Comparaison d'Algorithmes de Réseaux de Neurones Artificiels en Classification d'Images : CNN, RNN et DNN en exemple.

Mehdi Mansour *

ABSTRACT

Les algorithmes d'apprentissage profond, dont les réseaux de neurones convolutifs (CNN), récurrents (RNN) et denses (DNN), ont révolutionné la classification automatique d'images. Cette étude vise à comparer les performances relatives de ces trois architectures sur des tâches de complexité variable.

Des modèles CNN, RNN et DNN ont été entraînés sur les ensembles d'images MNIST et CIFAR-10. Leurs métriques de précision ont été analysées, ainsi que leur vitesse d'entraînement.

Les résultats montrent la supériorité des CNN pour la classification d'images, grâce à leur capacité à extraire des caractéristiques visuelles pertinentes. Ils surpassent nettement les DNN sur les images complexes. Les RNN peinent à capturer toutes les dépendances spatiales des images CIFAR-10. Les DNN restent attractifs pour leur rapidité d'entraînement sur GPU.

Cette étude fournit des indications précieuses sur le choix de l'architecture optimale selon le type de données d'images. Elle met en évidence le potentiel des CNN pour la plupart des applications pratiques de classification d'images.

Keywords: Computer Vision, Deep Learning, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), DNN, MNIST et CIFAR.

INTRODUCTION

Dans le domaine en évolution rapide de l'intelligence artificielle (IA), la classification d'images représente un défi technique captivant et cristallise des enjeux majeurs pour le futur. Cette technologie trouve des applications dans une multitude de domaines, allant de la reconnaissance faciale à la détection de maladies dans les images médicales. Au cœur de ces avancées se trouvent des algorithmes de classification d'image sophistiqués, dont les Dense Neural Networks (DNN), les Convolutional Neural Networks (CNN) et les Recurrent Neural Networks (RNN). Ces modèles d'apprentissage profond ont révolutionné notre capacité à interpréter et à traiter les images numériques.

* L'auteur M. Mansour est étudiant à l'université Lyon 2. Il est en reconversion professionnelle après une expérience dans le secteur des services. Il prépare un master en informatique en vue de se spécialiser en machine Learning et intelligence artificielle. Email: mehdi.mansour@univ-lyon2.fr

Le but de cet article est de comparer la performance et la consommation de ressources, de ces trois types d'algorithmes en utilisant deux sets de données distincts : un rudimentaire de faible dimension, et un autre d'une plus grande dimensionnalité et richesse en caractéristiques. Cette comparaison vise à explorer comment la taille et la complexité des données influencent la performance de chaque algorithme. En fournissant un aperçu détaillé de chacun de ces modèles, l'article cherche à éclairer les forces et les faiblesses inhérentes à chaque approche, offrant ainsi une perspective sur la manière dont ces algorithmes peuvent être optimisés et appliqués dans divers scénarios pratiques.

En abordant cette étude comparative, nous espérons non seulement enrichir la compréhension des lecteurs sur les mécanismes sous-jacents des DNN, CNN, et RNN, mais aussi fournir des insights précieux sur le choix de l'algorithme le plus adapté en fonction des caractéristiques spécifiques des données disponibles. Cette démarche est d'autant plus pertinente dans un contexte où la quantité et la variété des données d'image continuent de croître, posant ainsi de nouveaux défis et opportunités dans le domaine de l'IA.

II. FONCTIONNEMENT DES DNN, RNN ET CNN

A. Bases Communes aux Réseaux de Neurones

Un neurone artificiel (1) est une modélisation mathématique de la transmission d'influx, inspiré du fonctionnement d'un neurone biologique. Il constitue l'unité de base des réseaux de neurones artificiels, qui sont un système composé de nombreux neurones artificiels organisés en couches et reliés entre eux.

Un réseau de neurones artificiels (2) exploite la capacité d'un neurone artificiel à transformer un signal d'entrée en une sortie, en interconnectant de multiples neurones organisés en couches successives. Chaque neurone reçoit des entrées provenant des neurones en amont, auxquelles il applique des poids synaptiques, qui sont ajustés pendant la phase d'apprentissage. Il calcule la somme pondérée de ses entrées et y applique une fonction d'activation non linéaire pour déterminer sa sortie, qui sera l'entrée des neurones de la couche suivante :

$$\mathbf{sortie} = \mathrm{f}\left(\sum_{i=0}^{n} a_i x_i + \boldsymbol{biais}\right)$$

Dans l'équation ci-dessus , f représente la fonction d'activation (3) d'un neurone, il s'agit d'une transformation nécessaire. En effet, sans fonction d'activation, un réseau de neurones ne ferait que des combinaisons linéaires des entrées, ce qui limiterait grandement sa capacité à modéliser des problèmes complexes et sa capacité d'apprentissage. La non-linéarité apportée par la fonction d'activation permet à un réseau de neurones doté de suffisamment de couches et de données d'apprentissage d'approximer n'importe quelle fonction. En écrasant généralement la sortie entre 0 et 1 (sigmoïde, tanh) ou entre 0 et l'infini (ReLU), les fonctions d'activation modulent également l'amplitude de la sortie, évitant ainsi l'explosion de ces valeurs durant la phase d'apprentissage. De plus, Des fonctions comme la sigmoïde transforment la sortie agrégée en une valeur qui peut être interprétée comme une probabilité ou une décision binaire (0 ou 1 plutôt que -5 ou 5 par exemple). Cela permet au réseau d'apprendre à effectuer des décisions complexes.

