet une couverture de $1 - 2\alpha$:*

$$\mathbb{P}(Y \in \mathcal{C}^M) \geq 1 - 2\alpha.$$

Démonstration. Soit $\phi_k = \mathbb{1}\{Y \notin \mathcal{C}_k\}$ une variable de Bernoulli telle que $\mathbb{E}[\phi_k] \leq \alpha$, pour $k \in \{1, \dots, K\}$. Alors, par l'inégalité de Markov, on a :

$$\mathbb{P}(Y \notin \mathcal{C}^M) = \mathbb{P}\left(\frac{1}{K} \sum_{k=1}^K \phi_k \geq \frac{1}{2}\right) \leq 2\mathbb{E}\left[\frac{1}{K} \sum_{k=1}^K \phi_k\right] = \frac{2}{K} \sum_{k=1}^K \mathbb{E}[\phi_k] \leq 2\alpha.$$

On en conclut la preuve. □

4.3 Autres seuils et borne supérieure

On peut généraliser la procédure précédente en prenant un seuil $\tau \in]0; 1[$, en définissant \mathcal{C}^τ comme l'ensemble des classes votées par une proportion τ d'agents :

$$\mathcal{C}^\tau = \left\{ Y \in \mathcal{Y} : \frac{1}{K} \sum_{k=1}^K \mathbf{1}\{Y \in \mathcal{C}_k\} > \tau \right\}.$$

Théorème 4.2. *Soient $\mathcal{C}_1, \dots, \mathcal{C}_K$ $K \geq 2$ ensembles de confiance différents basés sur les mêmes données, vérifiant la propriété (1), alors :*

$$\mathbb{P}(Y \in \mathcal{C}^\tau) \geq 1 - \frac{\alpha}{1 - \tau}.$$

Démonstration. La preuve de ce théorème est similaire à celle du précédent. On pose, pour $k \in \{1, \dots, K\}$ $\phi_k = \mathbb{P}(Y \notin \mathcal{C}_k)$ une variable de Bernoulli telle que $\mathbb{E}[\phi_k] \leq \alpha$, pour $k \in \{1, \dots, K\}$. Alors :

$$\mathbb{P}(Y \notin \mathcal{C}^\tau) = \mathbb{P}\left(\frac{1}{K} \sum_{k=1}^K \phi_k \geq 1 - \tau\right) \leq \frac{1}{1 - \tau} \mathbb{E}\left[\frac{1}{K} \sum_{k=1}^K \phi_k\right] = \frac{1}{1 - \tau} \sum_{k=1}^K \mathbb{E}[\phi_k] \leq \frac{\alpha}{1 - \tau}.$$

□

On y retrouve d'ailleurs les cas de l'intersection ($\tau = 1 - 1/K$) et de l'union ($\tau = 0$).

Toutefois, le vote majoritaire admet une borne supérieure, limitant ainsi sa couverture.

Théorème 4.3. *Soient $\mathcal{C}_1, \dots, \mathcal{C}_K$ $K \geq 2$ ensembles de confiance différents basés sur les mêmes données, et ayant une couverture exacte de $1 - \alpha$. Alors :*

$$\mathbb{P}(Y \in \mathcal{C}^M) \leq 1 - \frac{K\alpha - \lceil \frac{K}{2} \rceil + 1}{K - \lceil \frac{K}{2} \rceil + 1}.$$

Démonstration. Posons $r := \lceil \frac{K}{2} \rceil$, $\phi_k = \mathbf{1}\{Y \notin \mathcal{C}_k\}$ une variable de Bernoulli telle que $\mathbb{E}[\phi_k] = \alpha$ et $S_K = \sum_{k=1}^K \phi_k$.

Par définition de S_K , on a :

$$\mathbb{E}[S_K] = \sum_{k=1}^K \mathbb{E}[\phi_k] = K\alpha.$$

Posons $\rho_j = \mathbb{P}(S_K = j)$, pour $j \in \{0, \dots, K\}$. En écrivant d'une autre manière $\mathbb{E}[S_K]$, on a :

$$\begin{aligned} \mathbb{E}[S_K] &= \sum_{j=0}^K j\rho_j \\ &= \sum_{j=0}^{r-1} j\rho_j + \sum_{j=r}^K j\rho_j + (r-1) \sum_{j=0}^{r-1} \rho_j + K \sum_{j=r}^K \rho_j - (r-1) \sum_{j=0}^{r-1} \rho_j - K \sum_{j=r}^K \rho_j \\ &= (r-1) \sum_{j=0}^{r-1} \rho_j + K \sum_{j=r}^K \rho_j - \sum_{j=0}^{r-1} (r-1-j)\rho_j - \sum_{j=r}^K (K-j)\rho_j \\ &= (r-1)(1 - \mathbb{P}(S_K \geq r)) + K\mathbb{P}(S_K \geq r) - m ; \quad \mathbb{P}(S_K \geq r) = \sum_{j=0}^{r-1} \rho_j = \sum_{j=r}^K \rho_j \end{aligned}$$

où $m = \sum_{j=0}^{r-1} (r-1-j)\rho_j + \sum_{j=r}^K (K-j)\rho_j$.

Tant que $m \geq 0$ et que pour $K \geq 0$, $K - \lceil \frac{K}{2} \rceil + 1 > 0$:

$$K\alpha \leq (r-1) \left(1 - \mathbb{P}\left(S_K \geq \lceil \frac{K}{2} \rceil\right) \right) + K\mathbb{P}\left(S_K \geq \lceil \frac{K}{2} \rceil\right) \iff \mathbb{P}\left(S_K \geq \lceil \frac{K}{2} \rceil\right) \geq \frac{K\alpha - r + 1}{K - r + 1}.$$

De plus, par le théorème 4.1, on a $\mathbb{P}(Y \notin \mathcal{C}^M) = \mathbb{P}\left(S_K \geq \lceil \frac{K}{2} \rceil\right)$, ce qui nous donne :

$$\mathbb{P}(Y \in \mathcal{C}^M) \leq 1 - \frac{K\alpha - \lceil \frac{K}{2} \rceil + 1}{K - \lceil \frac{K}{2} \rceil + 1}.$$

□

Ce résultat s'étend à l'ensemble \mathcal{C}^τ , pour $\tau \in]0; 1[$.

Théorème 4.4. Soient $\mathcal{C}_1, \dots, \mathcal{C}_K$ $K \geq 2$ ensembles de confiance différents basés sur les mêmes données, et ayant une couverture exacte de $1 - \alpha$. Alors, pour $\tau \in]0; 1[$, on a :

$$\mathbb{P}(Y \in \mathcal{C}^\tau) \leq 1 - \frac{K\alpha - \lceil K(1-\tau) \rceil + 1}{K - \lceil K(1-\tau) \rceil + 1}.$$

Pour montrer ce théorème, il suffit de poser $r := \lceil K(1-\tau) \rceil$, et de reprendre les hypothèses et le déroulé de la preuve précédente.

Maintenant que nous connaissons les bornes inférieure et supérieure de $\mathbb{P}(Y \in \mathcal{C}^\tau)$, ci-dessous se trouve le graphique montrant la bande d'appartenance de $\mathbb{P}(Y \in \mathcal{C}^\tau)$, pour différentes valeurs de α .

