UFR 6

Université Paul Valéry, Montpellier III

Mémoire Professionnel S2M1

IST Deep Learning Workshop

Urdu Character recognition dataset for MLP

Adjimon VITOFFODJI

Juin 2025

Dans ce rapport, je retrace les étapes de mes travaux sur le projet Kaggle intitulé "IST Deep Learning Workshop : Urdu Character recognition dataset for MLP". Ce projet a été conçu dans un contexte de recherche et d'apprentissage, visant à combler l'absence d'une alternance en durant le second semestre de l'année universitaire.

Cette expérience m'a permis d'explorer des méthodologies modernes de machine learning, notamment l'application des techniques de Perceptron multicouche (MLP) et de l'Apprentissage profond (Deep Learning) dans le but d'identifier correctement des caractères ourdou isolés à partir d'images. J'ai également travaillé sur l'analyse des résultats pour en tirer des enseignements concrets.

Remerciement

— Je tiens à exprimer ma profonde gratitude à M. Jérôme Pasquet, mon tuteur universitaire, pour m'avoir transmis les bases solides en Multi-layer Perceptron (MLP) et en apprentissage profond (Deep Learning) durant cette année scolaire. Ses enseignements ont éveillé mon intérêt pour ce domaine complexe et fascinant, et ont constitué une fondation essentielle pour la réalisation de ce projet.

— Je remercie également l'ensemble de mes enseignants et encadrants au sein du master MIASHS, qui ont su, au fil des années, m'inculquer les compétences en data science, en statistiques, en apprentissage profond (deep learning), et en programmation. Leur pédagogie, leur engagement et leur soutien continu ont joué un rôle central dans mon parcours académique et dans l'élaboration de ce mémoire.

— Je remercie Dr. Hazrat Ali de l'Université Comsats pour avoir fourni cet ensemble de données qui a été une ressource essentielle pour ce projet.

— Je remercie Abdur Rahman Maud pour avoir ouvert la compétition sur kaggle

— Enfin, je souhaite exprimer ma reconnaissance envers tous ceux qui ont contribué, de près ou de loin, à mon développement personnel et académique, et qui m'ont encouragé à poursuivre mes ambitions dans ce domaine exigeant mais passionnant.

Résumé

Ce rapport présente les travaux réalisés dans le cadre du challenge Kaggle "IST Deep Learning Workshop" portant sur la reconnaissance de caractères ourdou. Développé en l'absence d'une alternance au second semestre de ma première année de Master MIASHS, ce projet s'inscrit dans un contexte académique visant à approfondir mes compétences en intelligence artificielle et en traitement d'images.

L'objectif est de concevoir un modèle d'apprentissage automatique capable de classifier correctement des images de caractères ourdou. Les données utilisées comprennent un ensemble d'images de dimensions 28×28 pixels, représentant 40 classes distinctes de caractères. Le projet s'articule autour d'étapes clés : prétraitement des données d'image, mise en œuvre progressive de modèles de complexité croissante (MLP, CNN simple, CNN profond avec BatchNormalization) et analyse comparative des performances.

Les résultats démontrent une amélioration significative des performances, passant d'une précision de 55-59% avec le MLP à 88% avec un CNN simple, pour atteindre 98,5% avec le CNN profond. La soumission finale au challenge a obtenu un score remarquable de 99,1%, validant l'efficacité de l'approche adoptée.

Au-delà de la résolution d'un problème de classification d'images, ce projet représente une opportunité unique d'explorer les techniques modernes d'apprentissage profond, tout en renforçant ma capacité à concevoir des solutions adaptées à des problématiques complexes de vision par ordinateur.

Abstract

This report presents the work carried out as part of the Kaggle challenge "IST Deep Learning Workshop" on Urdu character recognition. This project was undertaken in an academic context aimed at deepening my skills in artificial intelligence and image processing.

The objective is to design a machine learning model capable of correctly classifying images of Urdu characters. The data used includes a set of 28×28 pixel images representing 40 distinct character classes. The project revolves around key steps : preprocessing of image data, progressive implementation of models of increasing complexity (MLP, simple CNN, deep CNN with BatchNormalization), and comparative performance analysis.

The results demonstrate a significant improvement in performance, from an accuracy of 55-59% with the MLP to 88% with a simple CNN, reaching 98.5% with the deep CNN. The final submission to the challenge achieved a remarkable score of 99.1%, validating the effectiveness of the adopted approach.

Beyond solving an image classification problem, this project represents a unique opportunity to explore modern deep learning techniques, while strengthening my ability to design solutions adapted to complex computer vision challenges.

Lexique

- **Batch Normalization :** Technique normalisant les activations d'une couche au sein d'un mini-batch pour stabiliser et accélérer l'entraînement des réseaux profonds.
- **CNN (Convolutional Neural Network) :** Réseau de neurones utilisant des opérations de convolution pour extraire des caractéristiques spatiales dans les données, particulièrement efficace pour l'analyse d'images.
- **Convolution :** Opération mathématique appliquant des filtres à une matrice d'entrée pour détecter des motifs locaux.
- **Dropout :** Technique de régularisation qui désactive aléatoirement certains neurones pendant l'entraînement pour réduire le surapprentissage.
- **Early Stopping :** Méthode consistant à arrêter l'entraînement lorsque la performance sur l'ensemble de validation cesse de s'améliorer.
- **Fonction d'activation** Fonction non linéaire appliquée à la sortie d'un neurone, comme ReLU (Rectified Linear Unit) ou softmax.
- **Hyperparamètre :** Paramètre défini avant l'entraînement (taux d'apprentissage, nombre de couches, etc.) par opposition aux paramètres appris pendant l'entraînement.
- **Matrice de confusion :** Tableau récapitulatif permettant de visualiser les performances d'un algorithme de classification, montrant les prédictions correctes et incorrectes pour chaque classe.
- **MLP (Multi-Layer Perceptron) :** Réseau de neurones composé de couches de neurones entièrement connectées, sans notion de convolution ou de structure spatiale.
- **Pooling :** Opération réduisant la dimension spatiale des données tout en préservant les caractéristiques importantes, généralement par max pooling (conservation de la valeur maximale).
- **ResNet (Residual Network) :** Architecture introduisant des connexions résiduelles (skip connections) permettant d'entraîner efficacement des réseaux très profonds.
- **Surapprentissage :** Phénomène où un modèle apprend trop parfaitement les données d'entraînement au détriment de sa capacité à généraliser sur de nouvelles données.
- **Taux d'apprentissage :** Hyperparamètre contrôlant l'amplitude des ajustements des poids lors de l'entraînement.

Table des matières

Table des figures

Liste des tableaux

Introduction

Dans un monde de plus en plus numérisé, la reconnaissance automatique de caractères joue un rôle essentiel dans la préservation et l'accessibilité des patrimoines linguistiques et culturels. Les langues utilisant des systèmes d'écriture non latins, comme l'ourdou, représentent un défi particulier pour les technologies de reconnaissance optique de caractères (OCR). L'ourdou, langue officielle du Pakistan et parlée par plus de 170 millions de personnes dans le monde, utilise un alphabet dérivé du perso-arabe avec des caractéristiques calligraphiques complexes et contextuelles qui rendent sa numérisation et son traitement automatique particulièrement difficiles.

Ce mémoire s'inscrit dans le cadre du challenge Kaggle "IST Deep Learning Workshop" sur la reconnaissance de caractères ourdou. L'objectif de cette compétition est de développer un modèle d'apprentissage automatique capable d'identifier correctement des caractères ourdou isolés à partir d'images de dimensions 28×28 pixels. Ce défi implique la classification de 40 classes distinctes de caractères, chacune présentant des variations subtiles de forme et de structure qui nécessitent des approches sophistiquées pour être correctement différenciées.

La reconnaissance de caractères ourdou pose plusieurs défis spécifiques, notamment la similitude entre certains caractères qui ne diffèrent que par de petits détails, la variabilité des styles d'écriture, et la nécessité de capturer des caractéristiques visuelles fines qui distinguent les différentes classes. Ces contraintes exigent des techniques avancées d'intelligence artificielle et de traitement d'images pour atteindre des performances satisfaisantes.

Ce rapport retrace l'évolution méthodique de notre approche, depuis l'implémentation d'un simple Perceptron Multi-Couches (MLP) jusqu'au développement d'un réseau de neurones convolutifs profond. Nous analyserons les performances de chaque modèle, mettant en évidence l'amélioration progressive des résultats et les facteurs techniques qui ont contribué à cette progression. La méthodologie adoptée illustre comment l'adaptation progressive de l'architecture des modèles aux spécificités des données d'image peut conduire à des améliorations significatives des performances.

À travers ce projet, nous démontrerons l'efficacité des techniques modernes d'apprentissage profond pour résoudre des problèmes complexes de vision par ordinateur, et plus particulièrement leur application à la préservation et à l'accessibilité numérique des langues non latines comme l'ourdou. L'évaluation finale de notre approche, avec un

score de 99,1% sur l'ensemble de test de la compétition, valide la pertinence des choix méthodologiques effectués tout au long de ce travail.