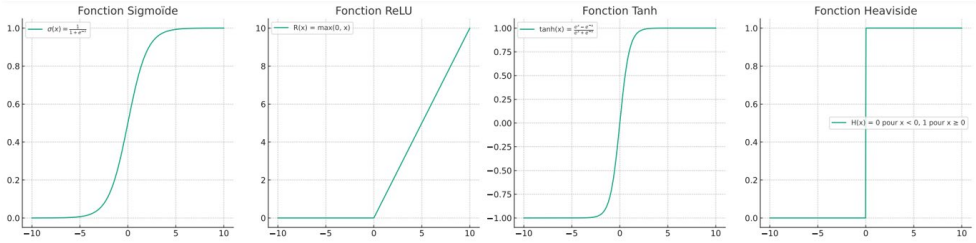


Figure 1: Fonctions d'activation usuelles

La phase d'apprentissage d'un réseau de neurones artificiels consiste à ajuster de façon itérative les poids et biais synaptiques, pour diminuer l'erreur entre les valeurs qu'il prédit et les sorties réelles observées. Cette quantité, l'erreur, est évaluée à l'aide d'une fonction de coût (4), également appelée fonction de perte. Les valeurs prédites étant elles fonction de tous les poids et biais du réseau, il va sans dire que l'erreur l'est aussi, et que les poids et biais deviennent dès lors des variables de la fonction perte.

L'une des fonctions de perte les plus utilisées pour l'apprentissage de classification multi classe est l'entropie croisée catégorielle, dont l'expression pour une observation est :

$$H(y, pred) = -\sum_{i=1}^{C} y_i \log(pred_i)$$

Où :

y est le vecteur de vérité terrain (one-hot encoded) pour une classe donnée.

pred est le vecteur de probabilités prédites.

C est le nombre total de classes.

y_i est le i-ème élément de y, et pred_i est le i-ème élément de pred. C'est une valeur comprise entre 0 et 1 car il s'agit d'une probabilité. Son logarithme naturel est par conséquent négatif.

Dans cette expression, seul le terme y_i valant 1 sera retenu, les autres étant égaux à 0. Ce qui en fait une valeur positive d'autant plus grande que la prédiction s'éloigne de la vérité. La fonction logarithme est introduite pour pénaliser fortement une prédiction confiante et incorrecte. En l'occurrence une prédiction donnant une probabilité proche de 0 à la vraie classe observée sur le terrain, aura un logarithme infiniment grand et négatif, il en résultera une erreur infiniment grande.

Dans la pratique, La moyenne de cette erreur est calculée sur un ensemble d'observations, avant d'être utilisée pour l'ajustement des poids et biais des neurones du réseau.

Après avoir calculé l'erreur à l'aide de la fonction de perte, la question devient : comment ajuster les poids synaptiques et les biais de façon optimale pour réduire cette erreur lors de la prochaine itération ?

La descente de gradient (5) est la technique communément utilisé à cet effet. Elle consiste à calculer la dérivée partielle de la fonction d'erreur par rapport à chaque paramètre. Cela donne les gradients de la fonction par rapport à ses paramètres. C'est-à-dire, pour chaque paramètre, la direction dans laquelle la fonction augmente quand ce paramètre augmente. L'ajustement consistera ensuite à mettre à jour les paramètre dans le sens inverse du gradient, de façon à descendre au fond du "creux" de la fonction, jusqu'à atteindre un point plus bas, un minimum.

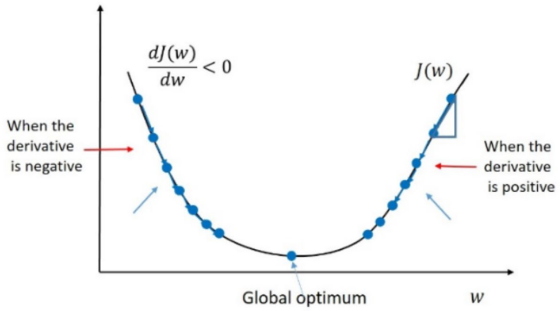


Figure 2: illustration de la descente de gradient sur une fonction de perte convexe

Calculer ce gradient sur une seule couche est évident. Néanmoins, dans un réseau de plusieurs couches, l'erreur est calculée au niveau de la sortie finale, c'est-à-dire que c'est une fonction composée des fonctions des couches précédentes.

C'est là qu'intervient l'algorithme de rétropropagation de l'erreur, il consiste à appliquer la règle de la dérivation en chaine des fonctions composées :

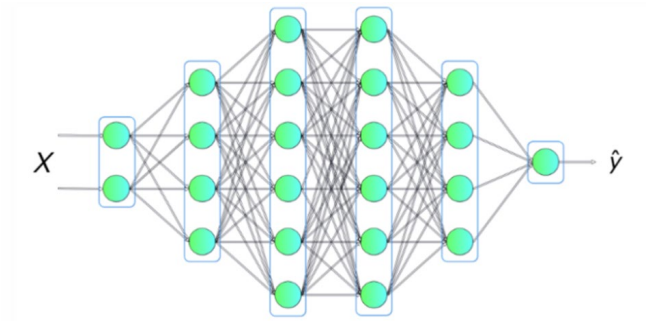
$$\frac{dw}{du} = \frac{dw}{dv} \cdot \frac{dv}{du}$$

Pour propager l'erreur en arrière, en remontant les couches parcourues initialement lors de la genèse de la sortie . Cela permet de retrouver mathématiquement la responsabilité de chaque paramètre dans l'erreur finale quel que soit le niveau de la couche à laquelle il appartient.

Les réseaux de neurones partagent ces principes généraux de fonctionnement. Ils diffèrent néanmoins par leurs architectures et la façon dont la sortie finale se construit à travers les couches. Dans notre travail nous nous sommes intéressés à trois architectures majeures : les réseaux entièrement connectés, les réseaux convolutifs, et les réseaux récurrents.