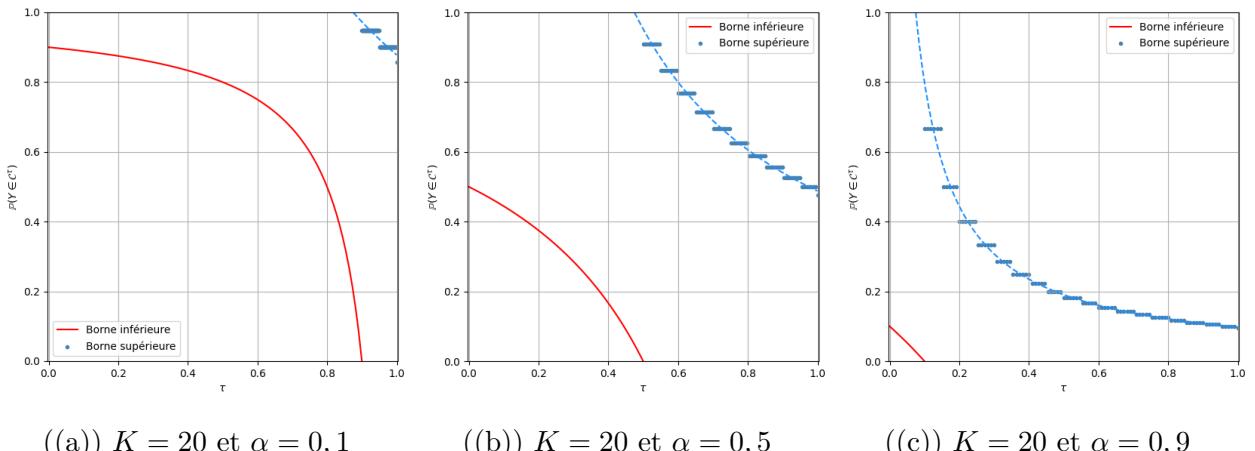


FIGURE 13 – Encadrement de $\mathbb{P}(Y \in \mathcal{C}^\tau)$ pour différentes valeurs de α

Dans la figure ci-dessus, les paliers bleus forment la borne supérieure et la courbe rouge forme la borne inférieure de $\mathbb{P}(Y \in \mathcal{C}^\tau)$, la ligne pointillée bleue étant ici pour aider le lecteur à suivre l'évolution de la borne supérieure (à titre informatif, elle passe par le milieu des paliers). Via ces 3 figures, nous pouvons observer que dans le cas du vote majoritaire classique ($\tau = 0.5$) :

- en $\alpha = 0, 1$: les agents sont sûrs à 90% au moins, ce qui donne une bonne couverture d'au moins 80% ;

- en $\alpha = 0,5$: les agents n'ont 1 chance sur 2 de croire en leurs propositions. De ce fait, il est normal de s'attendre à tout, d'où une couverture entre 0 et 100%. C'est en quelque sorte le pire cas dans la moitié ayant sélectionnée ladite classe, chacun d'entre eux a toujours 1 chance sur 2 d'avoir faux ;
- en $\alpha = 0,9$: les agents n'ont pas confiance en eux. S'ils n'ont même pas confiance en eux, on ne peut pas faire confiance à leurs propositions. Logiquement, on a dans ce cas $\mathbb{P}(Y \in \mathcal{C}^\tau) \leq \frac{2}{11}$.

Bien évidemment le meilleur cas est celui où $\alpha = 0,1$; plus généralement, plus les agents sont sûrs d'eux, plus on peut faire confiance à l'ensemble \mathcal{C}^τ final.

Ce qu'il faut comprendre de \mathcal{C}^τ , c'est que pour qu'une certaine classe y appartienne à celui-ci, il faut qu'une proportion minimale τ d'agents ait émis cette proposition. Autrement dit, plus τ est proche de 1, plus on est exigeant quant à l'appartenance à \mathcal{C}^τ . Cela signifie que même avec $\alpha = 0,1$, si τ est proche de 1, il devient difficile pour une classe y d'être prise, donc la probabilité qu'elle y est en est plus faible. On peut observer ce cas de figure dans la figure 13(a) : plus τ se rapproche de 1, plus la borne inférieure décroît fortement jusqu'à atteindre 0 en $\tau = 0,9$.

4.4 Combiner des ensembles mutuellement indépendants

Si on suppose les ensembles \mathcal{C}_k indépendants entre eux, alors on peut améliorer le seuil pour avoir un niveau de confiance $1 - \alpha$. Pour cela, il suffit de prendre comme seuil le quantile d'ordre α pour la loi binomiale $\mathcal{B}(K, 1 - \alpha)$. On définit le quantile $Q_K(\alpha)$ par :

$$Q_K(\alpha) := \sup\{x \in \mathbb{R} \mid F(x) \leq \alpha\}$$

où F est la fonction de répartition de la loi binomiale ci-dessus.

Proposition 4.5. *Soient $\mathcal{C}_1, \dots, \mathcal{C}_K$ $K \geq 2$ différents ensembles de prédiction indépendants, vérifiant la propriété (1). Alors l'ensemble*

$$\mathcal{C}^M = \left\{ Y \in \mathcal{Y} : \sum_{k=1}^K \mathbf{1}\{Y \in \mathcal{C}_k\} > Q_K(\alpha) \right\}$$

admet une couverture de niveau $1 - \alpha$.

Nous aurons besoin d'un lemme pour prouver la proposition ci-dessus. Dans ce qui suit, on note $X \hookrightarrow PB(p_1, \dots, p_K)$ la variable aléatoire qui suit la loi Poisson binomiale, qui est la somme de K variables de Bernoulli indépendantes de paramètres p_1, \dots, p_K .

Lemme 4.6. *Soient $X \hookrightarrow PB(p_1, \dots, p_K)$ et $Y \hookrightarrow \mathcal{B}(K, p)$. Alors X est stochastiquement plus grand que Y , qu'on note $X \geq_{st} Y$ (c'est-à-dire $\forall x \in \mathbb{R}$, $\mathbb{P}(X > x) \geq \mathbb{P}(Y > x)$) si*

$$p^K \leq \prod_{k=1}^K p_k.$$

La preuve du lemme est donnée dans [4] et discutée dans [6].

Preuve de la proposition. Rappelons que les ensembles \mathcal{C}_k sont aléatoires par leur construction. Dès lors, l'ensemble \mathcal{C}^M défini dans la proposition ci-dessus est aussi aléatoire. Le terme $\mathbb{P}(Y \in \mathcal{C}^M)$ a alors un sens.

Posons en premier lieu $S_K = \sum_{k=1}^K \mathbf{1}\{Y \in \mathcal{C}_k\}$. Les quantités $\mathbf{1}\{Y \in \mathcal{C}_k\}$ sont des variables de Bernoulli de paramètre $p_k \geq 1 - \alpha$. Elles sont indépendantes car les ensembles \mathcal{C}_k le sont par hypothèse. La variable aléatoire S_K suit donc une loi Poisson binomiale de paramètres p_1, \dots, p_K . Notons que ces derniers sont tels que $(1 - \alpha)^K \leq \prod_{k=1}^K p_k$.

Considérons la variable aléatoire $Y \hookrightarrow \mathcal{B}(K, 1 - \alpha)$. Alors on a que Y et S_K sont telles que $(1 - \alpha)^K \leq \prod_{k=1}^K p_k$. Le Lemme 4.6 implique donc que $S_K \geq_{\text{st}} Y$.

On en déduit le résultat final :

$$\begin{aligned} \mathbb{P}(Y \in \mathcal{C}^M) &= \mathbb{P}\left(\sum_{k=1}^K \mathbf{1}\{Y \in \mathcal{C}_k\} > Q_K(\alpha)\right) \\ &= \mathbb{P}(S_K > Q_K(\alpha)) \\ &> \mathbb{P}(Y > Q_K(\alpha)) \quad \text{car } S_K \geq_{\text{st}} Y \end{aligned}$$

Or, par définition de $Q_K(\alpha)$: $\mathbb{P}(Y \leq Q_K(\alpha)) < \alpha \iff \mathbb{P}(Y > Q_K(\alpha)) \geq 1 - \alpha$. On conclut donc que $\mathbb{P}(Y \in \mathcal{C}^M) \geq 1 - \alpha$.

□

Pourquoi avoir choisi la loi $\mathcal{B}(K, 1 - \alpha)$ pour le quantile et pas une autre ? Avant tout, le Lemme 4.6 nous permet de travailler avec cette loi binomiale plutôt que la loi Poisson binomiale car nous n'en connaissons pas de fonction de répartition. La seule différence est que dans la loi binomiale, les paramètres p_1, \dots, p_K sont tous égaux. Supposons alors que tous les agents ont le même niveau de confiance $1 - \alpha$ et que leur ensemble \mathcal{C}_k a une couverture exacte de $1 - \alpha$. Nous avons alors pour tout $k \in \{1, \dots, K\}$:

$$\mathbb{P}(Y \in \mathcal{C}_k) = \mathbb{E}[\mathbf{1}\{Y \in \mathcal{C}_k\}] = 1 - \alpha.$$

Si on note $S_K = \sum_{k=1}^K \mathbf{1}\{Y \in \mathcal{C}_k\}$, alors S_K est une variable aléatoire réelle puisque $\mathbf{1}\{Y \in \mathcal{C}_k\}$ l'est aussi à cause de \mathcal{C}_k , pour tout k . L'expérience ci-dessus revient à construire un ensemble de prédictions \mathcal{C} tel qu'on ait $\mathbb{P}(Y \in \mathcal{C}) = 1 - \alpha$, le tout en répétant K fois de façon identique et indépendante. Alors la variable aléatoire S_K suit la loi $\mathcal{B}(K, 1 - \alpha)$.