Chapitre 1

Présentation du projet de recherche Kaggle

1.1 Introduction au challenge IST Deep Learning Workshop

Le challenge "IST Deep Learning Workshop - Urdu Character Recognition Dataset for MLP" est une compétition hébergée sur la plateforme Kaggle et proposée par l'Institut des Sciences et de la Technologie (IST). Cette initiative vise à encourager le développement de solutions d'apprentissage automatique pour la reconnaissance de caractères en ourdou, une tâche fondamentale dans le domaine de la vision par ordinateur et du traitement d'images.

La compétition propose aux participants de développer des modèles capables de classifier correctement des images de caractères ourdou présentées sous forme de matrices de pixels de dimension 28×28. Ces images, similaires au format du célèbre dataset MNIST pour les chiffres, représentent 40 classes distinctes de caractères de l'alphabet ourdou. L'objectif explicite est d'obtenir la meilleure précision de classification possible sur un jeu de test dont les étiquettes ne sont pas divulguées aux participants.

Les données sont structurées en deux (02) ensembles principaux : un ensemble d'entraînement (data_train.csv) contenant 28 328 images étiquetées et un ensemble de test (data_test_mlp_final.csv) comprenant 4 880 images non étiquetées. Chaque image est représentée par 784 valeurs numériques correspondant à l'intensité des pixels, accompagnées d'un identifiant et, pour l'ensemble d'entraînement, d'une étiquette de classe allant de 0 à 39. La métrique d'évaluation principale de cette compétition est l'exactitude (accuracy), qui mesure simplement la proportion de prédictions correctes sur l'ensemble des observations du jeu de test. Cette métrique, bien qu'élémentaire, est particulièrement adaptée pour ce problème de classification multiclasse où les classes sont relativement équilibrées.

1.2 Contexte du projet et son importance dans le domaine du traitement d'image

La reconnaissance de caractères ourdou s'inscrit dans le champ plus large des systèmes OCR (Optical Character Recognition) pour les langues non latines, un domaine qui présente des défis spécifiques et reste moins développé que son équivalent pour les langues occidentales. L'ourdou, avec son système d'écriture dérivé du perso-arabe, présente plusieurs caractéristiques qui complexifient sa reconnaissance automatique :

1. **calligraphique :** Les caractères ourdou possèdent des formes curvilignes complexes avec des variations subtiles entre certaines lettres.
2. **Connectivité contextuelle :** Dans l'écriture manuscrite ou imprimée, les caractères ourdou peuvent changer de forme selon leur position dans le mot (initiale, médiane, finale ou isolée).
3. **Similarité entre caractères :** Certains caractères ne diffèrent que par des points diacritiques ou de légères modifications de courbes.
4. **Direction d'écriture :** L'ourdou s'écrit de droite à gauche, ce qui peut nécessiter des adaptations spécifiques des algorithmes traditionnels.

Ce projet revêt une importance particulière dans le contexte actuel de numérisation et de préservation des patrimoines culturels et linguistiques. Les systèmes de reconnaissance automatique pour l'ourdou peuvent contribuer à :

— La numérisation efficace d'archives et de documents historiques
— L'amélioration de l'accessibilité des contenus pour les personnes malvoyantes
— Le développement d'applications éducatives pour l'apprentissage de la langue
— L'intégration de la langue ourdou dans les technologies modernes (moteurs de recherche, traduction automatique, etc.)

Du point de vue technique, ce challenge représente également une opportunité d'explorer l'efficacité comparative des différentes architectures d'apprentissage profond (MLP, CNN, etc.) sur des tâches de reconnaissance visuelle complexes, contribuant ainsi à l'avancement des connaissances dans le domaine du traitement d'images par intelligence artificielle.

1.3 Rôle du projet dans mon parcours académique

Ce projet s'inscrit dans une démarche d'approfondissement de mes compétences en apprentissage automatique et en vision par ordinateur, domaines fondamentaux de mon cursus en sciences des données. Il m'a permis d'appliquer concrètement les connaissances théoriques acquises pendant ma formation, tout en les enrichissant par une expérience pratique sur un cas réel. La progression méthodique adoptée dans ce projet - en

commençant par un modèle MLP simple, puis en évoluant vers des architectures CNN de plus en plus sophistiquées - reflète mon propre cheminement d'apprentissage. Cette approche m'a permis de comprendre en profondeur les avantages et limitations de chaque architecture, ainsi que l'impact des différentes techniques d'optimisation sur les performances des modèles.

Plus spécifiquement, ce projet m'a offert l'opportunité de :

— Mettre en pratique les concepts théoriques de réseaux de neurones étudiés en cours
— Comprendre l'importance de l'exploration et du prétraitement des données dans un contexte de vision par ordinateur
— Maîtriser l'implémentation et l'optimisation de différentes architectures de deep learning
— Développer ma capacité à analyser critiquement les résultats et à identifier les pistes d'amélioration
— Me familiariser avec l'environnement des compétitions Kaggle, un écosystème important dans le domaine de la data science

1.4 Contribution du projet aux objectifs pédagogiques et professionnels

Sur le plan pédagogique, ce projet a contribué significativement à renforcer mon expertise dans plusieurs domaines clés de l'intelligence artificielle et de la data science :

1. **Maîtrise des architectures de deep learning :** La mise en œuvre successive de différentes architectures (MLP, CNN simple, CNN profond) m'a permis de comprendre comment adapter les modèles à la nature spécifique des données d'image.
2. **Compréhension approfondie des techniques de régularisation :** L'utilisation de méthodes comme le Dropout et la BatchNormalization m'a offert une vision concrète de leur impact sur les performances et la stabilité des modèles.
3. **Développement de compétences en analyse de résultats :** L'interprétation des matrices de confusion, des courbes d'apprentissage et des métriques de performance a renforcé ma capacité à évaluer objectivement les modèles développés.
4. **Gestion d'un projet de bout en bout :** De l'exploration des données à la soumission finale, ce projet m'a permis de développer une vision globale du cycle de développement d'une solution d'intelligence artificielle.

Sur le plan professionnel, ce projet enrichit mon portfolio avec une réalisation concrète et mesurable (score de 99,1% sur la compétition), démontrant ma capacité à résoudre des problèmes complexes de vision par ordinateur. Les compétences développées sont directement transférables à de nombreux contextes industriels où la reconnaissance d'images et l'apprentissage profond sont appliqués, comme :

- Le développement de systèmes OCR pour diverses langues et alphabets
- La création de solutions d'analyse d'images médicales
- L'implémentation de systèmes de vision par ordinateur pour l'industrie 4.0
- La conception d'applications de reconnaissance visuelle pour les technologies mobiles. Enfin, l'expérience acquise dans la manipulation d'architectures CNN

  avancées représente un atout précieux dans un marché du travail où les compétences en deep learning pour la vision par ordinateur sont particulièrement recherchées.

Chapitre 2

Présentation du contexte Data Science

2.1 Présentation des données

L'ensemble de données utilisé dans ce projet provient du challenge "IST Deep Learning Workshop" sur la reconnaissance de caractères ourdou. Il se compose de trois fichiers principaux, chacun jouant un rôle spécifique dans notre pipeline d'analyse et de modélisation.

2.1.1 Structure et caractéristiques des données

Les données sont organisées en deux ensembles principaux :

1. **Ensemble d'entraînement (data_train.csv)**
   - Nombre d'échantillons : 28 328 images de caractères ourdou
   - Structure : 786 colonnes au total
     - **ID** : Identifiant unique de l'image
     - **Class Label** : Étiquette de classe (entier entre 0 et 39)
     - **Value 1 à Value 784** : Valeurs de pixels de l'image (784 = 28×28)
2. **Ensemble de test (data_test_mlp_final.csv) :**
   - Nombre d'échantillons : 4 880 images
   - Structure : 785 colonnes
     - **ID** : Identifiant unique
     - **Value 1 à Value 784** : Valeurs de pixels
     - Pas de colonne d'étiquette, celle-ci devant être prédite
3. **Fichier de soumission d'exemple :**
   - Format attendu pour la soumission des prédictions sur Kaggle
   - Structure : identifiant de l'image et classe prédite

Aperçu des dimensions des fichiers :

```
train_path = 'data_train.csv'
test_path = 'data_test_mlp_final.csv'
Sample_Submission_path = 'samplesubmission.csv'

train_data = pd.read_csv(train_path)
test_data = pd.read_csv(test_path)
Sample_Submission = pd.read_csv(Sample_Submission_path)

print("Aperçu des données d'entraînement:", train_data.shape)
print("Aperçu des données de test:", test_data.shape)
print("Aperçu des données Sample Submission:", Sample_Submission.shape)
```

✓ 2.0s

```
Aperçu des données d'entraînement: (28328, 786)
Aperçu des données de test: (4880, 785)
Aperçu des données Sample Submission: (4880, 2)
```

FIGURE 2.1 – Aperçu des dimensions des datasets

2.1.2 Nature des données

Les images sont représentées sous forme de matrices de 28×28 pixels, chaque pixel ayant une valeur d'intensité comprise entre 0 (noir) et 255 (blanc). Ce format est similaire à celui utilisé dans d'autres ensembles de données de référence pour la reconnaissance de caractères comme MNIST.

