



كلية العلوم
والتقنيات - مراكش
FACULTE DES SCIENCES
ET TECHNIQUES - MARRAKECH

UNIVERSITE CADI AYYAD
FACULTÉ DES SCIENCES ET TECHNIQUES
MARRAKECH

Master IAI

Extraction du texte des images à faible
résolution (OCR)

Réalisé par :

LAALIJI ZAKARIAE
HNIQUA ABDESSAMAD

Encadré par :

Mme. Amir

Année Universitaire : 2024/2025

Table des matières

1	Remerciements	5
2	Introduction et Contexte de projet	6
2.1	Introduction	6
2.2	Contexte et problématique de projet	6
2.2.1	Contexte	6
2.2.2	Problématique	6
2.2.3	Objectifs de projet	6
3	Définition du cahier de charges et Analyse des besoins	7
3.1	Introduction	7
3.2	Spécifications non techniques	7
3.3	Fonctionnalités techniques	9
4	Bases Théoriques des Transformations d'Images pour l'Évaluation OCR	10
4.1	Transformations Basées sur le Bruit	10
4.1.1	Bruit Gaussien	10
4.1.2	Bruit Sel et Poivre	11
4.2	Dégradations Structurelles	11
4.2.1	Trous Noirs (Défauts Localisés)	11
4.3	Variations Photométriques	12
4.3.1	Variations de Luminosité	12
4.3.2	Éblouissement (Glare)	12
4.3.3	Amélioration du Contraste	13
4.4	Dégradations par Compression et Échantillonnage	14
4.4.1	Compression JPEG	14
4.4.2	Redimensionnement avec Flou	14
4.5	Transformations Morphologiques	14
4.5.1	Érosion et Dilatation	15
4.5.2	Ouverture et Fermeture	15
4.6	Filtres de Flou	16
4.6.1	Flou Moyenneur	16
4.6.2	Flou Gaussien	16
4.6.3	Filtre Médian	16
4.7	Objectifs de Robustesse pour l'OCR	18
5	Approche proposée	18
5.1	Préparation des données	19
5.2	Processus de Conversion : Aplatissement (Flattening) Principe Mathématique	21
5.3	Exemple Détaillé avec une Matrice 4×4	22
5.4	Schéma Conceptuel pour une Image 32×32	22
5.5	Équilibrage du dataset	23
5.6	Encodage One-Hot OCR : Chiffres et Lettres	24
5.6.1	Ensemble des classes	24
5.6.2	Mapping des classes	24
5.7	Architecture du modèle	25

5.7.1	MobileNetV2 et transfert d'apprentissage pour OCR	25
5.7.2	Recherche d'hyperparamètres avec Keras Tuner	26
5.8	Entraînement et Validation	28
5.9	Évaluation sur Données de Validation	30
5.10	Phase de Test et Prédiction	32
6	Conception et Architecture	33
6.1	Introduction	33
6.2	Diagramme de cas d'utilisation	33
6.2.1	Acteur	33
6.2.2	Cas d'utilisation	33
6.3	Diagramme d'activité	34
6.3.1	Flux Principal :	36
6.4	Architecture du projet	37
7	Démonstration et fonctionnement de l'application	37
7.1	Page d'accueil	37
7.2	Section des fonctionnalités principales	38
7.3	Section "Comment ça marche"	38
7.4	Choix de la méthode de reconnaissance	39
7.5	Module de reconnaissance rapide de texte	40
7.6	Processus complet pas à pas	42
7.6.1	Étape 1 : Téléchargement de l'image	42
7.6.2	Étape 2 : Validation et traitement	43
7.6.3	Étape 3 : Transformations	44
7.6.4	Étape 4 : Segmentation des caractères	44
7.6.5	Étape 5 : Résultats de la reconnaissance	46
7.7	Support multilingue	46
7.8	Conclusion	47
8	Conclusion générale	48
9	Bibliographie	49