5 Prédiction conforme – Comparaison des différentes méthodes d’agrégation

Une fois les différentes méthodes définies et leurs couvertures théoriques étudiées, il est temps de vérifier si ces méthodes sont bien efficaces avec des simulations numériques. Pour juger de l’efficacité d’une méthode en particulier, nous calculerons l’erreur moyenne sur un nombre N de répétitions de l’expérience, au fur et à mesure que l’on augmente le nombre d’agents K .

Pour détailler le protocole, prenons l’exemple du vote majoritaire. Nous appellerons un "run" une étape durant laquelle nous déterminons les ensembles $\mathcal{C}_1, \dots, \mathcal{C}_K$. Ces derniers sont définis uniquement à cette étape-là. Lorsque nous prenons un "run" différent, nous aurons des ensembles $\mathcal{C}_1, \dots, \mathcal{C}_K$ différents. Ainsi, pour avoir N runs, il suffit de réitérer la 1ère expérience de façon identique et indépendante $N - 1$ fois.

Nous noterons Γ l’ensemble résultant de l’agrégation de plusieurs ensembles de prédiction des K agents, quelle que soit la méthode d’agrégation. Dans cet exemple, on a $\Gamma = \mathcal{C}^M$.

Dans toutes nos simulations, nous ferons $N = 10$ runs et prendrons jusqu’à $K = 50$ agents avec un niveau de confiance de $1 - \alpha = 0.95$, soit $\alpha = 0.05$, l’objectif étant d’étudier la convergence de l’erreur moyenne de classification et celle de la taille moyenne de Γ lorsqu’on prend des grandes valeurs de K . À chaque étape $k \in \{1, \dots, K\}$, l’erreur moyenne de classification se calcule comme suit :

$$E_k = \frac{1}{N} \sum_{n=1}^N \mathbf{1}\{Y \notin \mathcal{C}_{n,k}\}$$

où $\mathcal{C}_{n,k}$ est l’ensemble de prédiction de l’agent k pour le run n .

En théorie, pour avoir nos K agents, nous devons prendre K modèles de machine learning différents, tous entraînés sur la même base de données ImageNet. Or si on veut savoir ce qui se passe pour de grandes valeurs de K , il faut avoir en main autant de modèles différents prêts à l’emploi, ce qui n’est pas une tâche aisée.

Afin de s’en rapprocher, une alternative serait de prendre un modèle déjà entraîné (dans notre cas, le CNN ResNet-50 : réseau neuronal résiduel à 50 couches), prendre la pire classe en termes de performance, prendre comme agents le modèle ResNet-50 avec en entrée des images différentes de la même classe.

Ainsi, la classe à prédire sera la même, tandis que les probabilités issues du modèle seront différentes car ainsi seront les entrées. Grâce à cette méthode, nous pouvons avoir jusqu’à 1300 agents, mais nous nous contenterons de 50 au maximum.

Avant de lancer les simulations, il y a une chose en plus à savoir sur la prédiction conforme : une fois l’ensemble Γ déterminé, chacune des classes de cet ensemble a ce qu’on appelle un score de conformité. Le **score de conformité** d’une classe prédite est la probabilité que celle-ci soit correcte.

En plus des graphes montrant l’évolution de l’erreur moyenne et la taille moyenne de Γ , on affichera également l’évolution du score de conformité de la vraie classe, selon la méthode d’agrégation utilisée, en fonction de K . Pour calculer ce dernier, on procède comme suit :

- on construit les ensembles $\mathcal{C}_1, \dots, \mathcal{C}_K$;
- on construit Γ en les agrégeant ;
- pour chaque classe de Γ , on calcule la moyenne des probabilités correspondantes issues de la fonction softmax. Autrement dit, si on note, pour l'image n° k avec $k \in \{1, \dots, K\}$, $(p_{i,1}, \dots, p_{i,L})$ le vecteur de probabilités issu de softmax, avec $L = 1000$ le nombre de classes, alors le score de conformité SC_l d'une classe l prédite se calcule de la façon suivante :

$$SC_l = \frac{1}{K} \sum_{k=1}^K p_{k,l}.$$

5.1 Union et intersection

En faisant l'union des ensembles \mathcal{C}_k , on inclut de plus en plus de classes sans en enlever. Il est évident qu'en conséquences, l'erreur moyenne de Γ_k diminue et converge vers 0 et sa taille moyenne augmente sans converger, si ce n'est vers 1000.

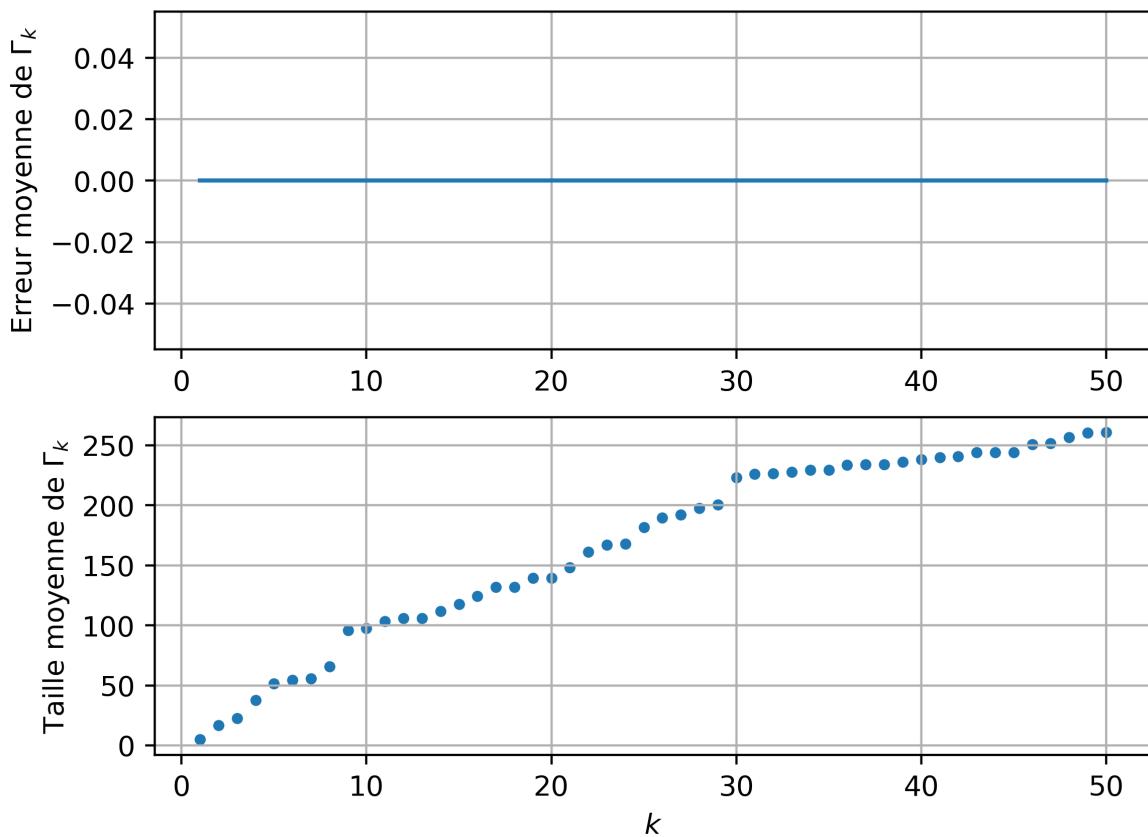


FIGURE 14 – Méthode d'agrégation utilisée : union

Quant à l'intersection, à l'inverse, nous nous privons de plus en plus de classes, ce qui a l'effet inverse : l'erreur moyenne augmente et la taille moyenne atteint la valeur de 0 à partir d'un certain rang.