B. Les Réseaux Denses, ou Entièrement Connectés, DNN

Aussi appelés les FCNN, fully connected networks. Ce sont une architecture de plusieurs couches de neurones, ou chaque neurone d'une couche est connecté à tous les neurones de la couche suivante:



La transmission de l'influx dans ce type de réseau se fait exclusivement d'une couche à la couche suivante sans boucle possible.

Les DNN ne requièrent aucune hypothèse particulière sur la structure des données en entrée. Ils sont donc très largement applicables.

C. Les Réseaux de Neurones Convolutifs, CNN

Les CNN (6) ont été conçus pour des données spatiales, ils adoptent une architecture enchainant des couches d'extraction de caractéristiques, et des couches de prédiction de type entièrement connecté :

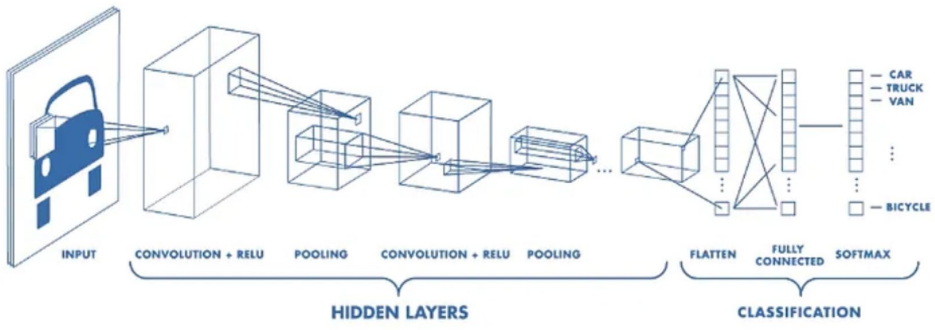


Figure 4: architecture type d'un CNN

Dans la couche d'extraction de caractéristiques, on alterne des couches de convolution et des couches de pooling.

La couche de convolution est dotée d'une matrice de poids, qui va balayer l'image d'entrée. A chaque déplacement, le produit scalaire entre la matrice de poids et la zone couverte est calculé, avant d'y appliquer une fonction d'activation. Une convolution complète produit une matrice de caractéristiques ou chaque valeur découle de l'opération précédemment décrite.

Cette carte de caractéristiques, subit ensuite une opération qui vise à réduire la dimensionnalité en perdant le minimum d'information. C'est la couche de pooling, elle consiste à parcourir la carte de caractéristiques par fenêtre et ne garder qu'une valeur signifiante de chaque fenêtre parcourue. Le plus souvent l'opération est un Max Pooling qui garde la valeur maximale de l'ensemble à chaque pas.

Ces deux opérations sont répétées, produisant une représentation de plus en plus abstraite de l'image d'entrée, mais ayant capté les caractéristiques cruciales pour distinguer les différentes classes à discerner.

Cette représentation abstraite est ensuite linéarisée et donnée en entrée à un réseau entièrement connecté qui s'occupe de calculer la probabilité d'appartenir à chaque classe.

D. Les Réseaux de Neurones Récurrents, RNN

Dans un RNN (7), les neurones peuvent envoyer des informations non seulement vers l'avant (vers la sortie), mais aussi en arrière vers eux-mêmes. Cela signifie que l'information de l'état actuel du réseau peut être passée au réseau lorsqu'il traite le prochain élément de la séquence.

Cette faculté confère une sorte de mémoire court terme qui permet de traiter les données reçues successivement comme des séquences et de détecter des liens entre la séquence en cours et les séquences passées.

Appliqué à une image, un RNN qui va la lire dans un certain ordre séquentiel, par exemple de gauche à droite et de haut en bas. À chaque pas spatial, une cellule du RNN reçoit en entrée la valeur de la séquence en cours, ainsi que son propre état interne résultant des séquences précédentes. Après avoir balayé toute l'image, le RNN a construit une représentation riche qui capture les motifs spatiaux dans l'image grâce aux dépendances locales explorées. Cette représentation finale peut être utilisée pour des tâches de classification.

L'intérêt des RNN est leur capacité implicite à modéliser des dépendances spatiales grâce à la récurrence, de façon similaire aux dépendances temporelles dans les séquences classiques. Leur mémoire interne leur permet d'apprendre des motifs spatiaux complexes.

III. PRESENTATION DES SETS DE DONNEES UTILISES

Dans cette section, nous présentons les deux ensembles de données utilisées dans notre étude, à savoir MNIST et CIFAR-10. Nous expliquons également les raisons pour lesquelles nous avons choisi ces ensembles de données et discutons de leurs caractéristiques qui peuvent apriori influencer les résultats.

A. MNIST

MNIST, Modified National Institute of Standards and Technology database (8), est un ensemble de données classique en vision par ordinateur. Il se compose de 70 000 images en niveaux de gris de chiffres manuscrits, chaque image étant de taille 28x28 pixels. Ces images sont réparties en 10 catégories, une pour chaque chiffre de 0 à 9. MNIST est largement utilisé comme point de départ pour les tâches de classification d'images, en particulier pour les réseaux de neurones. Nous avons choisi MNIST pour sa simplicité et sa capacité à servir de référence pour évaluer les performances de nos algorithmes. Les caractéristiques clés de MNIST incluent sa petite taille d'image, sa faible complexité et sa facilité de prétraitement.

B. CIFAR-10

CIFAR-10 (9) est un ensemble de données plus complexe, composé de 60 000 images couleur de 32x32 (x3) pixels répartis en 10 classes différentes. Chaque classe représente un type différent d'objet, tel que des avions, des voitures, des oiseaux, des chats, etc. Nous avons choisi CIFAR-10 pour sa nature plus réaliste et sa capacité à mettre nos algorithmes à l'épreuve face à des images plus variées et complexes. Les caractéristiques importantes de CIFAR-10 sont sa taille d'image plus grande, la couleur, et la diversité des objets représentés.