Liste des Figures

1	Exemple d'application du bruit gaussien sur un texte : image originale (gauche) et après ajout de bruit gaussien (droite)	10
2	Exemple d'application du bruit sel et poivre sur un texte : comparaison entre l'image originale et l'image dégradée par le bruit impulsif	11
3	Exemple de création de trous noirs dans un texte : simulation de zones d'encre manquante ou de détérioration du document	11
4	Exemple de variations de luminosité : image originale (centre), version assombrie (gauche) et version éclaircie (droite)	12
5	Exemple de simulation d'éblouissement : ajout de reflets localisés sur le texte pour tester la robustesse aux zones surexposées	12
6	Exemple d'égalisation d'histogramme : redistribution des niveaux de gris pour améliorer le contraste et la visibilité du texte	13
7	Exemple d'étirement de contraste : normalisation de la dynamique des niveaux de gris pour optimiser la distinction texte/arrière-plan	14
8	Exemple de compression JPEG : comparaison entre image originale et image compressée avec artefacts de quantification	14
9	Exemple de redimensionnement avec flou : dégradation de la résolution par sous-échantillonnage suivi d'un sur-échantillonnage	15
10	Exemple d'opérations morphologiques d'érosion et dilatation : modification de l'épaisseur des traits de caractères	15
11	Exemple d'opérations morphologiques d'ouverture et fermeture : traitement des connexions et coupures dans les caractères	16
12	Exemple de flou moyenneur : application d'un filtre moyenneur pour simuler les effets de mouvement	17
13	Exemple de flou gaussien : simulation d'une mise au point imparfaite avec un filtre gaussien	17
14	Exemple de filtre médian : réduction du bruit tout en préservant les contours des caractères	17
15	Schéma global de l'approche proposée pour l'OCR	19
16	Transformations d'Images pour l'Évaluation OCR exemple sur une observation	20
17	visualiser la distribution et la qualité des données	21
18	Dataset déséquilibré avant le Over-sampling	23
19	Dataset équilibré après le Over-sampling par la méthode SMOTE	24
20	Courbes d'entraînement et de validation du modèle OCR (précision et perte).	28
21	Matrice de confusion sur l'ensemble de validation du modèle OCR.	30
22	Répartition des prédictions correctes et incorrectes	32
23	Exemple de prédiction sur un caractère en niveaux de gris	33
24	Exemple de prédiction sur un caractère en RGB.	33
25	Diagramme de cas d'utilisation – Fonctionnalités principales	34
26	Diagramme d'activité	35
27	Page d'accueil de l'application OCR Vision	38
28	Section des fonctionnalités principales	38
29	Section explicative du fonctionnement	39
30	Interface de sélection de la méthode de reconnaissance	39

31	Interface de téléchargement pour la reconnaissance rapide	40
32	Résultats de la reconnaissance de texte rapide (partie 1)	41
33	Résultats de la reconnaissance de texte rapide (partie 2)	41
34	Interface de téléchargement d'image dans le processus complet	42
35	Validation et analyse de l'image	43
36	Transformations appliquées à l'image	44
37	Processus de segmentation des caractères (partie 1)	44
38	Processus de segmentation des caractères (partie 2)	45
39	Résultats de la reconnaissance OCR	46

1 Remerciements

Nous tenons à exprimer notre sincère gratitude à **Madame Laila Amir** pour son approche pédagogique exemplaire et son cours particulièrement pertinent dans le domaine du traitement d'images et de la vision par ordinateur. Nous la remercions vivement pour l'opportunité qu'elle nous a offerte de travailler sur des projets innovants, permettant d'appliquer concrètement les concepts théoriques dans un modèle d'apprentissage stimulant et enrichissant.

Nous remercions également l'ensemble du corps enseignant pour la qualité de la formation dispensée et les compétences transmises durant ce semestre agréable.