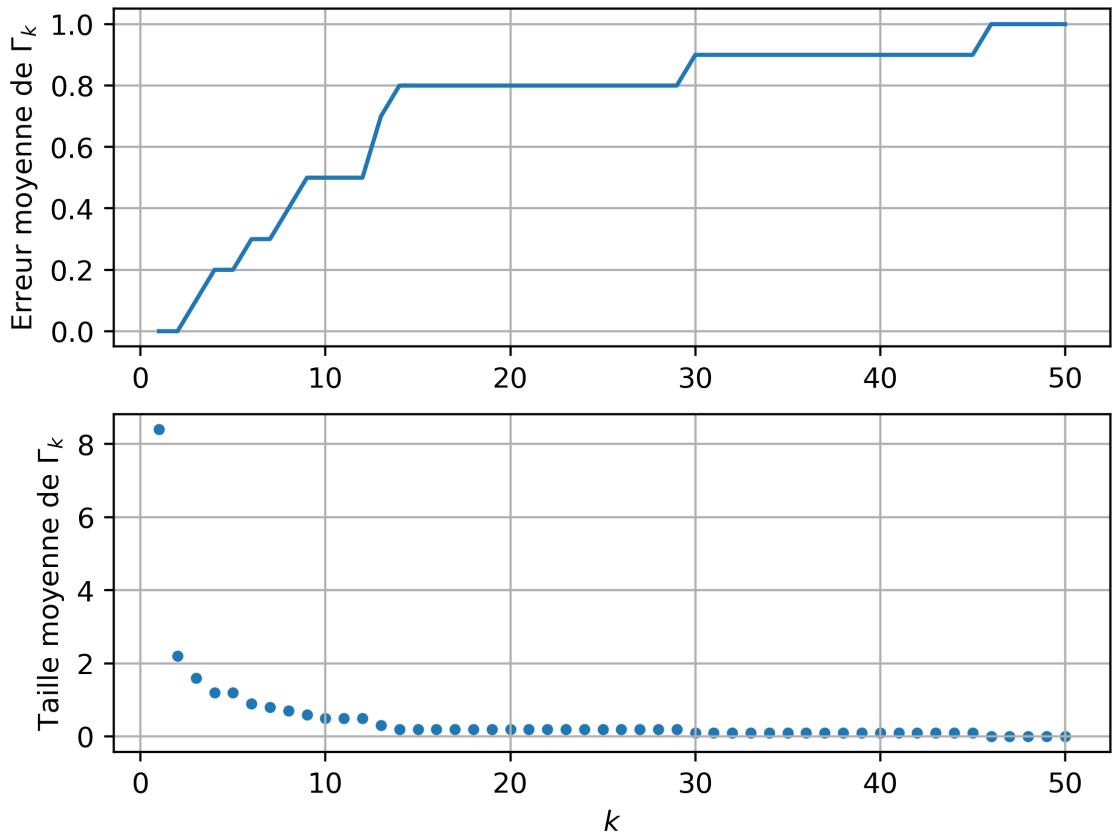


FIGURE 15 – Méthode d’agrégation utilisée : intersection

Bien que ces deux méthodes sont les premières auxquelles on pourrait penser lorsqu’on parle d’agrégation, elles s’avèrent inutilisables en pratique.

Une méthode intuitive serait de choisir une classe si elle est proposée par au moins la moitié des agents : c’est le vote majoritaire. Dans le cas où tous les agents ne font qu’une seule proposition et où il existe une classe choisie par au moins la moitié, on retrouve le principe de la majorité absolue.

5.2 Vote majoritaire classique

Pour rappel, l’ensemble Γ correspondant, qu’on note \mathcal{C}^M , est défini par :

$$\mathcal{C}^M = \left\{ Y \in \mathcal{Y} \mid \frac{1}{K} \sum_{k=1}^K \mathbf{1}\{Y \in \mathcal{C}_k\} > \frac{1}{2} \right\}$$

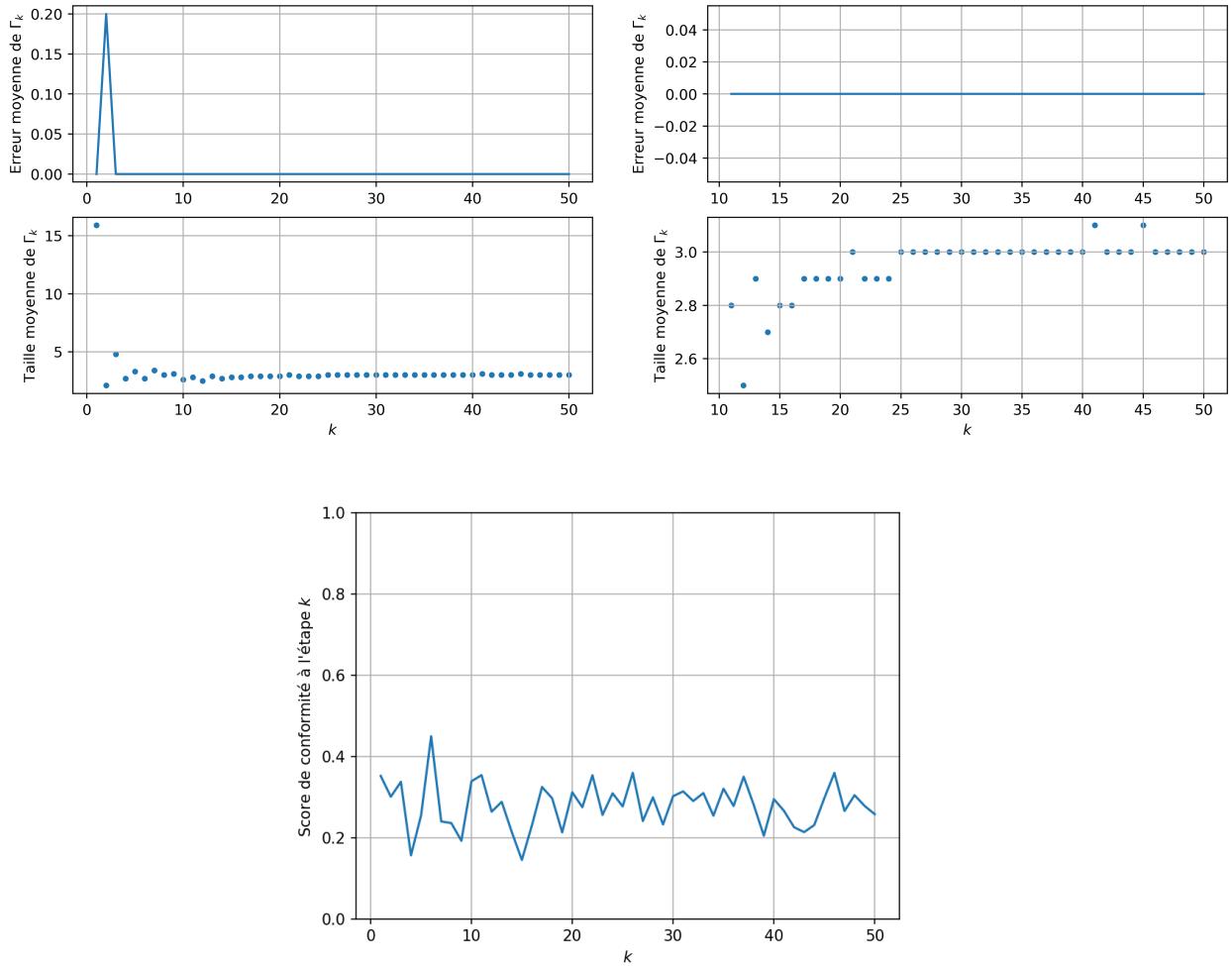


FIGURE 16 – Méthode d’agrégation utilisée : vote majoritaire classique

Le vote majoritaire donne de bons résultats, que ce soit pour l’erreur moyenne ou la taille moyenne qui converge ici vers 3, comme on peut le voir sur la figure de droite de (16). Une telle valeur reste trop élevée pour l’objectif à atteindre. Voyons si on peut améliorer ces résultats en modifiant la valeur du seuil.