C. Choix des Datasets et Influence Potentielle sur les Résultats

Nous avons choisi d'utiliser MNIST, en raison de sa simplicité, il permet de réaliser une évaluation initiale de base des algorithmes. En revanche, CIFAR-10 offre un défi plus réaliste et nous permet d'analyser la capacité de généralisation de nos modèles face à des données plus

complexes.

Les caractéristiques des ensembles de données, telles que la taille de l'image, la complexité des images et le nombre de classes, exerceront une influence sur les performances de nos algorithmes. Les diversifier permet donc d'explorer plusieurs scénarii d'application pour évaluer la polyvalence et les limites de nos modèles.

Ainsi, le choix de MNIST et CIFAR-10 comme ensembles de données dans notre étude nous permet de couvrir un plus grand éventail de tests et de mieux comprendre les performances relatives de nos algorithmes RNN, CNN et denses dans divers contextes.

IV. METHODOLOGIE

A. Prétraitement des Données

Une étape préliminaire cruciale consiste à préparer nos ensembles de données pour une utilisation dans nos modèles de réseaux de neurones. Pour MNIST et CIFAR-10, nous avons commencé par normaliser les valeurs des pixels, les remettant ainsi à une échelle entre 0 et 1. Cette normalisation est souhaitable pour faciliter une convergence efficace de l'apprentissage. De plus, nous avons réalisé un encodage one-hot (10) des étiquettes, une étape essentielle pour transformer les étiquettes de classe en un format adapté à l'apprentissage supervisé par la librairie d'implémentation utilisée.

B. Configuration des Modèles

Dans cette étude, nous avons implémenté trois architectures choisies de réseaux de neurone, de façon adaptée à la classification d'images, mais avec des configurations distinctes pour assurer un nombre de paramètres comparable, d'environ 2.000.000 de paramètres (11) pour chaque modèle. Pour préserver ce nombre de paramètres, et le garder en critère de comparabilité, la profondeur du modèle et la taille des couches le composant peuvent varier.

B.1. Modèle CNN

Le modèle CNN, utilise une série de couches de convolution et de normalisation (12). Chaque couche de convolution, avec 256 filtres de taille 3x3, est suivie d'une normalisation par lots (Batch Normalization) pour améliorer la stabilité et l'efficacité de l'apprentissage. Les couches de MaxPooling2D, intégrées après les couches de convolution, servent à réduire la dimensionnalité des caractéristiques extraites. Après les étapes de convolution, le modèle aplatit les caractéristiques et les envoie à travers des couches entièrement connectées de tailles 128, 64, et finalement 10, cette dernière utilisant une activation softmax pour la classification multi-classes.

B.2. Modèle RNN

Pour le modèle RNN, nous avons opté pour des cellules LSTM (Long Short-Term Memory),

prêtes pour capter les dépendances séquentielles dans les données. Le réseau commence par une couche LSTM de 512 neurones, adaptée à la taille d'image en entrée. Une seconde couche LSTM de 256 neurones suit, avec des opérations de normalisation par lots intercalées entre elles pour une meilleure stabilité lors de l'apprentissage. En sortie, le modèle se compose de couches entièrement connectées de tailles 256, 128 et 10, avec la dernière couche utilisant également une activation softmax.

B.3. Modèle DNN

Le modèle DNN, commence par transformer les images en vecteurs via une couche d'aplatissement (Flatten). Il comprend ensuite plusieurs couches entièrement connectées: deux couches de 512 neurones, suivies de deux couches de 256 neurones, et enfin plusieurs couches de 128 et 64 neurones. Des couches de normalisation par lots sont intégrées entre les couches de neurones pour régulariser et stabiliser l'apprentissage. La couche de sortie, avec 10 neurones, utilise également l'activation softmax pour la classification finale.

C. Compilation et Entrainement

Nous avons compilé nos modèles en utilisant l'optimiseur Adam et la fonction de perte Categorical Cross-entropy (13). Les modèles ont ensuite été entraînés sur l'ensemble de données pendant 10 époques, avec une taille de lot de 64 et un ratio de 20% pour la validation croisée.

D. Evaluation

L'évaluation des performances de nos modèles repose sur les métriques clés usuelles telles que la perte (loss), l'exactitude (accuracy) ou la précision, calculées sur les données de validation et de test.

Pour une compréhension plus approfondie des performances de chaque modèle au fil de l'apprentissage, nous avons généré des courbes d'apprentissage, montrant l'évolution de ces métriques au cours des époques.

Aussi, pour les six modèles, des rapports de classification par cible ont été établis afin de prendre en compte les forces et faiblesses de chaque modèle en fonction de la classe à prédire.

Le temps nécessaire à chaque apprentissage est également pris en compte.

V. EXPERIMENTATION ET RESULTATS

A. Vitesse de l'Entrainement

La vitesse d'entraînement des modèles d'apprentissage profond est une composante critique de la recherche en intelligence artificielle, influant directement sur l'itération rapide des expériences et l'efficacité globale du développement des modèles. Dans notre étude, nous avons examiné la réactivité de nos trois architectures lors de séries d'essais sur plusieurs plateformes de calcul.

