Projet Deep Learning - Classificateur générateur

Wayan Crain ,Yanis Aoudjit, Arnaud Chéridi

Mars 2025

Introduction

Ce projet s'inscrit dans le cadre du module de Deep Learning du Master de Mathématiques Appliquées. Il vise à explorer l'interprétabilité des réseaux de neurones à travers une approche originale : l'exploitation d'un classificateur pour générer des images modifiées de manière contrôlée.

Plus précisément, l'objectif est d'utiliser un réseau neuronal entraîné à classifier des images (ici, des chiffres manuscrits du jeu de données MNIST) pour modifier une image donnée de façon à ce qu'elle soit reconnue comme appartenant à une autre classe choisie. L'idée n'est donc pas de partir d'un bruit aléatoire, mais bien d'une image réelle, que l'on transforme progressivement afin de maximiser la probabilité d'une classe cible différente de la classe initiale.

Ce processus met en lumière les caractéristiques visuelles internes apprises par le modèle pour discriminer les classes, et permet de mieux comprendre ses zones de sensibilité. Ce travail s'inscrit dans une perspective d'analyse des réseaux profonds et de visualisation de leurs comportements.

1 Entraînement et étude du classificateur CNN

Dans un premier temps, nous avons conçu et entraîné un modèle de classification basé sur un réseau de neurones convolutif (CNN), destiné à reconnaître les chiffres manuscrits du jeu de données MNIST. L'objectif était d'obtenir un modèle performant sur cette tâche, afin de l'utiliser par la suite dans un processus d'optimisation inversée.

1.1 Structure générale du modèle

Le modèle repose sur une architecture CNN classique, adaptée aux images de petite taille (28×28 pixels, niveaux de gris). Il est composé de deux blocs de convolution suivis de max-pooling, puis de deux couches entièrement connectées (denses), dont la dernière applique une activation softmax sur les 10 classes (chiffres de 0 à 9).

1.1.1 Détail des couches

- **Conv2D** : applique 32 filtres 3×3 à l'image d'entrée. Ces filtres extraient des motifs locaux (bords, textures, etc.) et produisent une sortie de dimension $26 \times 26 \times 32$.
- **MaxPooling2D** : réduit la taille spatiale à 13×13 en prenant le maximum sur des blocs 2×2.
- **Conv2D_1** : applique 64 filtres 3×3 supplémentaires, extrayant des motifs plus complexes. La sortie devient $11 \times 11 \times 64$.
- **MaxPooling2D_1** : réduit à $5 \times 5 \times 64$.
- **Flatten** : transforme le tenseur 3D ($5 \times 5 \times 64$) en un vecteur plat de taille 1600.
- **Dense** : couche entièrement connectée avec 128 neurones, suivie d'une activation ReLU. Elle applique une transformation linéaire $z = Wx + b$, suivie de :

$$\sigma(z_i) = \max(0, z_i)$$

- **Dense_1** : dernière couche avec 10 neurones et activation softmax, qui transforme les scores en probabilités :

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{10} e^{z_j}}$$

1.1.2 Dimensions des données

Chaque image MNIST est représentée dans un espace $R^{28\times28\times1}$, soit une image 2D avec un seul canal. Le CNN traite ces images en ajoutant progressivement des filtres pour construire des représentations de plus haut niveau.

1.1.3 Nombre de paramètres

Le modèle comprend environ **225 000 paramètres**, répartis comme suit :

- Conv2D : $(3 \times 3) \times 1 \times 32 + 32 = 320$
- Conv2D_1 : $(3 \times 3) \times 32 \times 64 + 64 = 18\,496$
- Dense : $(1600 + 1) \times 128 = 204\,928$
- Dense_1 : $(128 + 1) \times 10 = 1\,290$

1.1.4 Fonctionnement global

Le modèle suit le pipeline suivant :

1. Extraction des caractéristiques par convolution et max-pooling.
2. Réduction de la taille spatiale tout en augmentant la profondeur des canaux (de 1 à 64).
3. Mise à plat des cartes de caractéristiques.
4. Classification via des couches pleinement connectées.
5. Prédiction finale par softmax sur les 10 classes.

1.2 Étude d'ablation

Dans cette seconde partie, nous avons mené une étude d'ablation sur l'architecture initiale du classificateur CNN. L'objectif était de comprendre le rôle et l'importance relative de chaque couche dans la performance globale du modèle.

L'ablation consiste à supprimer ou modifier certaines parties du réseau (par exemple des couches de convolution, de pooling, ou des couches denses) afin d'évaluer leur impact sur la capacité du modèle à apprendre et généraliser.