5.3 Vote majoritaire généralisé

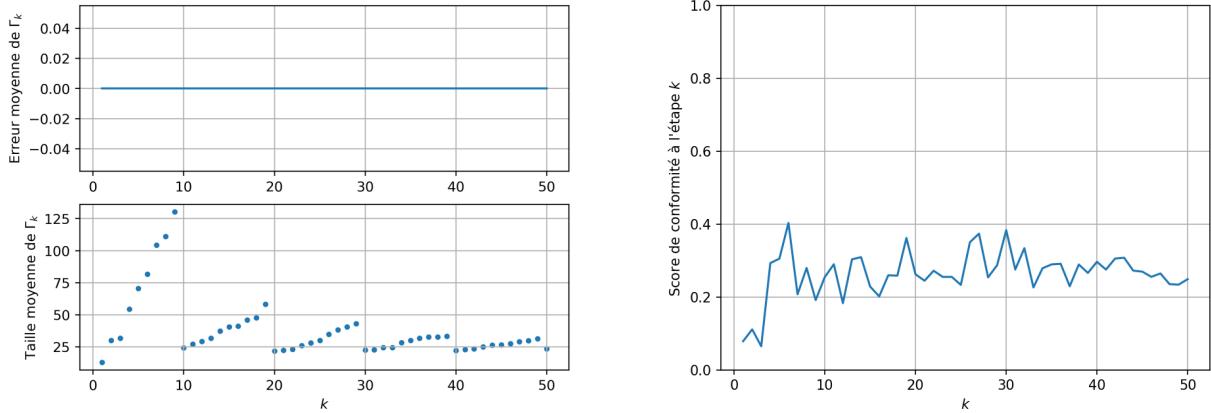


FIGURE 17 – Vote majoritaire généralisé avec $\tau = 0, 1$

À $\tau = 0, 1$, non seulement la taille moyenne converge vers une valeur proche de 25, mais la convergence est telle que la taille moyenne ne dépassera pas la valeur de 30 à partir de $k = 30$. La taille limite est beaucoup trop grande et cette méthode est trop inefficace pour notre besoin.

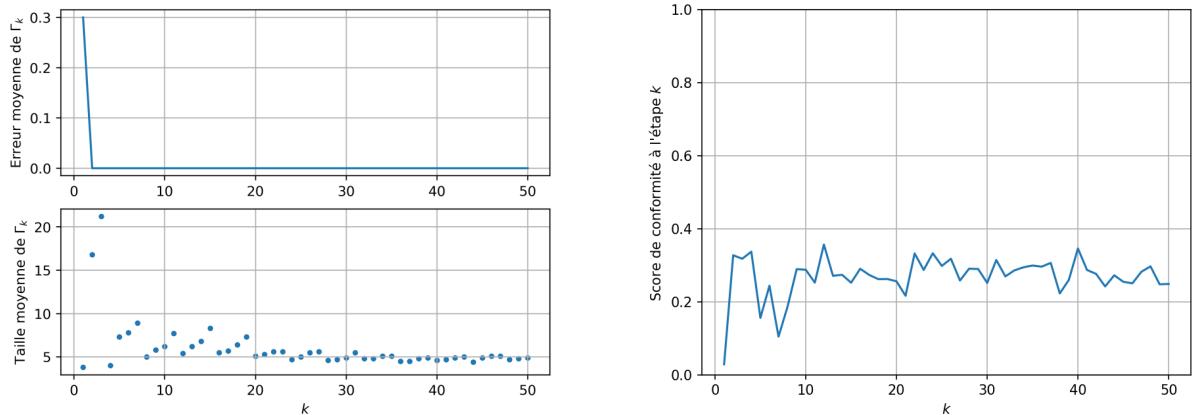


FIGURE 18 – Vote majoritaire généralisé avec $\tau = 0, 25$

À $\tau = 0.25$, la taille moyenne converge plus rapidement et vers une valeur inférieure à celle d'avant, ici 5, mais elle est toujours trop élevée. Continuons de chercher jusqu'à pouvoir converger vers 1.

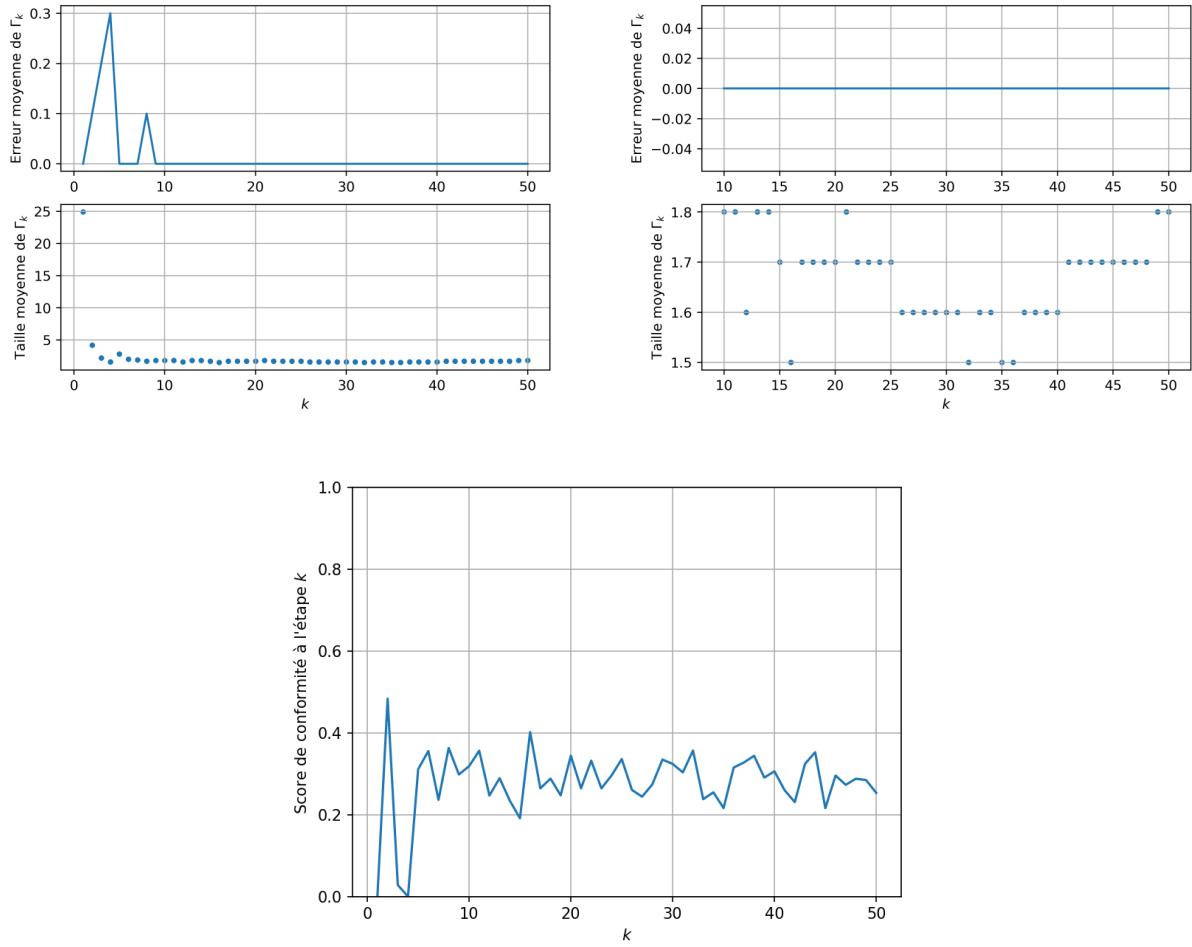


FIGURE 19 – Vote majoritaire généralisé avec $\tau = 0,75$

Nous sautons le cas $\tau = 0,5$ puisque cela revient au cas du vote majoritaire classique dont le résultat se situe Figure 16.