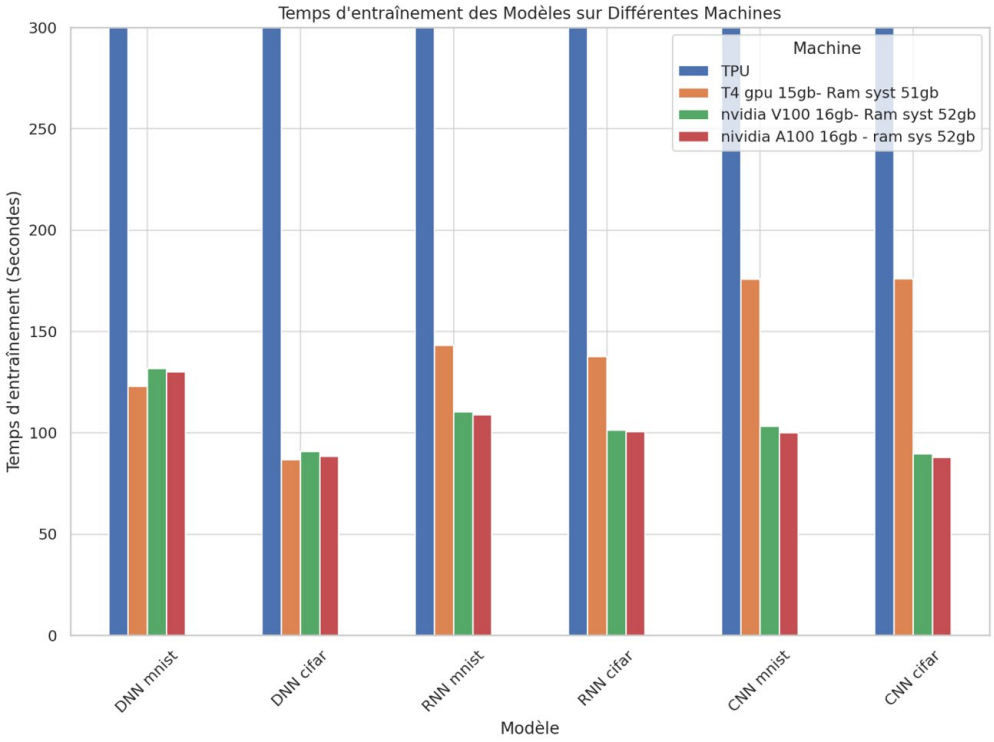


Figure 5: récapitulatif des temps d'entrainement par modèle et par plateforme.

L'Essai 1, réalisé sur un ordinateur personnel, a échoué à entraîner les modèles, mettant en exergue l'inadéquation des capacités de calcul standard pour les exigences intensives des réseaux de neurones profonds. Ce résultat souligne la nécessité d'infrastructures spécialisées pour l'entraînement de telles architectures.

Avec l'Essai 2 et l'introduction des Tensor Processing Units (TPUs) via Google Colab, tous les modèles ont pu être entraînés, mais avec des temps d'entraînement notables. Les architectures CNN se sont avérées être les plus gourmandes en temps, le modèle pour CIFAR atteignant 15868.20 secondes, tandis que les architectures RNN et DNN ont montré des performances plus rapides, notamment le modèle DNN pour CIFAR qui n'a nécessité que 390.14 secondes.

L'Essai 3 a impliqué des GPU T4 et a révélé une diminution considérable des temps d'entraînement pour tous les modèles. Les architectures DNN ont continué à montrer une efficacité supérieure, avec des temps inférieurs à ceux des modèles RNN et CNN pour les mêmes ensembles de données.

Les Essais 4 et 5 avec les GPU NVIDIA V100 et A100 ont fourni des résultats encore plus révélateurs. Les modèles CNN et RNN pour CIFAR ont vu leurs temps d'entraînement réduits à moins de 110 secondes sur les deux types de GPU. Cependant, ce sont les architectures DNN qui ont le plus bénéficié des capacités de calcul avancées de la A100, avec des temps d'entraînement raccourcis à 88.32 secondes pour CIFAR, indiquant plus de rapidité malgré l'aspect complètement connecté de ce type d'architecture.

B. Performance Globale

B.1 Performance Globale sur MNIST

Sur les données MNIST les trois architectures ont montré une performance très honorable dépassant les 97% d'exactitude sur les sets de validation.

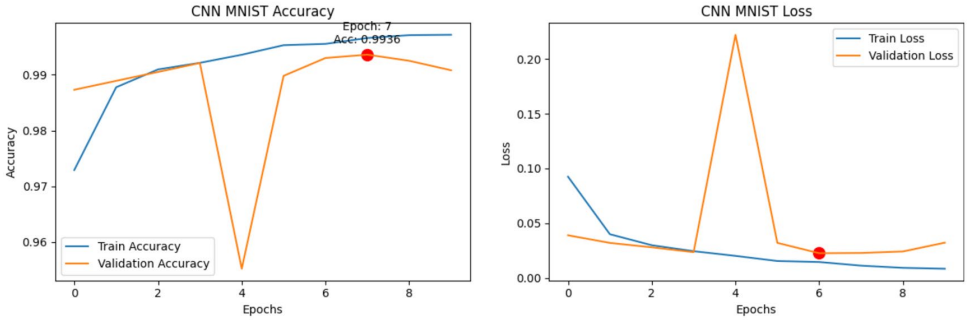


Figure 6: performance modèle CNN sur MNIST.

Le modèle CNN a la meilleure performance parmi ses concurrents, il l'atteint à l'epoch 7 avec une précision de validation (val_accuracy) de 0.9936. Bien que présentant une haute précision, le modèle a connu une fluctuation notable de la perte de validation (val_loss), indiquant une possible instabilité, en particulier à l'époque 5 où le val_loss a augmenté de manière significative.

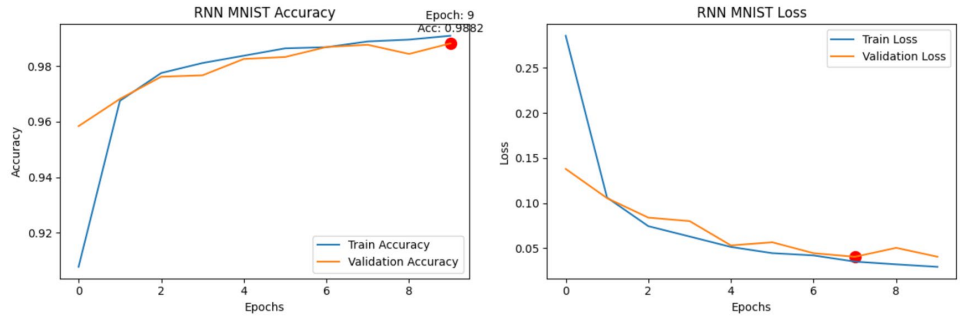


Figure 7: performance modèle RNN sur MNIST.