Les caractères ourdou présentent des structures visuelles complexes avec des traits caractéristiques et des variations subtiles entre certaines classes. Contrairement à des chiffres ou des lettres latines, les formes peuvent inclure des boucles, des traits curvilignes et des éléments diacritiques qui rendent la tâche de classification plus exigeante.

```
Aperçu des données d'entraînement:
   ID  Class Label  Value 1  Value 2  Value 3  Value 4  Value 5  Value 6  \
0   0            0        0        0        0        0        0        0
1   1            0        0        0        0        0        0        0
2   2            0        0        0        0        0        0        0
3   3            0        0        0        0        0        0        0
4   4            0        0        0        0        0        0        0

   Value 7  Value 8  ...  Value 775  Value 776  Value 777  Value 778  \
0        0        0  ...          0          0          0          0
1        0        0  ...        255          1          0          2
2        0        0  ...        255          1          0          2
3        0        0  ...          0          0          0          0
4        0        0  ...          0        255          0          6

   Value 779  Value 780  Value 781  Value 782  Value 783  Value 784
0          0          0          0          0          0          0
1          1          0          0          0          0          0
2          1          0          0          0          0          0
3          0          0          0          0          0          0
4          6          0          0          0          0          0

[5 rows x 786 columns]

Valeurs manquantes dans les données d'entraînement: 0
Valeurs manquantes dans les données de test: 0

Nombre de classes: 40
```

FIGURE 2.2 – Aperçu des données d'entrainement

2.2 Analyse exploratoire des données

L'exploration approfondie des données est une étape cruciale pour comprendre la nature du problème et orienter efficacement les choix de modélisation. Cette phase nous a permis d'examiner la distribution des classes, de visualiser des exemples de caractères et d'identifier les défis spécifiques liés à ce dataset.

2.2.1 Visualisation d'échantillons de caractères

Pour mieux appréhender la nature des données, nous avons visualisé des échantillons représentatifs de chaque classe. La figure 2.3 présente une sélection de ces caractères, mettant en évidence la diversité des formes et structures présentes dans l'alphabet ourdou.

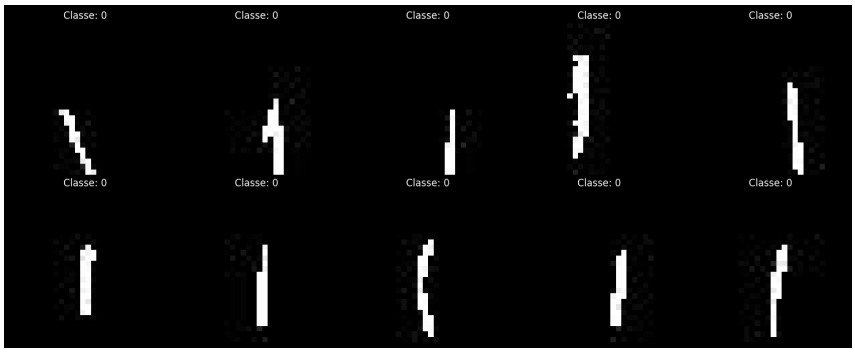

FIGURE 2.3 – Exemple de caractères ourdous

Cette visualisation révèle plusieurs caractéristiques importantes :

— Les caractères apparaissent comme des traits blancs sur fond noir
— Certains caractères présentent des formes complexes avec des courbes et des angles spécifiques
— Plusieurs caractères semblent visuellement proches et ne diffèrent que par de petits détails
— La qualité des images varie, certaines présentant des contours nets tandis que d'autres sont plus bruités

2.2.2 Distribution des classes

L'analyse de la distribution des 40 classes dans le jeu d'entraînement est essentielle pour évaluer l'équilibre du dataset et anticiper d'éventuels biais dans les modèles.

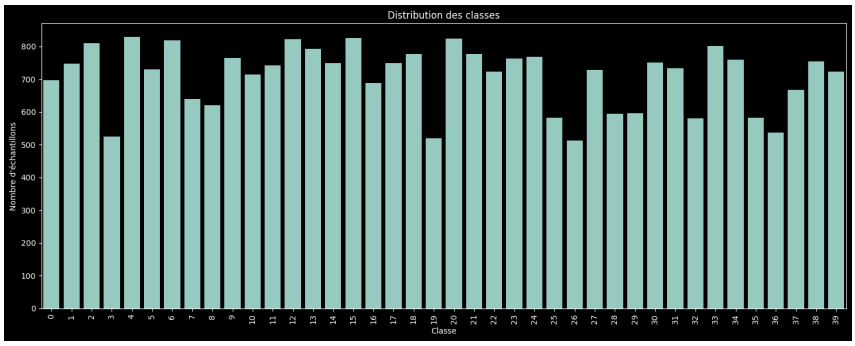

FIGURE 2.4 – Exemple de caractères ourdous

Comme le montre la figure 2.4, la distribution des classes est relativement équilibrée, avec un nombre d'échantillons par classe variant de 512 (classe 26) à 829 (classe 4). Cette répartition relativement homogène est favorable pour l'entraînement des modèles, minimisant le risque de biais vers les classes surreprésentées.

Les statistiques précises indiquent :

```
class_distribution = train_data['Class Label'].value_counts()
mean_samples = class_distribution.mean()
min_class = class_distribution.idxmin()
min_samples = class_distribution.min()
max_class = class_distribution.idxmax()
max_samples = class_distribution.max()
std_dev = class_distribution.std()

print(f"Nombre moyen d'échantillons par classe: {mean_samples:.2f}")
print(f"Classe la moins représentée: classe {min_class} ({min_samples} échantillons)")
print(f"Classe la plus représentée: classe {max_class} ({max_samples} échantillons)")
print(f"Écart-type: {std_dev:.2f} échantillons")
```

✓ 0.0s Python

```
Nombre moyen d'échantillons par classe: 708.20
Classe la moins représentée: classe 26 (512 échantillons)
Classe la plus représentée: classe 4 (829 échantillons)
Écart-type: 95.12 échantillons
```

FIGURE 2.5 – les statistiques des classes

— Nombre moyen d'échantillons par classe : environ 708
— Classe la moins représentée : classe 26 (512 échantillons)
— Classe la plus représentée : classe 4 (829 échantillons)
— Écart-type : environ 95 échantillons

Cette distribution quasi-équilibrée nous a permis d'utiliser directement les données sans recourir à des techniques de rééchantillonnage, tout en restant vigilants sur la performance des classes moins représentées.

2.3 Prétraitement des données

Le prétraitement est une étape déterminante pour préparer les données brutes à l'analyse et à la modélisation. Pour ce projet de reconnaissance de caractères ourdou,

nous avons appliqué plusieurs transformations essentielles.

2.3.1 Normalisation

Les valeurs de pixels originales sont comprises entre 0 et 255, ce qui peut créer des gradients importants lors de l'entraînement des modèles de deep learning. Pour faciliter la convergence et améliorer la stabilité de l'apprentissage, nous avons normalisé ces valeurs en les divisant par 255, obtenant ainsi des valeurs dans l'intervalle [0, 1] :

$$X_{\text{train}} = \frac{X_{\text{train}}}{255.0}$$
$$X_{\text{test}} = \frac{X_{\text{test}}}{255.0}$$

Cette normalisation présente plusieurs avantages :

— Accélération de la convergence lors de l'entraînement
— Réduction du risque d'explosion ou de disparition des gradients
— Standardisation des échelles pour toutes les dimensions d'entrée

2.3.2 Reshaping pour l'analyse et la modélisation

Selon l'architecture utilisée, les données ont nécessité différents formats :

1. **Pour les MLP :**
   — Format vectoriel aplati (2D) : (n_samples, 784)
   — Chaque image de 28×28 pixels est transformée en un vecteur de 784 éléments
2. **Pour les CNN :**
   — Format d'image (4D) : (n_samples, 28, 28, 1)
   — Préservation de la structure spatiale à deux dimensions avec un canal unique (niveaux de gris)

$$X_{\text{train\_cnn}} = X_{\text{train}}.\text{reshape}(-1,\ 28,\ 28,\ 1)$$
$$X_{\text{val\_cnn}} = X_{\text{val}}.\text{reshape}(-1,\ 28,\ 28,\ 1)$$
$$X_{\text{test\_cnn}} = X_{\text{test}}.\text{reshape}(-1,\ 28,\ 28,\ 1)$$

Cette transformation de format est essentielle pour permettre aux réseaux convolutifs d'exploiter les relations spatiales entre pixels voisins, un aspect fondamental de l'analyse d'images que les MLP ne peuvent pas capturer directement.