Nous avons considéré plusieurs variantes du modèle original, en testant notamment :

- La suppression de la seconde couche `Conv2D` (réduction de la profondeur d'extraction de features).
- La suppression du `MaxPooling2D` intermédiaire (conservation d'une plus grande résolution).
- La réduction de la taille de la couche dense (passage de 128 à 64 neurones).

Chacune de ces modifications a été testée indépendamment pour isoler son effet sur la performance, évaluée par l'exactitude (accuracy) sur le jeu de test MNIST.

Cette approche permet de mieux comprendre la contribution individuelle des composants du réseau à sa robustesse, et d'envisager des allègements futurs de l'architecture tout en maintenant des performances acceptables.

Lors de nos différentes simulations, nous n'avons pas obtenu de modèle avec une différence significative de résultats.

Dans un contexte où le nombre de paramètres serait important, nous aurions privilégié le modèle 3 ou le modèle 4 :

- **Modèle 3** : même architecture que le modèle initial, mais avec un nombre de filtres réduit dans les couches de convolution, ce qui diminue la complexité tout en conservant une extraction hiérarchique de caractéristiques.
- **Modèle 4** : architecture similaire, mais la première couche dense de 128 neurones est supprimée, entraînant une réduction notable du nombre total de paramètres.

1.3 Comparaison du modèle CNN avec des Méthodes Classiques

fin d'évaluer l'efficacité de notre modèle de réseau de neurones convolutif (CNN), nous avons comparé ses performances à deux méthodes classiques de classification sur le jeu de données MNIST :

- Le classificateur k-plus proches voisins (k-NN)
- La régression logistique multinomiale

Ces deux modèles ont été entraînés sur les mêmes données d'entraînement que le CNN, sans extraction de caractéristiques manuelle. Contrairement au réseau convolutif, ces méthodes ne tirent pas parti de la structure spatiale des images.

1.3.1 Résultats de la comparaison

Modèle	**Exactitude sur test (%)**
CNN (modèle complet)	98.2
Régression logistique	92.4
k-NN (k=3)	96.7

1.3.2 Analyse

Les résultats montrent que le modèle CNN surpasse largement les méthodes classiques en termes de précision. Cette différence s'explique principalement par la capacité du réseau convolutif à extraire automatiquement des caractéristiques hiérarchiques pertinentes à partir des images, en tenant compte de leur structure spatiale.

Le classificateur k-NN, bien qu'il atteigne une performance correcte (96.7%), présente plusieurs limitations : il est sensible à la dimensionnalité, coûteux en mémoire, et ne généralise pas aussi bien qu'un modèle entraîné de manière paramétrique.

La régression logistique, quant à elle, repose sur une modélisation linéaire des classes, ce qui limite fortement sa capacité à capturer la complexité des formes manuscrites.

Ces comparaisons confirment l'intérêt des modèles profonds pour les tâches de vision par ordinateur, même sur des données relativement simples comme MNIST.

2 Analyse de la Confiance

Dans cette section, nous analysons la confiance du modèle à travers la distribution des valeurs de $m(x)$, définies comme la probabilité maximale prédite par le modèle pour chaque exemple x du jeu de validation :

$$m(x) = \max_{i \in \{0,\ldots,9\}} softmax_i(x)$$

2.1 Histogramme

L'histogramme obtenu représente la fréquence des différentes valeurs de $m(x)$. L'axe des abscisses correspond aux valeurs de confiance $m(x) \in [0, 1]$, et l'axe des ordonnées indique le nombre d'exemples pour lesquelles cette valeur a été observée.

2.1.1 Interprétation

L'histogramme montre que la grande majorité des valeurs de $m(x)$ sont fortement concentrées autour de 1. Cela signifie que, pour la plupart des échantillons, le modèle attribue une probabilité très élevée à la classe prédite.

Cette confiance élevée est cohérente avec le taux de classification correcte supérieur à 98 %, et suggère que le modèle ne se contente pas de deviner mais qu'il fait des prédictions nettes et assurées. Cela reflète une bonne capacité de séparation entre les classes.