Le modèle RNN quant à lui, a montré une amélioration constante, atteignant son meilleur val_accuracy à l'epoch 9 avec 0.9882. Le meilleur val_loss a été enregistré à l'époque 7 mais semble pouvoir repartir à la baisse au-delà de dix epochs.

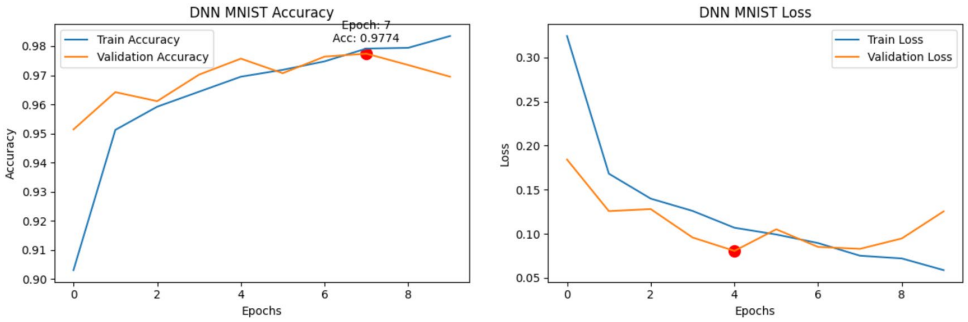


Figure 8: performance du modèle DNN sur MNIST.

Le modèle DNN a présenté une performance inférieure aux autres modèles, avec un meilleur val_accuracy de 0.9774 à l'epoch 7. Le meilleur val_loss a été observé à l'epoch 4.

B.2. Performance Globale sur CIFAR-10

Sur ces données qui sont plus complexes par la taille de l'image et la présence de trois canaux, Les modèles voient leurs performances respectives se dégrader de façon significative même si les prédictions restent largement au-dessus du hasard. Cette dégradation affecte particulièrement le modèle DNN.

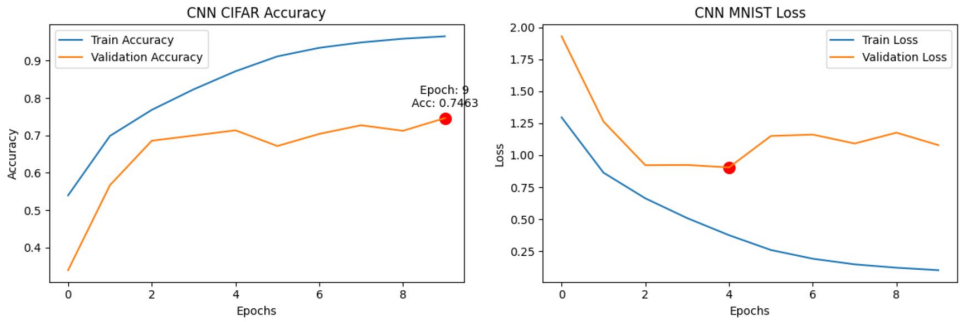


Figure 9: performance du modèle CNN sur CIFAR.

Le CNN a atteint son pic de performance à l'epoch 9 avec un val_accuracy de 0.7463.Il a montré une augmentation régulière de l'accuracy avec une certaine volatilité dans le val_loss.

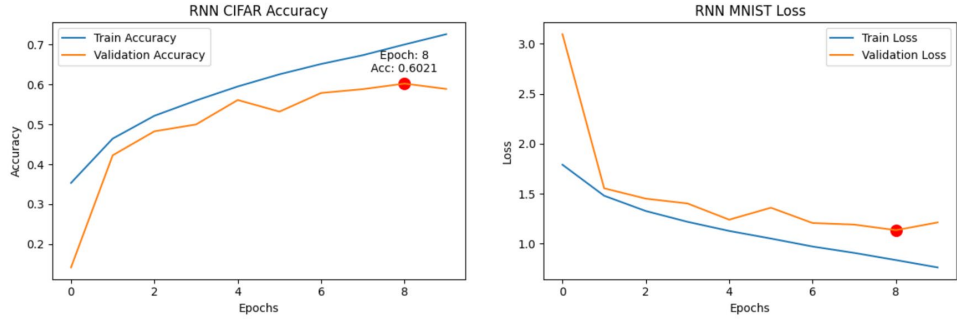


Figure 10: performance du modèle RNN sur CIFAR.

Le RNN a montré une amélioration graduelle, atteignant son meilleur val_accuracy de 0.6021 à l'epoch 8.

Quant au DNN, il a enregistré un meilleur val_accuracy de 0.4903 à l'epoch 7, ce qui est significativement inférieur aux autres modèles.

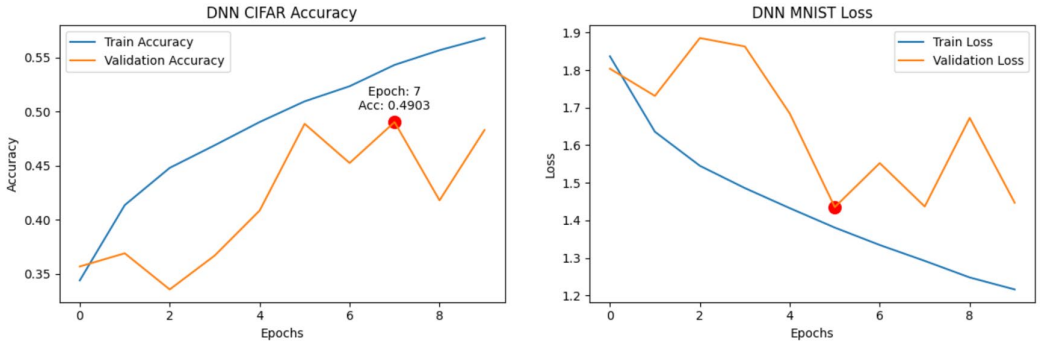


Figure 11: performance du modèle DNN sur CIFAR.