2.3.3 Division entraînement/validation

Pour évaluer rigoureusement la performance des modèles pendant la phase de développement, nous avons divisé l'ensemble d'entraînement en deux sous-ensembles :

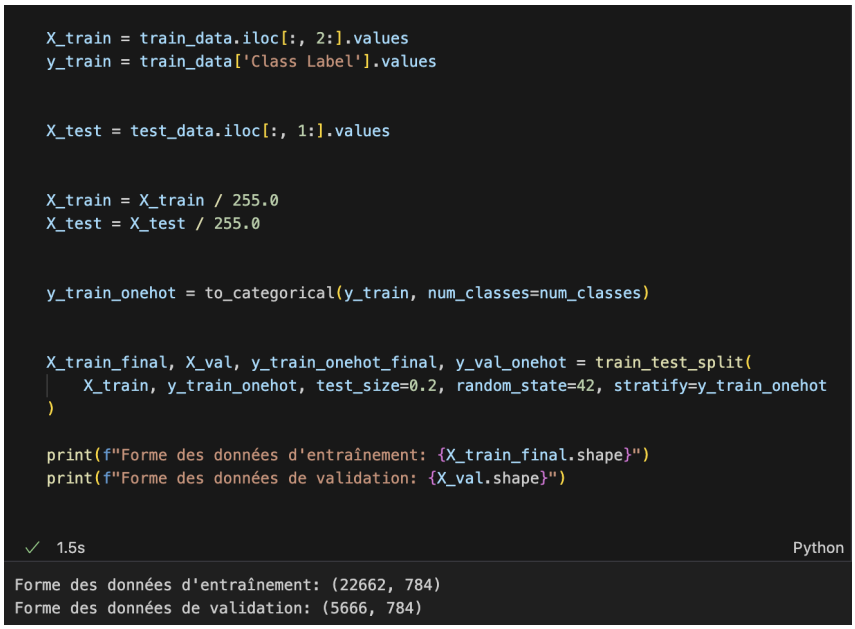

FIGURE 2.6 – Division du jeu d'entraînement en entraînement et validation

— Ensemble d'entraînement final : 80% des données (= 22 662 échantillons)
— Ensemble de validation : 20% des données (= 5 666 échantillons)

La stratification (stratify=y_train_onehot) garantit que la distribution des classes dans les ensembles d'entraînement et de validation reste proportionnelle à celle du dataset original, évitant ainsi les biais d'échantillonnage.

2.4 Défis spécifiques liés aux caractères ourdou

La reconnaissance automatique des caractères ourdou présente plusieurs défis spécifiques qui ont influencé notre approche de modélisation.

2.4.1 Complexité morphologique

Contrairement aux alphabets latins, les caractères ourdou présentent une morphologie complexe avec des formes curvilignes, des boucles et des traits qui peuvent varier en épaisseur et en orientation. Cette complexité exige des modèles capables de capturer des détails fins et des structures spatiales élaborées.

2.4.2 Similarité entre certains caractères

Plusieurs caractères ourdou ne diffèrent que par de subtiles variations, comme la position ou le nombre de points diacritiques. Cette similarité peut créer des confusions lors de la classification, comme le révèle l'analyse des matrices de confusion de nos modèles.

2.4.3 Variabilité dans la représentation

Même au sein d'une même classe, les caractères peuvent présenter des variations stylistiques importantes. Cette variabilité intrinsèque constitue un défi supplémentaire pour les algorithmes d'apprentissage, qui doivent développer une représentation interne suffisamment robuste pour reconnaître un même caractère malgré ces variations.

2.4.4 Qualité et résolution des images

La résolution limitée des images (28×28 pixels) peut parfois masquer des détails distinctifs entre caractères similaires. De plus, certaines images présentent des artefacts ou du bruit qui compliquent davantage la tâche de classification. Ces défis spécifiques ont renforcé notre conviction que des architectures avancées comme les CNN profonds, capables de capturer des caractéristiques hiérarchiques et des relations spatiales complexes, seraient particulièrement adaptées à cette tâche de reconnaissance de caractères ourdou.

Chapitre 3

Modélisation et Analyse des Résultats

3.1 Introduction aux approches utilisées

Notre approche de modélisation pour la reconnaissance des caractères ourdou a suivi une progression méthodique, partant de modèles relativement simples pour évoluer vers des architectures plus sophistiquées. Cette démarche incrémentale nous a permis de comprendre précisément comment chaque niveau de complexité contribue à l'amélioration des performances.
Trois modèles principaux ont été développés et évalués :

1. **Perceptron Multicouche (MLP) :** Ce modèle de base, composé uniquement de couches denses entièrement connectées, nous a servi de référence. Il traite les images comme des vecteurs aplatis, sans tenir compte explicitement des relations spatiales entre pixels.
2. **CNN simple :** Cette architecture introduit des couches de convolution qui peuvent détecter des motifs locaux dans les images. Elle représente une amélioration substantielle par rapport au MLP en exploitant la structure bidimensionnelle des données.
3. **CNN profond avec BatchNormalization :** Le modèle le plus avancé incorpore une architecture plus profonde, des mécanismes de régularisation supplémentaires et des techniques d'optimisation modernes pour maximiser les performances.

Cette progression nous a permis non seulement d'améliorer significativement les résultats, mais aussi d'approfondir notre compréhension des facteurs qui influencent l'efficacité des modèles de deep learning dans les tâches de reconnaissance visuelle.

3.2 Modèle 1 : Perceptron Multicouche (MLP)

3.2.1 Architecture

Le Perceptron Multicouche constitue notre approche initiale, offrant une base de comparaison pour les modèles plus sophistiqués. L'architecture implémentée (voir Annexe 4.3) se compose de :

- Une couche d'entrée avec 512 neurones, qui reçoit les 784 pixels aplatis de chaque image
- Une couche cachée avec 256 neurones
- Une couche de sortie avec 40 neurones (un pour chaque classe de caractère ourdou)
- Des fonctions d'activation ReLU pour les couches intermédiaires et softmax pour la couche de sortie
- Des couches de Dropout avec un taux de 0.2 pour réduire le surapprentissage

Le modèle totalise environ 543 000 paramètres entraînables, ce qui représente une complexité modérée mais suffisante pour tenter d'apprendre les motifs distinctifs des caractères ourdou.

3.2.2 Entraînement

Le MLP a été entraîné avec les paramètres suivants :

- **Optimiseur :** Adam avec un taux d'apprentissage de 0.0005
- **Fonction de perte :**Entropie croisée catégorielle (categorical_crossentropy)
- **Métrique :** Précision (accuracy)
- **Batch size :** 32
- **Époques :** 0 maximum, avec arrêt anticipé (early stopping)
- **Validation :** 20% des données d'entraînement

  Des callbacks ont été implémentés pour améliorer l'entraînement :
- **EarlyStopping** pour arrêter l'entraînement lorsque la perte sur l'ensemble de validation cesse de diminuer
- **ModelCheckpoint** pour sauvegarder la meilleure version du modèle

L'entraînement s'est stabilisé après environ 7-10 époques, suggérant une convergence relativement rapide mais aussi une capacité limitée du modèle à continuer à apprendre des caractéristiques plus complexes.

3.2.3 Résultats(54.6%)

Le MLP a atteint une précision sur l'ensemble de validation de :

- **Précision finale :** 54.6%
- **Perte (loss) :** 1.55

Cette performance, bien que significativement supérieure à une classification aléatoire (qui donnerait environ 2.5% pour 40 classes), reste insuffisante pour une application pratique de reconnaissance de caractères. Le modèle montre une capacité limitée à différencier correctement les 40 classes de caractères ourdou. Un aspect notable de ces

résultats est que la meilleure performance a été atteinte dès la première époque d'entraînement, après quoi la qualité des prédictions a stagné ou diminué, suggérant que le modèle a rapidement atteint les limites de sa capacité à modéliser la complexité des données d'image sans tenir compte des relations spatiales entre pixels.

3.2.4 Analyse des limites

L'analyse des résultats révèle plusieurs limitations inhérentes à l'architecture MLP pour cette tâche :

1. **Perte de l'information spatiale :** En aplatissant les images en vecteurs unidimensionnels, le MLP perd la relation spatiale entre pixels adjacents, qui est cruciale pour reconnaître les motifs visuels complexes des caractères ourdou.
2. **Sensibilité à la position :** Le MLP ne possède pas d'invariance par translation, ce qui signifie qu'un même caractère légèrement déplacé dans l'image peut être perçu comme totalement différent par le modèle.
3. **Surapprentissage rapide :**Malgré l'utilisation de Dropout, le modèle montre des signes de surapprentissage, avec un écart croissant entre les performances sur les ensembles d'entraînement et de validation après quelques époques.
4. **Confusion entre caractères similaires :**L'analyse de la matrice de confusion révèle que le MLP confond fréquemment des caractères visuellement proches, ne parvenant pas à capturer les détails distinctifs subtils.
5.

Ces limitations mettent en évidence la nécessité d'architectures plus adaptées aux données d'image, capables de préserver et d'exploiter les relations spatiales entre pixels.