2.1.2 Comparaison

- **CNN** : l'histogramme de $m(x)$ est très concentré autour de la valeur 1, indiquant une très forte confiance dans la quasi-totalité des prédictions. Cette concentration est cohérente avec un taux d'accuracy supérieur à 98 %, et reflète une excellente capacité du réseau à discriminer les classes.
- **Régression logistique** : bien que l'on observe également une concentration vers 1, la distribution est **plus étalée** sur l'intervalle [0.2, 0.9]. Cela montre que ce modèle attribue des probabilités plus nuancées aux exemples ambigus, traduisant une meilleure gestion de l'incertitude.
- **k-NN** : l'histogramme présente des pics marqués à certaines valeurs discrètes comme 0.4, 0.6, 0.8, ou 1.0. Cela s'explique par le fait que le score correspond à une proportion de votes parmi les k voisins, rendant les valeurs possibles limitées et non continues. Cette propriété rend le modèle moins flexible pour exprimer des incertitudes, en particulier dans les cas où les frontières entre classes sont proches ou floues (ex. un 1 et un 4 manuscrits ressemblants).

2.1.3 Conclusion

L'analyse de ces histogrammes met en lumière la capacité des modèles à exprimer leur confiance de manière plus ou moins granulaire. Le CNN combine une forte confiance et une très bonne séparation des classes. La régression logistique, bien que moins performante en accuracy, fournit des scores plus lissés et informatifs sur les exemples ambigus. Le k-NN, de par sa structure discrète, manque de finesse dans ce type d'évaluation.

L'analyse de la distribution des $m(x)$ confirme ainsi la robustesse et la fiabilité du classificateur entraîné (CNN). Sa capacité à faire des prédictions très confiantes, tout en maintenant un haut niveau de précision, en fait un modèle performant pour cette tâche. Ce type d'analyse peut également servir de point de départ à des travaux plus approfondis sur la calibration des probabilités ou sur la détection des exemples incertains dans un contexte d'apprentissage actif ou semi-supervisé.

2.2 Analyse quantile des scores de confiance

Pour approfondir l'analyse de la confiance des modèles dans leurs prédictions, nous avons calculé les quantiles des scores $m(x)$ pour les trois classificateurs étudiés : CNN, k-NN, et régression logistique. Ces quantiles permettent de caractériser la répartition des valeurs de confiance sur l'ensemble des données de test.

2.2.1 Quantiles pour le modèle CNN

Les quantiles calculés pour le modèle CNN montrent une extrême concentration des scores autour de 1.0. On observe notamment :

- $Q_{0.05} = 0.9949$
- $Q_{0.25} = 1.0$
- $Q_{0.50} = 1.0$

- $Q_{0.75} = 1.0$

Cela signifie que plus de 75 % des exemples sont classés avec une confiance maximale ($m(x) = 1.0$), et même les 5 % les moins confiants restent très proches de 1. Ce comportement témoigne de la **netteté extrême des prédictions** du CNN, et d'une incertitude très faible, même pour les exemples les plus ambigus.

2.2.2 Quantiles pour le modèle k-NN

Les quantiles du modèle k-NN reflètent la nature discrète de son mécanisme de décision :

- $Q_{0.05} = 0.8$
- $Q_{0.10} = 1.0$
- $Q_{0.25} = 1.0$

La majorité des exemples sont donc prédits avec une confiance maximale, mais une proportion non négligeable d'exemples plus ambigus retombe à des niveaux significativement plus faibles, comme 0.8. Cette structure en paliers résulte du fonctionnement du vote majoritaire basé sur les voisins, qui limite la finesse de l'échelle de confiance.

2.2.3 Quantiles pour la régression logistique

La régression logistique présente une répartition plus étalée et continue :

- $Q_{0.05} = 0.5748$
- $Q_{0.25} = 0.8946$
- $Q_{0.50} = 0.9757$
- $Q_{0.75} = 0.9956$

Ces résultats montrent que la régression logistique attribue des probabilités intermédiaires plus nuancées aux exemples ambigus. Le modèle conserve une forte confiance sur la majorité des prédictions, mais sait aussi exprimer son incertitude dans les cas limites.

2.2.4 Conclusion

Les trois modèles ont été comparés sur leur capacité à prédire correctement les chiffres manuscrits de MNIST, ainsi que sur la répartition de leur confiance via les quantiles de $m(x)$.

- **CNN** : très grande précision (99.00 %) et prédictions extrêmement confiantes. Plus de 75 % des exemples obtiennent $m(x) = 1.0$.
- **k-NN** : précision correcte (96.88 %) mais échelle de confiance discrète, avec moins de nuances dans les cas ambigus.
- **Régression logistique** : précision plus faible (92.58 %) mais scores $m(x)$ répartis de manière plus continue et informative.