À $\tau = 0,75$, on peut remarquer que l'erreur moyenne a une vitesse de convergence plus faible, là où dans les précédents cas, on atteignait plus rapidement la valeur nulle. Cependant, la limite de la taille continue de baisser avec une valeur oscillant entre 1,5 et 1,8, ce qui est bon signe.

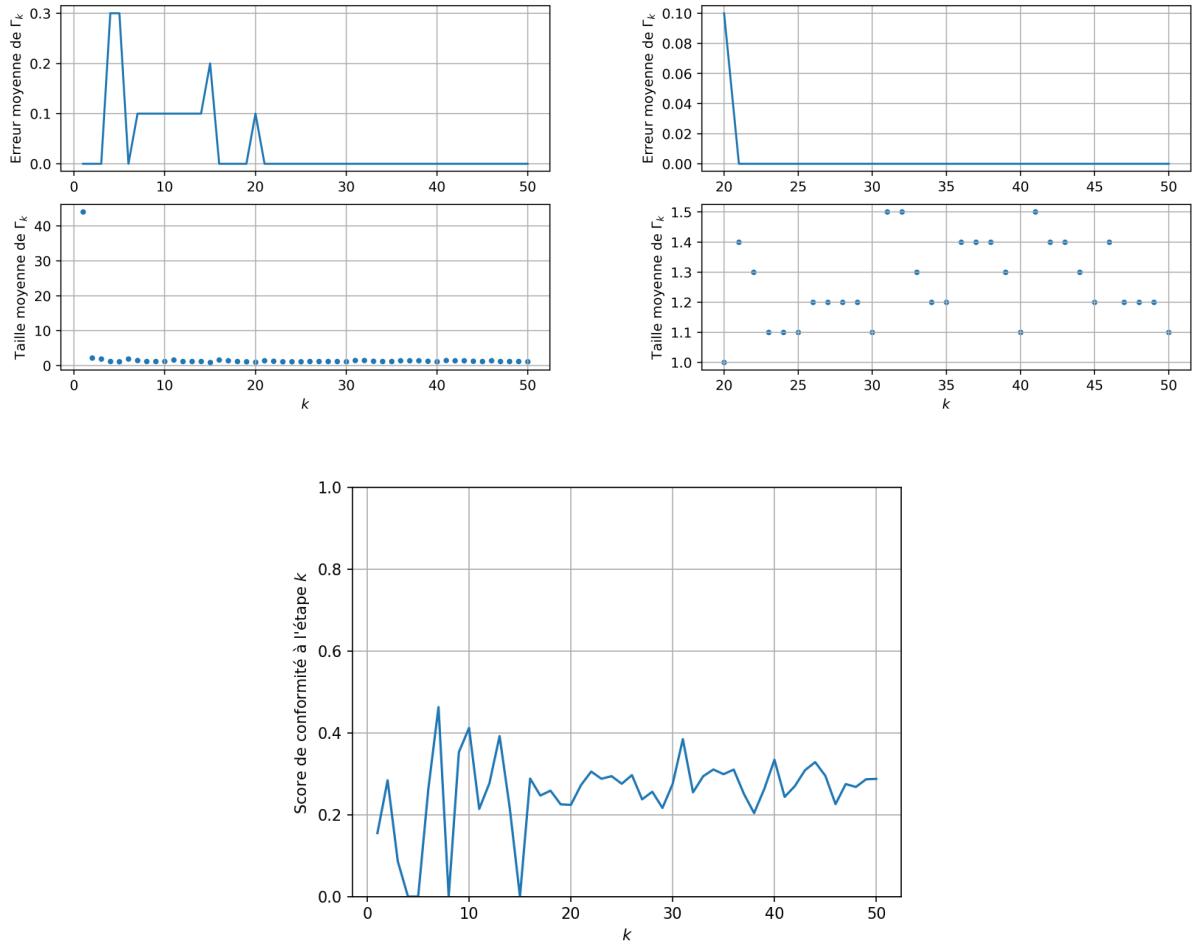


FIGURE 20 – Vote majoritaire généralisé avec $\tau = 0,8$

Pour ce qui est du cas où $\tau = 0,8$, nous avons toujours une taille moyenne qui converge vers une valeur encore plus proche de 1 avec une limite se situant entre 1 et 1,5. Cependant il est à noter que, au fur et à mesure qu'on augmente la valeur de τ , bien entendu la taille moyenne limite se rapproche de plus en plus de 1, mais l'erreur moyenne converge vers 0 avec une vitesse de plus en plus faible.

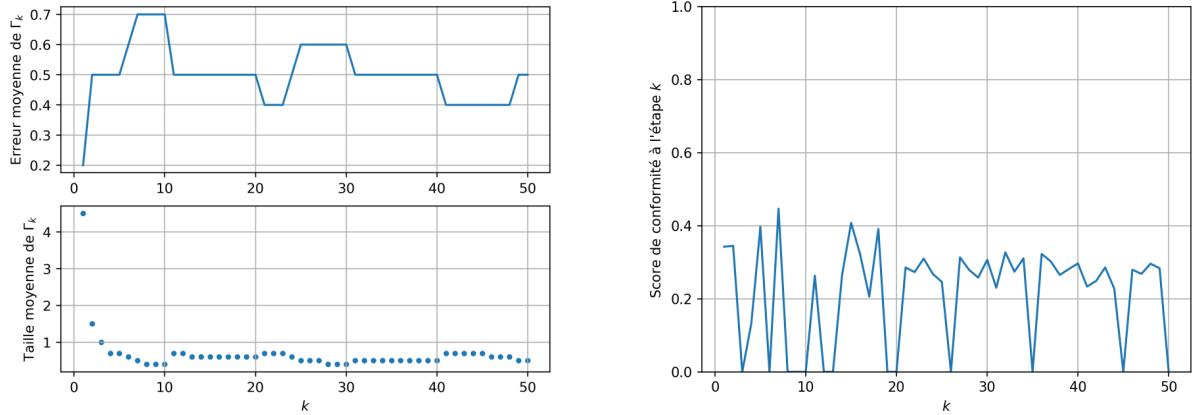


FIGURE 21 – Vote majoritaire généralisé avec $\tau = 0,9$

À $\tau = 0,9$, la valeur limite de la taille moyenne est devenue inférieure à 1 mais cette fois-ci,

l'erreur moyenne semble converger vers une valeur proche de 0,5. Notre objectif est d'avoir une méthode d'agrégation qui donne en sortie un ensemble Γ ne contenant idéalement qu'**un** élément : la classe attendue. On se contentera d'une taille limite supérieure à 1 mais la plus proche de 1 si possible. Pour continuer les recherches, on peut creuser dans l'intervalle $[0, 8; 0, 9]$ avec, par exemple, $\tau = 0, 85$.