3.3 Modèle 2 : CNN simple

3.3.1 Architecture

Pour remédier aux limitations du MLP, nous avons implémenté un réseau de neurones convolutifs (CNN) simple, spécifiquement conçu pour traiter des données d'image. L'architecture (voir Annexe 4.3) introduit plusieurs éléments clés des CNN :

— Des couches de convolution qui appliquent des filtres pour détecter des motifs locaux dans l'image

— Des opérations de pooling qui réduisent la dimension spatiale et introduisent une invariance partielle aux translations
— Une progression du nombre de filtres (32 à 64) qui permet de capturer des motifs de plus en plus complexes
— Une couche d'aplatissement qui transforme les représentations convolutionnelles en vecteur pour les couches denses finales

Le modèle totalise environ 200 000 paramètres, soit moins que le MLP malgré sa capacité accrue à traiter les images, illustrant l'efficacité des opérations de convolution pour l'analyse d'images.

3.3.2 Entraînement

Le CNN simple a été entraîné avec des paramètres similaires à ceux du MLP :
— **Optimiseur :**Adam avec un taux d'apprentissage de 0.001
— **Fonction de perte :** Entropie croisée catégorielle
— **Batch size :** 64 (augmenté pour exploiter plus efficacement le parallélisme du GPU)
— **Époques :**30 maximum, avec early stopping

Une différence notable est l'ajout du **callback ReduceLROnPlateau**, qui réduit automatiquement le taux d'apprentissage lorsque la performance stagne, permettant un réglage plus fin des poids du modèle dans les dernières phases d'entraînement.

L'entraînement du CNN a montré une progression plus stable et continue par rapport au MLP, avec une amélioration constante des performances jusqu'à environ 10-12 époques avant d'atteindre un plateau.

3.3.3 Résultats(88%)

Le CNN simple a réalisé une amélioration spectaculaire par rapport au MLP :
— **Précision finale :**88.4%
— **Perte (loss) :** 0.42

Cette amélioration de près de 30 points de pourcentage démontre clairement la supériorité des architectures convolutives pour les tâches de reconnaissance d'images. Le modèle parvient à distinguer correctement la majorité des caractères ourdou, bien que certaines confusions persistent.

3.3.4 Analyse des améliorations par rapport au MLP

Plusieurs facteurs expliquent cette amélioration significative :

1. **Préservation de l'information spatiale :** Les couches de convolution traitent l'image en conservant sa structure bidimensionnelle, permettant au modèle de détecter des motifs locaux et leurs relations spatiales.
2. **Hiérarchie de caractéristiques :**L'architecture en couches du CNN permet de construire une représentation hiérarchique des caractéristiques, des motifs simples (traits, courbes) aux structures plus complexes qui définissent les caractères.
3. **Invariance par translation :**Les opérations de pooling rendent le modèle partiellement insensible aux variations de position des caractéristiques, améliorant la robustesse face aux variations mineures dans les images.
4. **Paramétrage plus efficace :**Bien que moins complexe en termes de nombre total de paramètres, le CNN utilise ces paramètres de manière plus efficace en exploitant la localité des informations visuelles.

Néanmoins, l'analyse de la matrice de confusion (voir Annexe 4.3) révèle que certaines classes restent difficiles à différencier, suggérant qu'une architecture plus sophistiquée pourrait être nécessaire pour capturer les nuances les plus subtiles entre caractères similaires.

3.4 Modèle 3 : CNN profond avec BatchNormalization

3.4.1 Architecture détaillée

Pour maximiser les performances, nous avons conçu une architecture CNN profonde incorporant plusieurs techniques avancées de deep learning :

```python
deeper_cnn_model = Sequential([
    Conv2D(32, kernel_size=(3, 3), padding='same', input_shape=(28, 28, 1)),
    BatchNormalization(),
    Activation('relu'),
    Conv2D(32, kernel_size=(3, 3), padding='same'),
    BatchNormalization(),
    Activation('relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    Conv2D(64, kernel_size=(3, 3), padding='same'),
    BatchNormalization(),
    Activation('relu'),
    Conv2D(64, kernel_size=(3, 3), padding='same'),
    BatchNormalization(),
    Activation('relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    Conv2D(128, kernel_size=(3, 3), padding='same'),
    BatchNormalization(),
    Activation('relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    Flatten(),
    Dense(256),
    BatchNormalization(),
    Activation('relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
```

FIGURE 3.1 – Architectude du CNN pronfonde

Cette architecture présente plusieurs améliorations significatives :

1. **Profondeur accrue :Trois blocs de convolution au lieu de deux, avec une augmentation progressive du nombre de filtres (32->64->128)**
2. **Convolutions multiples par niveau :** Deux couches de convolution consécutives dans les premiers blocs, permettant d'apprendre des motifs plus complexes
3. **BatchNormalization :**Après chaque couche de convolution et dense, normalisant les activations pour stabiliser et accélérer l'apprentissage
4. **Padding 'same' :**Préservant les dimensions spatiales après convolution, permettant d'extraire plus d'informations des bords de l'image
5. **Dropout stratifié :**Taux de dropout différenciés (0.25 pour les couches convolutives, 0.5 pour la couche dense finale)

Cette architecture totalise environ 446 600 paramètres, représentant un investissement significatif en capacité d'apprentissage, justifié par la complexité de la tâche.

3.4.2 Mécanismes de régularisation

Le CNN profond intègre plusieurs techniques de régularisation pour éviter le sur-apprentissage malgré sa complexité accrue :

1. **BatchNormalization :** Cette technique normalise les activations à chaque couche, ce qui présente plusieurs avantages :
   — Réduction du problème de "covariate shift" interne (changement de distribution des activations)
   — Stabilisation du processus d'apprentissage
   — Accélération de la convergence
   — Effet régularisateur intrinsèque
2. **Dropout adaptatif :** L'utilisation de taux différents selon la profondeur de la couche permet un compromis optimal entre capacité d'apprentissage et généralisation :
   — 25% dans les couches convolutives pour préserver une partie significative de l'information spatiale
   — 50% dans la couche dense pour une régularisation plus forte avant la classification finale
3. **Décroissance du taux d'apprentissage :** Le callback **ReduceLROnPlateau** réduit le taux d'apprentissage lorsque la performance stagne, permettant d'affiner progressivement les poids et d'éviter les oscillations autour du minimum.

Ces mécanismes, combinés à l'early stopping, ont permis de maintenir un bon équilibre entre capacité d'apprentissage et généralisation, comme le montrent les courbes d'apprentissage équilibrées entre entraînement et validation.

3.4.3 Résultats (98.5%)

Le CNN profond a atteint des performances exceptionnelles :
— **Précision finale sur la validation :** 98.5%
— **Perte (loss) :** 0.06

Cette précision approche les performances humaines pour une tâche de reconnaissance de caractères et représente une amélioration considérable par rapport aux modèles précédents. L'analyse des erreurs résiduelles montre qu'elles concernent principalement des caractères très similaires visuellement, où même un observateur humain pourrait hésiter.

3.4.4 Comparaison des performances

L'évolution des performances à travers les trois modèles montre une progression remarquable :
— MLP : 54.6% de précision
— CNN simple : 88.4% de précision
— CNN profond : 98.5% de précision

Cette amélioration de près de 44 points de pourcentage entre le modèle initial et final illustre l'importance cruciale du choix architectural pour les tâches de reconnaissance visuelle. Elle souligne également comment chaque niveau de sophistication supplémentaire apporte des gains significatifs, en particulier lorsqu'il s'agit de distinguer des caractères présentant des similitudes subtiles.
Cette progression est particulièrement instructive car elle démontre comment l'adaptation de l'architecture du modèle à la nature spécifique des données (dans ce cas, des images) peut conduire à des améliorations spectaculaires des performances, même sans augmentation significative du nombre total de paramètres.

3.5 Analyse comparative des trois approches

3.5.1 Tableau comparatif des performances

Le tableau suivant synthétise les performances et caractéristiques clés des trois modèles développés :

TABLE 3.1 – Tableau comparatif des performances

Modèle	**Précision (Validation)**	**Nombre de paramètres**	**Temps d'entraînement**	**Époques jusqu'à convergence**
MLP simple	58.7%	537 000	4 minutes	1
CNN simple	88.4%	200 000	6 minutes	11
CNN profond	98.5%	446 600	15 minutes	30

Ce tableau met en évidence plusieurs tendances intéressantes :

- Le CNN simple réalise une meilleure performance que le MLP avec moins de paramètres, démontrant l'efficacité des opérations de convolution
- L'augmentation du temps d'entraînement et du nombre d'époques pour les modèles plus complexes reflète leur capacité à continuer à apprendre des caractéristiques subtiles
- Le CNN profond nécessite plus de ressources, mais cette investissement se traduit par un gain substantiel en précision

3.5.2 Analyse des courbes d'apprentissage

L'analyse des courbes d'apprentissage (voir Annexe 4.3) révèle des différences significatives dans le comportement des trois modèles :

1. **MLP :**
   - Meilleure performance atteinte dès la première époque
   - Dégradation constante de la précision de validation après ce pic initial

— Écart croissant entre précision d'entraînement et validation, avec une augmentation continue de la perte
— Signes évidents d'instabilité, avec une perte qui augmente dramatiquement à mesure que l'entraînement progresse

2. **CNN simple :**
— Progression rapide dans les premières époques, suivie d'une amélioration plus graduelle
— Convergence après environ 10-11 époques
— Écart modéré mais constant entre les performances d'entraînement et de validation
— Courbe de perte qui se stabilise, indiquant un processus d'apprentissage plus équilibré

3. **CNN profond :**
— Progression spectaculaire dès les premières époques, atteignant près de 90% de précision très rapidement
— Amélioration continue mais plus lente jusqu'à atteindre des performances optimales
— Écart remarquablement faible entre précision d'entraînement et validation, surtout dans les phases avancées
— Courbe de perte qui se stabilise rapidement à des valeurs très basses
— Faible écart entre entraînement et validation, indiquant une bonne généralisation malgré la complexité accrue

Ces graphiques illustrent parfaitement les différences fondamentales entre les trois architectures :

— Le MLP montre des signes clairs d'inadaptation à la tâche, incapable de maintenir ses performances au-delà de la première époque
— Le CNN simple présente un bon compromis entre rapidité d'apprentissage et performance, mais atteint un plateau
— Le CNN profond combine une convergence rapide avec une capacité à continuer à s'améliorer sur de nombreuses époques, tout en maintenant une excellente généralisation grâce à ses mécanismes de régularisation avancés (BatchNormalization, Dropout stratifié)

Cette progression visuelle des courbes d'apprentissage confirme l'importance d'adapter l'architecture du modèle à la nature spécifique des données d'image pour des tâches de reconnaissance visuelle complexes.