Le modèle CNN se distingue donc comme le plus performant, alliant une excellente précision et une confiance très nette dans ses prédictions. Sa capacité à extraire des caractéristiques complexes et à modéliser les frontières non linéaires en fait un outil particulièrement bien adapté à ce type de tâche de classification d'images.

3 Optimisation d'une image pour maximiser une classe cible

Nous cherchons ici à générer une image artificielle appartenant à une classe cible i en optimisant directement les pixels d'une image x afin de maximiser la sortie $p_i(x)$ du modèle :

$$\max_x p_i(x),$$

où $p_i(x)$ est la probabilité associée à la classe i dans le vecteur de sortie du modèle : $p(x) = model(x)$.

3.1 Gradient Ascent et Clipping

Cette optimisation est effectuée via une méthode de **gradient ascent** sur les pixels de l'image :

$$x \leftarrow x + \eta \cdot \nabla_x p_i(x),$$

où :

- η est le taux d'apprentissage,
- $\nabla_x p_i(x)$ est le gradient de la probabilité cible par rapport à l'image.

À chaque itération, une étape de **clipping** est appliquée pour garantir que les pixels restent dans l'intervalle $[0, 1]$:

$$x \leftarrow clip(x, 0, 1),$$

c'est-à-dire que toute valeur inférieure à 0 est remplacée par 0, et toute valeur supérieure à 1 est ramenée à 1, assurant une image valide.

Approche Naïve et ses Limites

Une première approche intuitive consiste à définir la fonction de perte comme l'opposé de la probabilité cible :

$$Loss(x) = -p_i(x),$$

et donc le gradient est simplement :

$$\nabla_x Loss(x) = -\nabla_x p_i(x).$$

L'image est mise à jour selon :

$$x \leftarrow x - \eta \cdot \nabla_x p_i(x),$$

avec un critère d'arrêt basé sur la norme du gradient :

$$\|\nabla_x p_i(x)\| < \epsilon.$$

Cependant, cette approche s'est révélée inefficace dans notre cas. Les gradients calculés étaient très faibles (proches de zéro), ce qui entraînait une stagnation de l'optimisation.

3.2 Approche Améliorée : Perte Logarithmique et Normalisation

Pour améliorer la stabilité et la convergence, nous avons redéfini la fonction de perte comme une **perte logarithmique** :

$$Loss(x) = -\log(p_i(x) + \epsilon),$$

avec $\epsilon = 10^{-10}$ pour éviter les logaritmes de zéro.

Le gradient est alors normalisé pour éviter les mises à jour trop faibles ou trop violentes :

$$\tilde{\nabla}_x = \frac{\nabla_x Loss(x)}{\|\nabla_x Loss(x)\| + \epsilon},$$

et la mise à jour devient :

$$x \leftarrow x - \eta \cdot \tilde{\nabla}_x.$$

Cette stratégie garantit des mises à jour directionnelles bien conditionnées, améliorant la convergence même lorsque $p_i(x)$ est initialement très faible.

3.3 Visualisation : Effet de l'Initialisation et du Taux d'Apprentissage

Nous avons testé cette optimisation avec différentes conditions initiales (images aléatoires) et différents taux d'apprentissage η.

- Un taux d'apprentissage trop élevé provoque des sauts brutaux, une instabilité dans la montée du gradient, et empêche parfois la convergence vers une image plausible. Le processus oscille autour du maximum ou diverge.
- Un taux d'apprentissage trop faible entraîne des mises à jour trop lentes : l'optimisation progresse très peu à chaque itération, ce qui allonge considérablement le temps nécessaire pour atteindre une probabilité satisfaisante.
- Une initialisation aléatoire différente produit des prototypes visuellement différents, mais convergeant tous vers des structures typiques de la classe cible. Cela montre la capacité du modèle à guider l'optimisation malgré un point de départ non informé.

3.4 Conclusion

Cette expérience montre qu'il est possible d'utiliser un classificateur entraîné pour générer des images représentatives de classes cibles, en optimisant directement l'entrée. La stabilité de l'optimisation repose fortement sur le choix de la fonction de perte, la normalisation du gradient, et le taux d'apprentissage. Cette approche illustre de manière concrète l'interprétabilité par visualisation inverse, et la structure interne du modèle.

4 Optimisation d'une Image via k-NN

4.1 Maximisation de la probabilité de classe

Dans le cas du réseau de neurones, la fonction $p_i(x)$ désignait directement la probabilité prédite pour la classe i par le modèle.