C. Performance par Classe Cible

Les résultats étant très proches sur MNIST, nous développons ici les résultats de chaque architecture, par classe de prédiction , sur les données CIFAR.

Rappelons que les classes cibles en CIFAR sont des images de la vie courante d'objets ou d'animaux du quotidien.

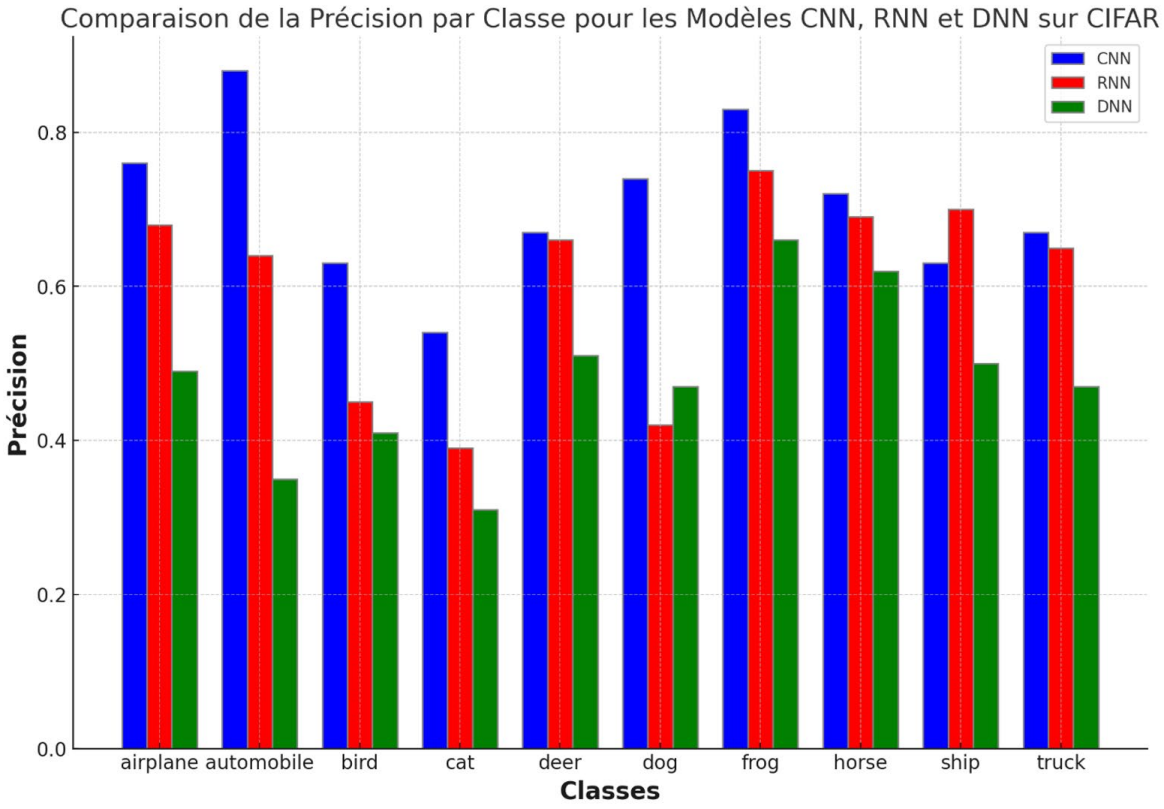


Figure 12: précision par classe CIFAR, des trois architectures.

Nous choisissons ici d'utiliser la métrique de la précision par classe, soit le ratio de prédictions correctes par rapport au nombre total de prédictions faites par le modèle sur une classe donnée. C'est une métrique simple à interpréter qui donne une bonne indication de la capacité du modèle à faire des prédictions justes.

Le CNN a excellé dans la classification des « automobile » , des « frog » et des « airplane ». Des performances plus faibles ont été observées pour les classes « cat » et « bird », ce qui peut indiquer des difficultés à gérer des formes et des textures plus complexes. Avec une précision moyenne de 0.71 et un score F1 de 0.69, le CNN a montré une performance globale solide sur l'ensemble de données CIFAR.

Le RNN a bien performé pour les classes « frog » et "ship", mais a montré des difficultés notables avec les classes « cat » et « dog ». Avec une précision moyenne de 0.60 et un score F1 moyen de 0.58, le RNN a démontré une efficacité moindre par rapport au CNN dans la classification des cibles.

Le DNN a enregistré des performances globalement inférieures, avec une précision moyenne de 0.48 et 0.44 pour le score F1 moyen. Il a eu des difficultés avec la majorité des classes, en particulier « cat » et « automobile ». Néanmoins, il surpasse confortablement les RNN dans la reconnaissance de « dog » . Et il a également un score assez proche des deux concurrents relativement, pour la reconnaissance de « Horse ».

VI. DISCUSSION

Cette étude comparative nous permet de tirer plusieurs enseignements sur les performances relatives des architectures de réseaux de neurones RNN, CNN et DNN pour la classification d'images.

Tout d'abord, les résultats sur l'ensemble de données MNIST montrent que les trois architectures sont capables d'atteindre une précision de classification très élevée, dépassant 97%, sur ces images simples de chiffres manuscrits. Le modèle CNN s'est montré légèrement supérieur, tandis que le DNN était un peu en retrait.