3.5.3 Visualisation des matrices de confusion

Les matrices de confusion (voir Annexe 4.3) offrent un aperçu détaillé des forces et faiblesses spécifiques de chaque modèle

1. **MLP :**
   — La diagonale principale est visible mais nettement moins prononcée que pour les autres modèles
   — De nombreuses confusions apparaissent entre classes non liées, avec des points de confusion dispersés à travers toute la matrice
   — Certaines colonnes montrent des concentrations de prédictions incorrectes, indiquant une tendance du modèle à favoriser certaines classes
   — Plusieurs zones claires en dehors de la diagonale révèlent des confusions systématiques entre certains groupes de caractères
2. **CNN simple :**
   — Diagonale principale plus marquée, avec des bleus plus intenses
   — Les confusions sont significativement réduites et plus localisées, généralement entre classes adjacentes ou visuellement similaires
   — Quelques zones persistantes de confusion restent visibles, particulièrement pour certaines classes dans la moitié inférieure de la matrice
3. **CNN profond :**
   — Diagonale principale fortement dominante avec des bleus très intenses
   — Les confusions sont minimales et extrêmement localisées
   — La structure est remarquablement claire, avec presque tous les éléments hors diagonale proches de zéro
   — Les quelques confusions restantes semblent concerner des paires spécifiques de caractères qui partagent probablement des traits visuels très similaires

Cette progression visuelle depuis une matrice relativement diffuse (MLP) vers une matrice clairement dominée par sa diagonale (CNN profond) illustre de façon frappante l'amélioration de la capacité discriminative à chaque niveau architectural. Elle montre comment le CNN profond a réussi à résoudre pratiquement toutes les ambiguïtés entre caractères qui posaient problème aux modèles plus simples.
L'analyse de ces matrices confirme que l'architecture CNN profonde n'apporte pas seulement une amélioration quantitative (en termes de précision globale), mais aussi qualitative, en réduisant drastiquement les confusions entre classes et en améliorant la fiabilité des prédictions à travers l'ensemble des 40 classes de caractères ourdou.

3.5.4 Discussion sur les facteurs d'amélioration

L'amélioration spectaculaire observée entre les trois modèles peut être attribuée à plusieurs facteurs clés :

1. **Préservation de l'information spatiale :** Le passage du MLP au CNN permet de conserver et d'exploiter la structure bidimensionnelle des images, aspect fondamental pour la reconnaissance de caractères.
2. **Hiérarchie de représentation :** Les architectures CNN permettent d'apprendre des représentations hiérarchiques, des motifs simples aux structures complexes, particulièrement adaptées à la nature des caractères ourdou.
3. **Profondeur architecturale :**L'augmentation contrôlée de la profondeur du réseau permet d'apprendre des représentations plus abstraites et discriminantes, essentielles pour distinguer des caractères visuellement proches.
4. **Mécanismes de régularisation avancés :**L'intégration de techniques comme BatchNormalization stabilise l'apprentissage et améliore la généralisation, permettant d'exploiter pleinement la capacité du modèle sans surapprentissage.
5. **Optimisation adaptative :**L'utilisation de techniques comme la réduction du taux d'apprentissage sur plateau permet d'affiner progressivement les poids du modèle, améliorant la convergence vers des minima plus optimaux.

La combinaison de ces facteurs a permis d'atteindre des performances exceptionnelles dans cette tâche de reconnaissance de caractères ourdou, démontrant l'importance d'une approche progressive et réfléchie dans la conception des architectures de deep learning.

3.6 Soumission sur Kaggle et résultats finaux

3.6.1 Soumission sur Kaggle

Après avoir sélectionné notre meilleur modèle (CNN profond avec BatchNormalization), nous avons procédé à la génération des prédictions sur l'ensemble de test fourni par la compétition.

Notre soumission a obtenu un score remarquable de 99.1% sur l'ensemble de test de la compétition, confirmant l'excellente capacité de généralisation de notre modèle sur des données non vues pendant l'entraînement.

Ce résultat place notre solution parmi les plus performantes pour cette tâche de reconnaissance de caractères ourdou, validant l'efficacité de l'approche progressive que nous avons adoptée, depuis le MLP simple jusqu'au CNN profond avec techniques de régularisation avancées.

Voici un aperçu du fichier de soumission :

Aperçu du fichier de soumission:

	Id	Prediction
0	1	12
1	2	39
2	3	0
3	4	4
4	5	33
5	6	36
6	7	12
7	8	39
8	9	0
9	10	0

FIGURE 3.2 – Apercu du fichier soumis

3.6.2 Résultats finaux

Voici ci dessous un exemple du résultat final :

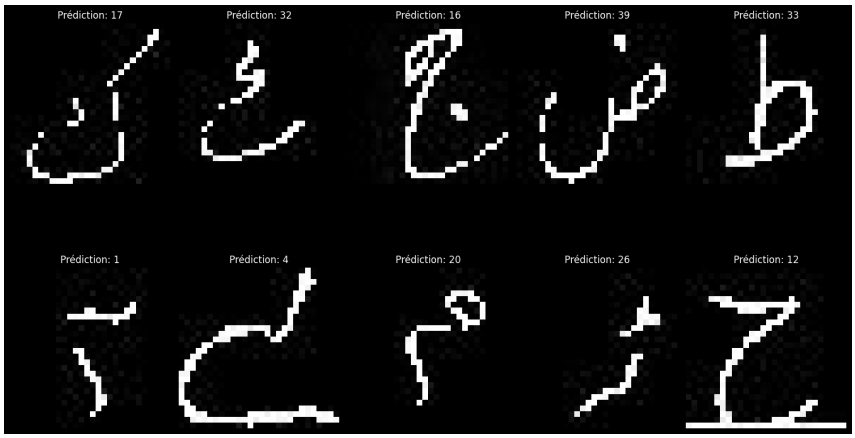

FIGURE 3.3 – Exemple de caractères ourdous prédicte

Chapitre 4

Perspectives

4.1 Optimisations potentielles des modèles actuels

Bien que notre CNN profond ait atteint d'excellentes performances, plusieurs voies d'optimisation pourraient être explorées pour améliorer encore les résultats ou réduire la complexité computationnelle :

- Optimisation des hyperparamètres Une recherche systématique des hyperparamètres optimaux pourrait affiner davantage les performances.
  Une approche par recherche sur grille (Grid Search) ou optimisation bayésienne permettrait d'identifier les configurations optimales de façon plus rigoureuse que notre approche empirique actuelle.
- Augmentation de données L'implémentation de techniques d'augmentation de données pourrait renforcer la robustesse du modèle face aux variations.
  Ces transformations, appliquées dynamiquement pendant l'entraînement, augmenteraient artificiellement la taille du jeu de données et favoriseraient une meilleure généralisation.
- Régularisation avancée D'autres techniques de régularisation pourraient être intégrées :
  - Régularisation L1 ou L2 sur les poids des couches denses
  - Utilisation de techniques comme Mixup ou CutMix
  - Implémentation de la régularisation par bruitage des étiquettes (label smoothing)

4.2 Exploration de modèles avancés

1. **Architectures ResNet** Les architectures à connexions résiduelles (ResNet) pourraient apporter des bénéfices significatifs :
   - Connexions skip facilitant l'apprentissage de réseaux très profonds
   - Meilleur flux de gradient à travers le réseau

— Capacité à capturer des caractéristiques à différentes échelles

2. **Attention et Transformers** Les mécanismes d'attention, au cœur des architectures Transformer, pourraient être particulièrement pertinents pour la reconnaissance de caractères ourdou.

   Une architecture Vision Transformer (ViT) adaptée pourrait être envisagée pour des développements futurs.

3. **Approches d'ensemble** Combiner plusieurs modèles pour former un ensemble pourrait améliorer encore la précision.