Pour le k-NN, nous définissons une pseudo-probabilité $p_i(x)$ comme la fraction de voisins (parmi les k plus proches voisins de x) appartenant à la classe i :

$$p_i(x) = \frac{nombredevoisinsdontlaclasseesti}{k}$$

Notre objectif est donc le même :

$$\max_x p_i(x)$$

4.2 Mise à jour de x

Contrairement au CNN, $p_i(x)$ n'est pas différentiable dans le cas du k-NN, car il dépend de la structure de voisinage — une opération discrète. Toutefois, nous pouvons approximer une direction d'amélioration à chaque itération.

Étapes de mise à jour :

1. **Calcul des k plus proches voisins** de x, notés $X^{(n_1)}, \ldots, X^{(n_k)}$.
2. **Estimation de** $p_i(x)$ comme la proportion de voisins appartenant à la classe i.
3. **Approximation d'un "gradient"** : En se basant sur les vecteurs $(x - X^{(n_j)})$, on construit une direction qui éloigne x des voisins de mauvaise classe et le rapproche de ceux de la classe cible.
4. **Mise à jour par déplacement** :

$$x \leftarrow x - \eta \cdot direction$$

Critères d'arrêt :

- $p_i(x) \geq$ seuil fixé (ex. 0.6)
- Nombre maximal d'itérations
- Masquage aléatoire : à chaque itération, seules certaines composantes de x sont modifiées, afin d'éviter des ajustements trop brutaux ou incohérents.

4.3 Résultats et limites de l'approche k-NN

Les résultats de l'optimisation par k-NN révèlent rapidement les limites de cette méthode. Bien que $p_i(x)$ puisse atteindre des valeurs proches de 0.6 pour la classe cible, l'image générée reste floue, difficilement reconnaissable, et n'exprime pas clairement les traits visuels de la classe choisie.

Cette difficulté s'explique par l'absence de gradients exacts, remplacés ici par des estimations basées sur des différences de voisinage. Ces directions ne permettent pas de convergence cohérente, produisant des modifications dispersées, voire contradictoires.

Par ailleurs, cette méthode est coûteuse : chaque mise à jour de x nécessite de recalculer la distance à l'ensemble des points d'entraînement, ce qui ralentit fortement l'optimisation.

4.4 Comparaison avec le CNN

À l'inverse, l'optimisation via CNN permet de générer une image beaucoup plus structurée, bien que la probabilité finale $p_i(x)$ soit parfois inférieure (ex. 0.5065 contre 0.60 pour k-NN). La présence de gradients précis permet au CNN d'effectuer des mises à jour plus cohérentes et ciblées, donnant lieu à une image plus fidèle à la classe visée.

Le CNN est également plus rapide à exécuter : une fois le modèle entraîné, aucune comparaison directe avec les données d'apprentissage n'est nécessaire.

5. Conclusion sur cette comparaison

Ces expériences confirment les limites du k-NN pour la génération inverse : bien que la pseudo-probabilité puisse être maximisée, le manque de différentiabilité empêche une optimisation efficace. Le CNN reste plus adapté, à la fois en termes de qualité visuelle, de stabilité, et de performance de calcul.

5 Conclusion et Perspectives

Nous avons mené à bien les différentes étapes de ce projet, en atteignant les objectifs fixés. Cela inclut l'entraînement d'un classificateur CNN performant, la comparaison avec des méthodes classiques (k-NN, régression logistique), l'analyse fine de la confiance du modèle via $m(x)$, et enfin la génération inverse d'images à partir d'une classe cible.

Ce travail a permis de poser des bases solides pour la compréhension du fonctionnement interne des modèles de classification, tout en illustrant l'intérêt de l'interprétation par optimisation directe.

5.1 Pistes d'amélioration

Plusieurs prolongements sont envisageables :

- **Étendre la comparaison** à d'autres modèles : SVM, arbres de décision ou forêts aléatoires, pour explorer d'autres approches non neuronales.
- **Tester sur d'autres jeux de données** (comme CIFAR-10 ou Fashion-MNIST), afin d'évaluer la généralisation des méthodes étudiées à des images plus complexes ou de nature différente.
- **Évaluer la robustesse** des modèles face à du bruit, des perturbations ou des adversaires simples, pour analyser leur stabilité hors distribution.

- **Adapter ces approches à des cas concrets**, notamment en milieu universitaire : reconnaissance d'écriture manuscrite d'enseignants, lecture automatique de copies, ou interprétation d'annotations manuscrites au tableau.

Ce projet démontre l'intérêt croissant de l'apprentissage profond pour des tâches de classification d'image, mais aussi la richesse des méthodes d'interprétation associées.