Enfin, notre reconnaissance s'adresse à nos camarades pour leur collaboration et leur soutien, ainsi qu'à toutes les personnes qui ont contribué, de près ou de loin, à l'aboutissement de ce projet.

2 Introduction et Contexte de projet

2.1 Introduction

Dans un monde où la numérisation et l'analyse des données visuelles jouent un rôle croissant, l'extraction automatique de texte à partir d'images, connue sous le nom de reconnaissance optique de caractères (OCR), est devenue une technologie clé. Cependant, les images de faible résolution, souvent rencontrées dans des contextes réels tels que les documents scannés de mauvaise qualité, les captures d'écran ou les photographies floues, posent des défis significatifs pour les systèmes OCR traditionnels. Ce projet s'inscrit dans le cadre du module de traitement d'images et vise à développer une solution robuste pour extraire du texte à partir d'images intentionnellement dégradées en résolution et en qualité. En combinant des techniques avancées de traitement d'images avec des outils OCR, ce projet cherche à démontrer l'efficacité de ces méthodes pour améliorer la reconnaissance de texte dans des conditions difficiles.

2.2 Contexte et problématique de projet

2.2.1 Contexte

La reconnaissance optique de caractères (OCR) est une technologie essentielle dans de nombreuses applications, telles que la numérisation de documents, la reconnaissance de plaques d'immatriculation, ou encore l'extraction d'informations à partir de photographies. Cependant, la qualité des images joue un rôle déterminant dans les performances des systèmes OCR. Dans des scénarios réels, les images peuvent être affectées par une faible résolution, du bruit, un flou ou des artefacts de compression, rendant l'extraction de texte complexe. Ce projet explore l'intégration de techniques de traitement d'images pour prétraiter et améliorer les images dégradées avant l'application de l'OCR. En dégradant intentionnellement des images de haute qualité, nous simulons ces conditions difficiles et évaluons l'impact des méthodes de prétraitement sur la précision de l'OCR. Ce travail s'appuie sur des outils open-source tels qu'OpenCV pour le traitement d'images et Tesseract pour l'OCR, tout en adoptant une approche systématique pour optimiser les performances.

2.2.2 Problématique

Comment extraire efficacement du texte à partir d'images à faible résolution, où la qualité est dégradée par des facteurs tels que le bruit, le flou ou la compression ? Quelles techniques de traitement d'images permettent d'améliorer la lisibilité du texte pour optimiser les performances de l'OCR dans ces conditions difficiles ? Ce projet cherche à répondre à ces questions en étudiant l'effet des méthodes de prétraitement, telles que le seuillage adaptatif, le débruitage et l'amélioration de la netteté, sur la précision de l'extraction de texte. Il vise également à quantifier l'amélioration apportée par ces techniques à travers des métriques comme le taux d'erreur de caractères (CER) et le taux d'erreur de mots (WER), tout en démontrant leur applicabilité à des cas d'usage réels.

2.2.3 Objectifs de projet

Les objectifs principaux de notre projet sont les suivants :

- Simuler des images à faible résolution en appliquant des techniques de dégradation contrôlée, telles que le redimensionnement, l'ajout de bruit, le flou et la compression JPEG.
- Développer un pipeline de traitement d'images incluant la conversion en niveaux de gris, le seuillage adaptatif, le débruitage et l'amélioration de la netteté pour optimiser les images avant l'OCR.
- Appliquer des techniques avancées de reconnaissance de caractères pour extraire le texte des images, en utilisant à la fois une approche de segmentation de caractères avec réseau de neurones (CNN) et une comparaison avec d'autres méthodes (comme Tesseract OCR).
- Évaluer les performances du système à travers des métriques de confiance et une analyse comparative des différentes approches de reconnaissance.
- Créer une interface web intuitive et éducative permettant aux utilisateurs de visualiser et comprendre chaque étape du processus OCR.
- Démontrer l'intérêt des techniques de traitement d'images pour améliorer l'extraction de texte dans des contextes réels, comme la lecture de documents anciens ou de captures d'écran de faible qualité.