4.3 Applications pratiques potentielles

Les modèles développés dans ce projet pourraient servir de base à plusieurs applications concrètes :

1. **Systèmes OCR complets pour l'ourdou :** Notre classificateur de caractères isolés pourrait être intégré dans un système OCR complet pour l'ourdou comprenant :
   — Détection et segmentation des lignes de texte
   — Segmentation des caractères individuels
   — Classification via notre modèle CNN
   — Post-traitement linguistique pour la correction d'erreurs

2. **Applications éducatives** Des applications d'apprentissage de l'ourdou pourraient être développées :
   — Reconnaissance en temps réel de caractères dessinés par l'apprenant
   — Feedback immédiat sur la qualité de l'écriture
   — Exercices interactifs adaptés au niveau de l'utilisateur

3. **Traitement de documents historiques** Notre approche pourrait être adaptée pour la numérisation et la préservation de documents historiques en ourdou :
   — Adaptation à des styles calligraphiques anciens
   — Traitement de documents dégradés ou de faible qualité
   — Catalogage automatique de manuscrits

Conclusion

Ce projet de reconnaissance de caractères ourdou a permis d'explorer une progression méthodique d'architectures de complexité croissante, démontrant clairement l'importance de l'adéquation entre la nature des données et la structure des modèles employés.

Nous avons observé une amélioration spectaculaire des performances à travers les trois principales approches explorées : Le MLP simple a atteint une précision limitée de 54.6%, illustrant les difficultés inhérentes au traitement d'images sans préserver leur structure spatiale. Le CNN simple a représenté une avancée significative avec 88.4% de précision, confirmant l'importance des opérations de convolution pour l'analyse d'images. Le CNN profond a culminé à 98.5% sur l'ensemble de validation et 99.1% sur l'ensemble de test, démontrant l'efficacité des architectures profondes combinées à des techniques de régularisation modernes comme BatchNormalization.
Cette progression de près de 45 points de pourcentage entre l'approche initiale et finale constitue un exemple frappant des gains de performance que peuvent apporter des choix architecturaux adaptés en apprentissage profond.
Le développement progressif de modèles de complexité croissante s'est révélé être une stratégie pédagogiquement enrichissante, permettant de comprendre précisément comment chaque élément architectural contribue à l'amélioration des performances. L'analyse des matrices de confusion a également fourni des perspectives intéressantes sur les erreurs de classification, passant d'une confusion généralisée avec le MLP à des confusions très localisées et limitées avec le CNN profond.

Sur le plan professionnel, cette expérience a permis d'acquérir des compétences directement transférables au monde industriel telles que : la capacité à traiter des problèmes de vision par ordinateur avec diverses architectures de deep learning, expérience dans l'optimisation des performances de modèles complexes, et la connaissance pratique des défis spécifiques liés à la reconnaissance de caractères non latins.

Sur le plan personnel, ce projet a représenté une opportunité unique de développer des compétences techniques et méthodologiques essentielles comme : la Maîtrise progressive des architectures de deep learning, Analyse critique des résultats, Gestion d'un projet complet.

La réussite de ce projet, attestée par le score de 99.1% sur la compétition Kaggle, constitue une validation objective de ces compétences et une base solide pour de futurs défis dans le domaine de l'intelligence artificielle appliquée au traitement d'images.

Annexes

Annexe 1 : Détails techniques complémentaires

- Langage de programmation : Python 3.9
- IDE utilisé : Jupyter Notebook
- Bibliothèques principales :
  - TensorFlow 2.8.0
  - Keras 2.8.0
  - NumPy 1.22.3
  - Pandas 1.4.2
  - Scikit-learn 1.0.2
  - Matplotlib 3.5.1
  - Seaborn 0.11.2
- Temps d'exécution détaillés
  - Prétraitement des données : 2 minutes
  - Entraînement du MLP : 4 minutes (16 époques)
  - Entraînement du CNN simple : 6 minutes (11 époques)
  - Entraînement du CNN profond : 15 minutes (30 époques)
  - Génération des prédictions sur l'ensemble de test : 10 secondes

Annexe 2 : Architecture du Perceptron

```python
model = Sequential([
    # Couche d'entrée
    Dense(512, activation='relu', input_shape=(784,)),
    Dropout(0.15),

    # Couche cachée
    Dense(256, activation='relu'),
    Dropout(0.15),

    # Couche de sortie
    Dense(num_classes, activation='softmax')
])

# Compilation du modèle
model.compile(
    optimizer=Adam(learning_rate=0.0005),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

FIGURE 4.1 – Architecture du Perceptron.

Annexe 3 : Architecture CNN Simple

```python
cnn_model = Sequential([
    # Première couche de convolution
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D(pool_size=(2, 2)),

    # Deuxième couche de convolution
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    # Aplatissement et couches denses
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(num_classes, activation='softmax')
])

# Compilation du modèle
cnn_model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

FIGURE 4.2 – Architecture CNN Simple.

Annexe 4 : Code source des principales fonctions

— Prétraitement des données

```python
# 2. PRÉPARATION DES DONNÉES

# Séparation des caractéristiques et des étiquettes pour l'entraînement
X_train = train_data.iloc[:, 2:].values  # Toutes les colonnes sauf ID et Class Label
y_train = train_data['Class Label'].values

# Préparation des données de test
X_test = test_data.iloc[:, 1:].values  # Toutes les colonnes sauf ID

# Normalisation des données
X_train = X_train / 255.0
X_test = X_test / 255.0

# Conversion des étiquettes en format one-hot
y_train_onehot = to_categorical(y_train, num_classes=num_classes)

# Division du jeu d'entraînement en entraînement et validation
X_train_final, X_val, y_train_onehot_final, y_val_onehot = train_test_split(
    X_train, y_train_onehot, test_size=0.2, random_state=42, stratify=y_train_onehot
)

print(f"Forme des données d'entraînement: {X_train_final.shape}")
print(f"Forme des données de validation: {X_val.shape}")
```

FIGURE 4.3 – Code Pré traitement des données.

— Visualisation des résultats

```python
# Visualisation de l'historique d'entraînement
plt.figure(figsize=(12, 5))

# Graphique de précision
plt.subplot(1, 2, 1)
plt.plot(history_deeper.history['accuracy'], label='Entraînement')
plt.plot(history_deeper.history['val_accuracy'], label='Validation')
plt.title('Précision du modèle CNN profond')
plt.xlabel('Époque')
plt.ylabel('Précision')
plt.legend()

# Graphique de perte
plt.subplot(1, 2, 2)
plt.plot(history_deeper.history['loss'], label='Entraînement')
plt.plot(history_deeper.history['val_loss'], label='Validation')
plt.title('Perte du modèle CNN profond')
plt.xlabel('Époque')
plt.ylabel('Perte')
plt.legend()

plt.tight_layout()
plt.savefig('deeper_cnn_training_history.png')
plt.show()

# Calcul des prédictions sur l'ensemble de validation
y_val_pred = deeper_cnn_model.predict(X_val_cnn)
y_val_pred_classes = np.argmax(y_val_pred, axis=1)
y_val_true_classes = np.argmax(y_val_onehot, axis=1)

# Matrice de confusion
plt.figure(figsize=(15, 15))
conf_mat = confusion_matrix(y_val_true_classes, y_val_pred_classes)
sns.heatmap(conf_mat, annot=False, cmap='Blues', xticklabels=range(num_classes), yticklabels=range(num_classes))
plt.title('Matrice de confusion du CNN profond sur l\'ensemble de validation')
plt.xlabel('Prédictions')
plt.ylabel('Valeurs réelles')
plt.tight_layout()
plt.savefig('deeper_cnn_confusion_matrix.png')
plt.show()

# Rapport de classification
print(classification_report(y_val_true_classes, y_val_pred_classes))
```

FIGURE 4.4 – Code de visualisation des résultats.

Annexe 5 : Annexe MLP

— Précision et Perte du modèle

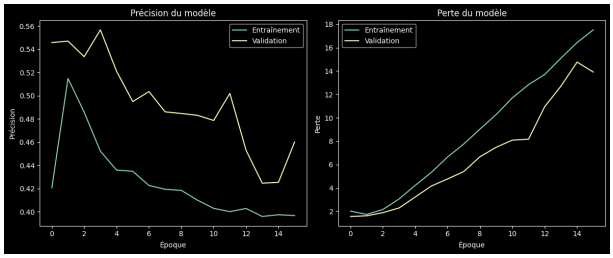

FIGURE 4.5 – Courbe de précision et de perte.

— Matrice de confusion

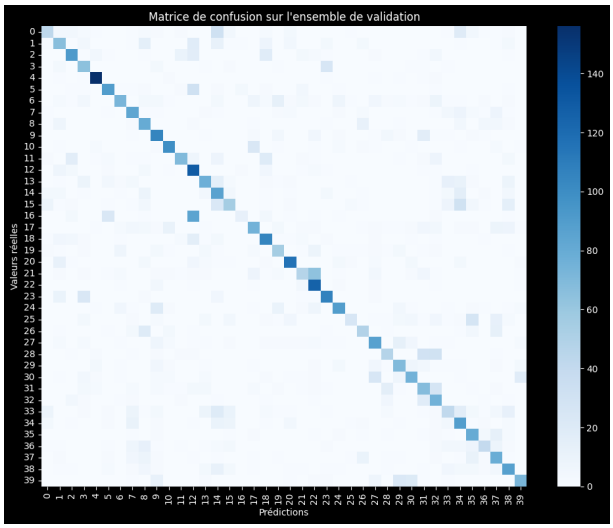

FIGURE 4.6 – Matrice de confusion.