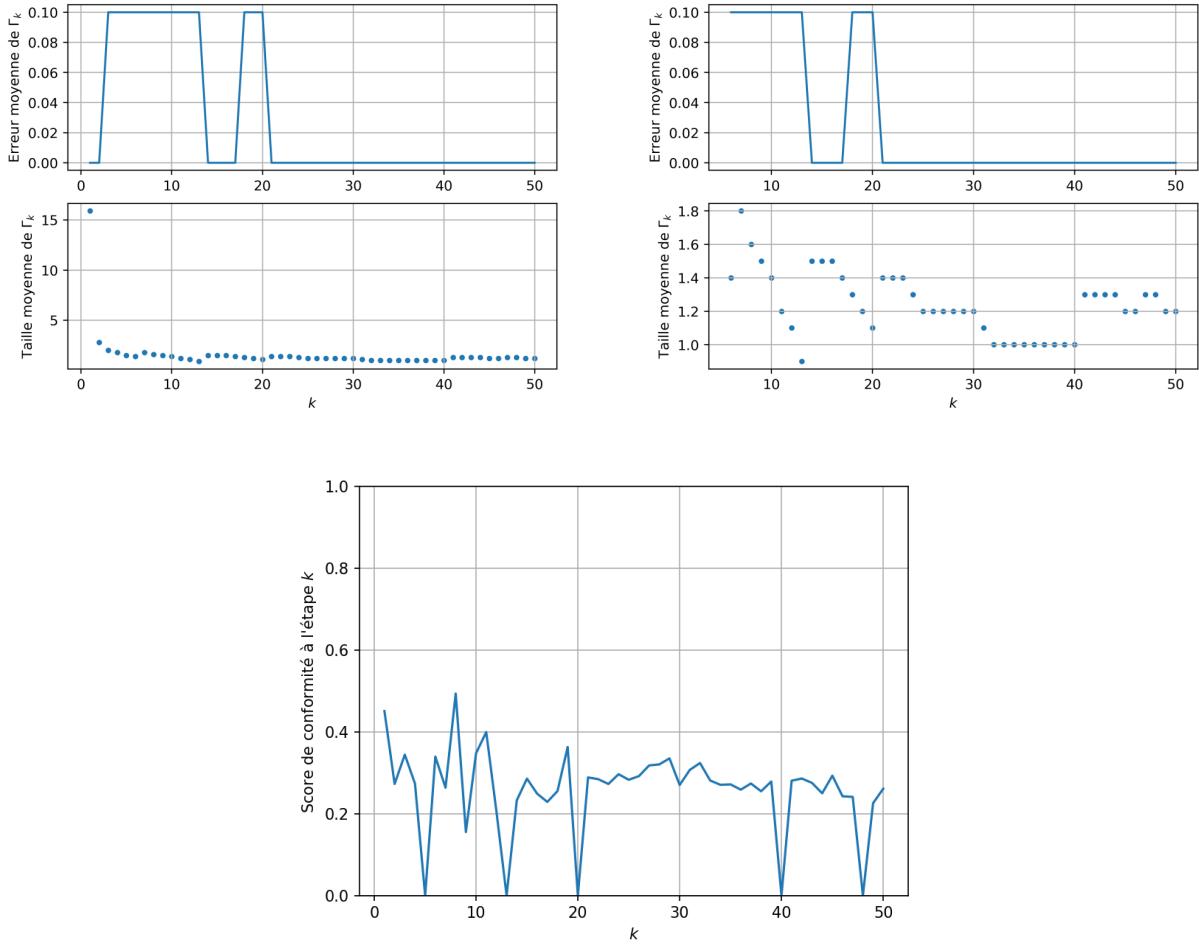


FIGURE 22 – Vote majoritaire généralisé avec $\tau = 0, 85$

Observons que, tandis que la limite de la taille moyenne se rapproche de plus en plus de 1, l'erreur moyenne de Γ converge toujours vers 0, mais à une vitesse de plus en plus faible. Autrement dit, pour s'assurer de la justesse du résultat, nous devrions prendre de plus en plus d'agents dans le but de minimiser l'erreur de prédiction finale.

Pour rappel, on a :

$$\mathbb{P}(Y \in \mathcal{C}^\tau) \geq 1 - \frac{\alpha}{1 - \tau}.$$

Donc au fur et à mesure qu'on augmente τ , la borne inférieure décroît fortement jusqu'à $-\infty$. Il devient alors plus probable d'avoir des erreurs dans le cas où la probabilité $\mathbb{P}(Y \in \mathcal{C}^\tau)$ est potentiellement petite.

D'un autre côté, lorsqu'on observe l'évolution du score de conformité dans les Figures 17, 18, 16, 19, 20 et 22, elle est assez similaire dans le sens où entre 0 et 10, il atteint environ 40%

tout en pouvant être faible parfois. Pour se rendre compte naïvement de la grandeur du score de 40%, on peut imaginer que si notre classe prend 40%, les 999 autres classes se partagent les 60%. Après $k = 10$, il se stabilise et a tendance à converger vers une valeur autour de 30%, sauf dans la Figure 22 où il chute à 2 reprises mais reste toutefois fiable. Par contre la Figure 21 confirme le rejet du choix $\tau = 0,9$, au vu du nombre d’itérations où la vraie classe Y n’est pas choisie pour l’ensemble final.

Après observation des erreurs et tailles moyennes ainsi que des scores de conformité, on peut conclure que si on choisit comme méthode d’agrégation le vote majoritaire, alors les paramètres $\tau = 0.85$ et $K \in [15, 25]$ sont ceux qui conviennent le mieux.¹

5.4 Seuil égal au quantile d’ordre α

Dans cette section, nous prenons comme seuil le quantile d’ordre α défini pour la loi binomiale $\mathcal{B}(K, 1 - \alpha)$ comme ci-dessous :

$$Q_K(\alpha) = \sup\{x \in \mathbb{R} \mid F(x) \leq \alpha\}.$$

Sont répertoriées à la Figure 23 ci-dessous 3 simulations du vote majoritaire avec comme seuil le quantile (la colonne de droite est une version zoomée sur la plage $k \in \{10, \dots, 50\}$ de la colonne gauche) :

1. À noter que ces valeurs ne sont en rien fixées, elles sont jugées adéquates dans cette situation particulière où le modèle utilisé est ResNet-50. L’utilisateur est libre de choisir le paramètre τ à sa volonté. Le nombre K représentant le nombre d’agents, il est théoriquement possible de prendre une grande valeur de K mais cela ne sera pas toujours faisable en pratique.