Cependant, avec l'ensemble de données CIFAR-10 contenant des images plus complexes, les performances se sont dégradées de façon marquée, en particulier pour le DNN. Le CNN est resté le plus performant, suivi du RNN, et du DNN fermant la marche.

L'analyse des précisions par classe cible sur CIFAR-10 donne un aperçu plus fin des forces et faiblesses relatives des architectures. Le CNN s'est montré le plus polyvalent, avec de bonnes performances sur la plupart des classes. Le RNN a eu plus de difficultés avec certaines classes comme "cat" et "dog". Le DNN a eu des résultats mitigés selon les classes, montrant des capacités limitées de généralisation.

Plusieurs facteurs peuvent expliquer ces résultats. Tout d'abord, l'architecture par couches de convolution du CNN est spécialement conçue pour extraire des caractéristiques visuelles, ce qui lui donne un avantage certain pour la classification d'images. Les DNN sont à l'inverse désavantagés par leur incapacité à exploiter la structure spatiale des images.

De plus, le RNN semble peiner à capturer toutes les dépendances spatiales dans des images complexes via ses connexions récurrentes. Sa mémoire interne apparaît insuffisante face à la haute dimensionalité visuelle de CIFAR-10.

En termes de vitesse d'entraînement, les DNN denses ont montré la plus grande efficacité une

fois déployée sur des plateformes favorisant la parallélisation de calcul. En effet, L'architecture des DNN avec la propagation Feedforward, donne la possibilité de calculer les sorties de chaque couche indépendamment des autres, et le calcul s'y limite souvent à des multiplications matricielles, ce qui se prête particulièrement bien au fonctionnement des GPU.

Cette étude comporte cependant certaines limites. Une implémentation type de chaque architecture a été testée, alors que de nombreuses variantes existent pour chaque architecture. De plus, l'analyse s'est limitée à deux ensembles de données qui restent de faible complexité par rapport aux données de la vie réelle, et dont le nombre d'observations peut brider les capacités d'un réseau de neurones qui est par nature gourmand en data. Une validation plus poussée avec des datasets de meilleure définition et ayant une plus grande quantité d'images, permettrait un affinage de nos conclusions.

CONCLUSION

En conclusion, cette étude comparative montre que les réseaux de neurones convolutifs (CNN) sont l'architecture la plus efficace pour la classification d'images, en particulier sur des ensembles de données complexes.

Les réseaux de neurones récurrents (RNN) peuvent également être performants mais semblent limités pour capturer toutes les dépendances spatiales dans des images riches.

Les réseaux denses (DNN) restent attractifs par leur rapidité d'entraînement sur GPU mais sont moins adaptés aux données visuelles complexes.

Ces résultats soulignent le potentiel prometteur des CNN pour la plupart des applications pratiques de classification d'images. Ils ouvrent également des pistes quant à l'optimisation des RNN et DNN pour améliorer leurs capacités sur ce type de données.

Des travaux futurs pourraient se pencher sur des architectures hybrides tirant parti des forces respectives de chaque type de réseau. L'augmentation des datasets d'apprentissage, ou l'apprentissage par transfert à partir de modèles pré-entraînés pourrait également permettre d'améliorer les performances, notamment des RNN et DNN.

En continuant à explorer les frontières de ces algorithmes d'apprentissage profond, de nouveaux progrès sont envisageables pour relever les défis posés par la classification d'images dans les applications du monde réel.

APPENDIX

1. Code du projet
2. Résultats des entrainements sous forme d'objets keras
3. Rapports de classification des différents modèles par classe de prédiction

ACKNOWLEDGMENT

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué au succès de l'écriture de cet article. Un remerciement spécial est adressé aux enseignants du département d'informatique de l'Université Lyon 2. En particulier à Madame Alawieh pour ses précieux conseils et son soutien tout au long de ce projet.

REFERENCES

(1) Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386–408.

(2) Rumelhart, D., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536.

(3) Szandała, T. (2021). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. In: Bhoi, A., Mallick, P., Liu, CM., Balas, V. (eds) Bio-inspired Neurocomputing. Studies in Computational Intelligence, vol 903. Springer, Singapore.

(4) Janocha, K., & Czarnecki, W. M. (2017). On loss functions for deep neural networks in classification. arXiv preprint arXiv:1702.05659.

(5) Johansson, E. M., Dowla, F. U., & Goodman, D. M. (1991). Backpropagation Learning for Multilayer Feed-forward Neural Networks Using the Conjugate Gradient Method. International Journal of Neural Systems, 2(4), 291-301.

(6) LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation, 1(4), 541–551.

(7) Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306. Elsevier.

(8) Deng, L. (2012). The MNIST Database of Handwritten Digit Images for Machine Learning Research. IEEE Signal Processing Magazine, 29(6). IEEE.

(9) MadhanMohan S., & Karthikeyan, E. (2022). Classification of Image using Deep Neural Networks and SoftMax Classifier with CIFAR datasets. In 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE.

(10) Seger, C. (2018). An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing. School of Electrical Engineering and Computer Science (EECS), KTH.

(11) Geiger C. & Wyart M. (2020). Scaling description of generalization with number of parameters in deep learning. Journal of Statistical Mechanics: Theory and Experiment,

2020(February), 023401. IOP Publishing Ltd and SISSA Medialab srl.

(12) Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer Normalization. arXiv preprint arXiv:1607.06450.

(13) Gordon-Rodriguez, E., Loaiza-Ganem, G., Pleiss, G., & Cunningham, J. P. (2020). Uses and Abuses of the Cross-Entropy Loss: Case Studies in Modern Deep Learning. In Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops (Vol. 137, pp. 1-10). Proceedings of Machine Learning Research. PMLR.