— Résumé statistique du modèle

accuracy			0.55	5666
macro avg	0.57	0.54	0.53	5666
weighted avg	0.57	0.55	0.54	5666

FIGURE 4.7 – Résumé Statistique du modèle MLP.

Annexe 6 : Modèle CNN Simple

— Précision et Perte du modèle

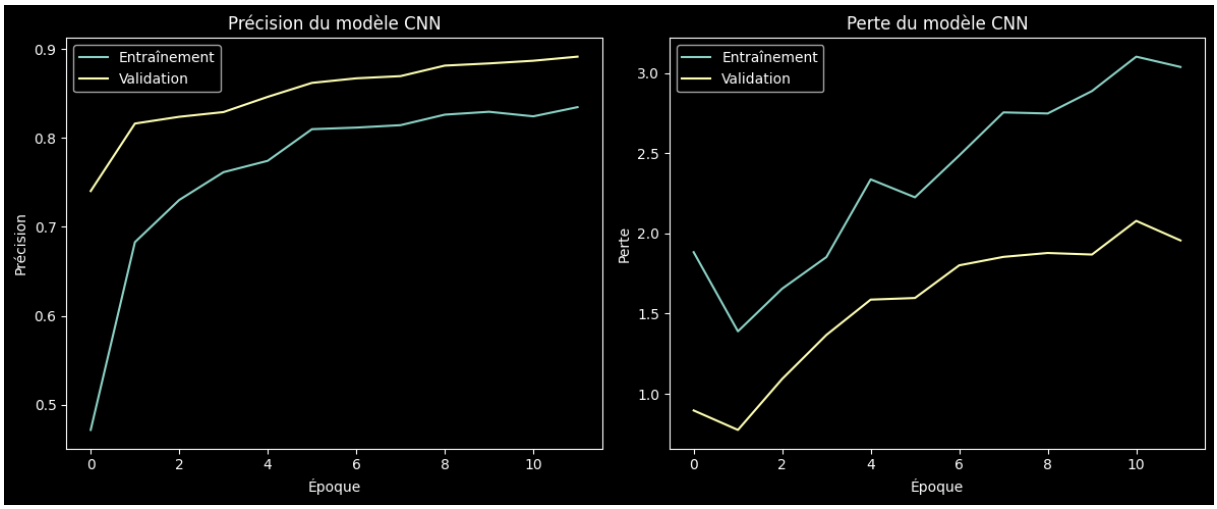

FIGURE 4.8 – Courbe de précision et de perte.

— Matrice de confusion

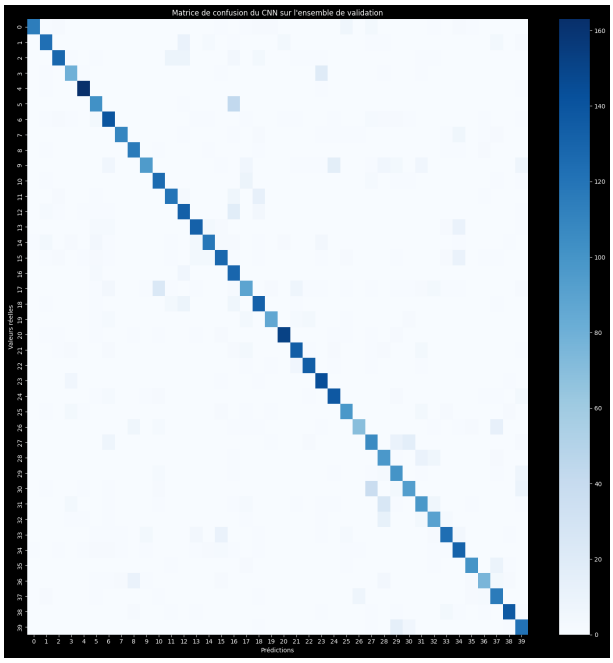

FIGURE 4.9 – Matrice de confusion.

— Résumé statistique du modèle

accuracy			0.82	5666
macro avg	0.83	0.81	0.82	5666
weighted avg	0.83	0.82	0.82	5666

FIGURE 4.10 – Résumé Statistique du modèle CNN Simple.

Annexe 7 : Modèle CNN Profonde

— Précision et Perte du modèle

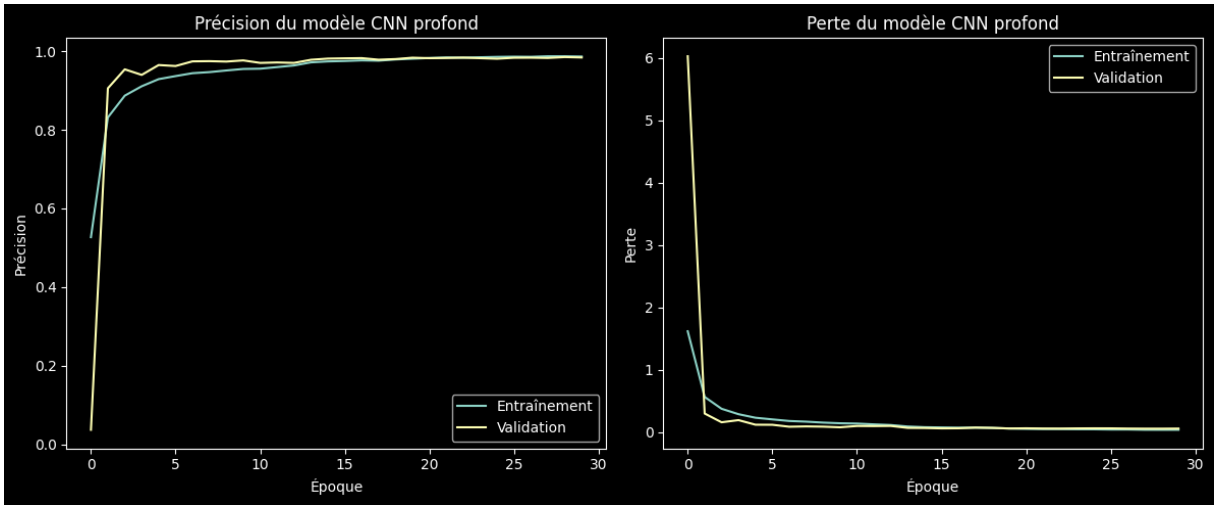

FIGURE 4.11 – Courbe de précision et de perte.

— Matrice de confusion

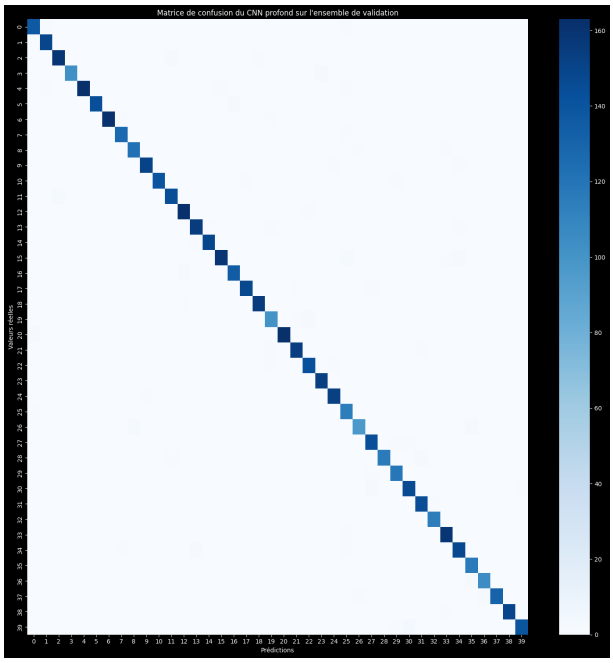

FIGURE 4.12 – Matrice de confusion.

— Résumé statistique du modèle

accuracy			0.98	5666
macro avg	0.98	0.98	0.98	5666
weighted avg	0.99	0.98	0.99	5666

FIGURE 4.13 – Résumé Statistique du modèle CNN Profonde.

Bibliographie

[1] Hazrat Ali et al. Urdu character recognition dataset, 2023. IST Deep Learning Workshop, Comsats University.

[2] François Chollet. *Deep Learning with Python*. Manning Publications, 2018.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[4] Abdur Rahman Maud. Ist deep learning workshop. `https://kaggle.com/competitions/istdlworkshop`, 2021. Kaggle.

[5] TensorFlow Team. Convolutional neural networks, 2023. URL : `https://www.tensorflow.org/tutorials/images/cnn`.

[6] Adjimon Jerome VITOFFODJI. Ist deep learning workshop, 2025. URL : `https://github.com/JeromeVitoff/IST-Deep-Learning-Workshop`.