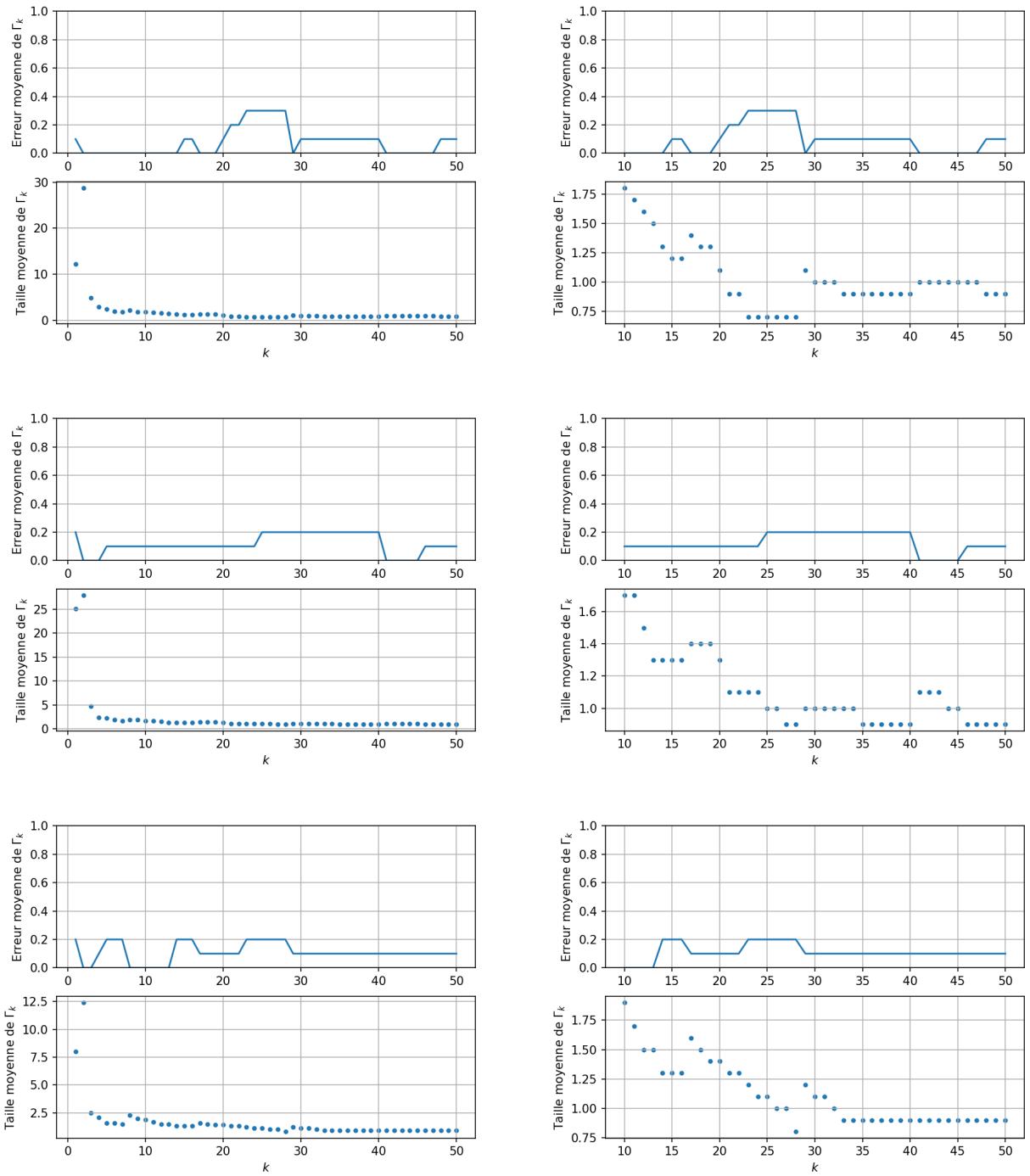


FIGURE 23 – Vote majoritaire : seuil = quantile d'ordre α

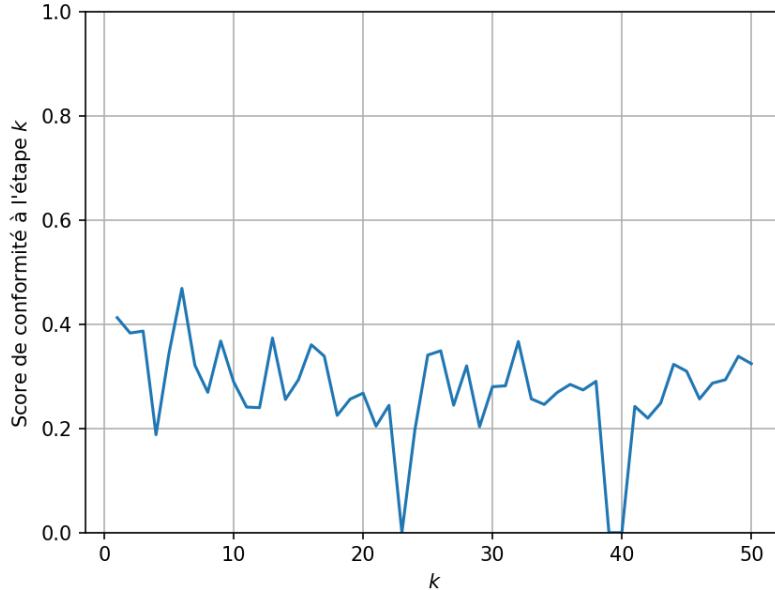


FIGURE 24 – Score de conformité avec comme seuil le quantile

Dans le cas où on prend comme seuil $Q_K(\alpha)$, nous pouvons constater dans la Figure 23 qu'à chaque fois, l'erreur moyenne semble plafonner 0,2, parfois 0,3, et la taille moyenne converge systématiquement vers une valeur inférieure à 1. Pour avoir une taille moyenne de Γ comprise entre 1 et 1,5 on peut s'arrêter à $K = 15$. Quant au score de conformité (voir Figure 24), il reste aussi haut aux environs de 40% pour k compris entre 0 et 20 et semble converger vers 30%, tout comme les précédents votes majoritaires.

5.5 Moyenne des probabilités issues de softmax

Une autre méthode a été expérimentée pour faire de l'agrégation. Voici les étapes pour construire l'ensemble Γ :

- obtenir les vecteurs de probabilités en sortie de la fonction softmax, des K agents ;
- faire la moyenne de ces probabilités classe par classe. On obtient un vecteur de taille 1000 ;
- trier ses coordonnées par ordre décroissant et sélectionner les classes jusqu'à ce que la somme de leurs probabilités dépasse $1 - \alpha$.

On procède ensuite comme on ferai pour le vote majoritaire, par exemple, en faisant la moyenne sur $N = 10$ répétitions, et ce pour k allant de 1 à $K = 50$.

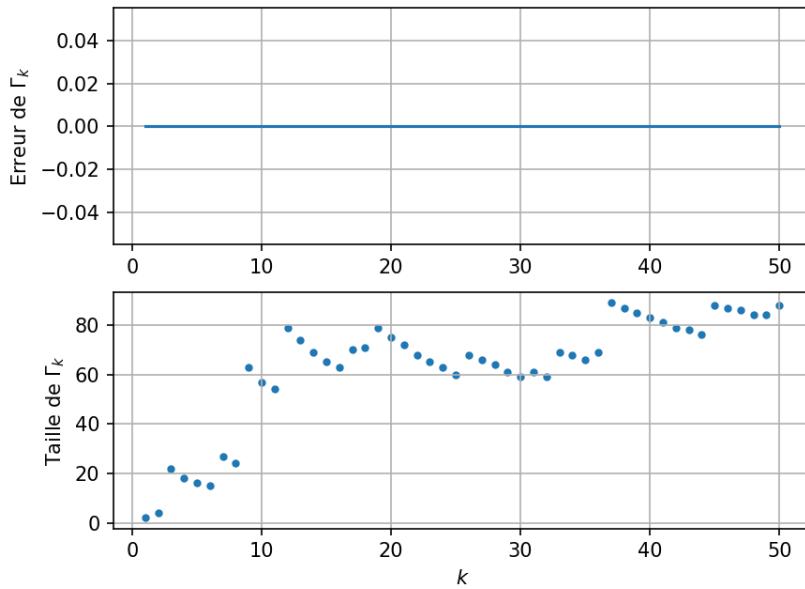


FIGURE 25 – Moyenne des probabilités de softmax

La Figure 25 qui en résulte est étonnante puisque là où on voyait souvent l'erreur moyenne ne pas toujours converger vers 0 et la taille moyenne toujours converger vers une valeur non nulle, il se passe l'inverse : l'erreur moyenne est nulle (très bon point) mais la taille moyenne diverge systématiquement. Il est donc impensable de choisir cette méthode pour agréger des ensembles de prédiction.

6 Conclusion

Le modèle de machine learning ResNet-50 a eu une précision de 30,5% sur la pire classe de ImageNet. En partant de ce même modèle, tout au long du stage, j'ai travaillé avec la pire classe pour utiliser la prédiction conforme et tester différentes méthodes d'agrégation afin de trouver une méthode permettant de renvoyer un ensemble Γ de taille idéalement égale à 1 avec une erreur moyenne faible et un bon score de conformité.

La plupart des méthodes présentées ici étaient toutes basées sur le vote majoritaire – un moyen efficace de choisir une classe – seul le seuil changeait. En observant l'évolution de l'erreur et la taille moyenne, ainsi que le score de conformité selon le nombre d'agents, j'ai estimé que, pour cette base de données et pour ce modèle-là, les meilleurs paramètres sont $\tau = 0,85$, si on choisit le vote majoritaire généralisé, et K compris entre 10 et 20, valable aussi pour le vote majoritaire avec pour seuil le quantile.

Cependant, nous sommes partis d'une précision de 30,5% sur la classe la moins performante avec une classification élémentaire, pour obtenir des ensembles de prédiction dont la précision moyenne va jusqu'à 80%. Cette augmentation démontre la force de la prédiction conforme à faire des prédictions plus fiables et dignes de confiance, surtout avec des performances initialement faibles et notamment dans un contexte où la précision des résultats ne laisse aucun doute sur des prises de décision opérationnelles. Pour résumer, en plus de rendre les modèles plus performants, elle rend les prédictions plus sûres et exploitables.

Références

- [1] Imagenet. URL <https://www.image-net.org/about.php>.
- [2] Pl@ntnet. URL <https://plantnet.org>.
- [3] Classification des modèles selon la précision top 1 pour la base imagenet. URL https://paperswithcode.com/sota/image-classification-on-imagenet?tag_filter=171.
- [4] P. J. Boland, H. Singh, and B. Cukic. Stochastic orders in partition and random testing of software. *Journal of Applied Probability*, 39(3) :555–565, 2022.
- [5] M. Gasparin and A. Ramdas. Merging uncertainty sets via majority vote, 2024. URL <https://arxiv.org/abs/2401.09379>.
- [6] W. Tang and F. Tang. The Poisson Binomial Distribution— Old & New. *Statistical Science*, 38(1) :108 – 119, 2023. doi : 10.1214/22-STS852. URL <https://doi.org/10.1214/22-STS852>.
- [7] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*, volume 29. Springer, 2005.