Ces objectifs visent à créer un système OCR complet et performant, capable de fonctionner efficacement même avec des images de qualité variable, tout en offrant une valeur éducative sur les processus de vision par ordinateur.

3 Définition du cahier de charges et Analyse des besoins

3.1 Introduction

Le cahier des charges définit les exigences nécessaires à la réalisation d'un projet d'extraction de texte à partir d'images à faible résolution, dans le cadre du module de traitement d'images. Ce projet vise à concevoir un système robuste combinant des techniques avancées de traitement d'images et la reconnaissance optique de caractères (OCR) pour surmonter les défis posés par des images dégradées. Une interface web, développée avec React, HTML, CSS, Tailwind CSS et un backend Flask, sera intégrée pour permettre aux utilisateurs d'interagir facilement avec le système, de charger des images, d'appliquer des traitements et de visualiser les résultats. Cette section détaille les besoins non techniques, qui englobent les objectifs globaux et les contraintes, ainsi que les fonctionnalités techniques, qui précisent les composants et processus du système.

3.2 Spécifications non techniques

Le système répond aux besoins suivants :

- **Objectif principal** : Développer une solution complète d'extraction de texte à partir d'images, capable de fonctionner même avec des images de qualité variable, en utilisant des techniques avancées de traitement d'image et un modèle de reconnaissance de caractères. L'application offre une interface web intuitive et éducative

qui permet aux utilisateurs de visualiser et comprendre le processus OCR étape par étape.

- **Utilisateurs ciblés :**

- Étudiants et chercheurs en traitement d'images et vision par ordinateur, pouvant observer l'impact des différentes étapes de prétraitement et dégradation sur les performances de reconnaissance.
- Professionnels dans des domaines comme la numérisation de documents ou l'extraction de texte à partir de sources de qualité variable.
- Utilisateurs non techniques bénéficiant d'une interface simple pour charger des images et obtenir rapidement des résultats d'OCR.

- **Caractéristiques principales :**

- Interface bilingue (français/anglais) permettant une accessibilité internationale.
- Deux modes d'utilisation : reconnaissance rapide de texte ou processus complet avec visualisation détaillée
- Visualisation des caractères segmentés et du processus de reconnaissance.
- Comparaison des résultats entre différentes méthodes d'OCR.
- Fonctionnalités d'export des résultats (copie dans le presse-papier, téléchargement).

- **Contraintes :**

- Utilisation exclusive d'outils open-source (OpenCV, Flask, React, TailwindCSS) pour garantir l'accessibilité et la reproductibilité.
- Support des formats d'image courants (JPEG, PNG, BMP, TIFF).
- Interface web responsive et intuitive, compatible avec les navigateurs modernes.
- Traitement côté serveur pour optimiser les performances même sur des machines standards sans matériel spécialisé.
- Architecture modulaire permettant l'évolution et l'extension des fonctionnalités.

- **Livrables attendus :**

- Un pipeline complet de traitement d'images incluant prétraitement, segmentation et reconnaissance de caractères.
- Une application web complète avec interface utilisateur réactive permettant l'upload d'images, le traitement, et l'affichage des résultats.
- Visualisations détaillées du processus de reconnaissance pour une meilleure compréhension.
- Implémentation d'un système multilingue complet.
- Documentation technique intégrée expliquant chaque étape du processus OCR directement dans l'interface.

Cette solution offre à la fois un outil pratique d'OCR et une plateforme éducative permettant de comprendre les différentes étapes et défis de la reconnaissance optique de caractères.

3.3 Fonctionnalités techniques

Le système implémente les fonctionnalités techniques suivantes :

1. Dégradation contrôlée des images :

- Redimensionnement des images à une résolution standardisée (32x32 pixels) avec interpolation linéaire ou bicubique (`cv2.resize`).
- Ajout de bruit gaussien et de bruit sel et poivre pour simuler des conditions réelles et tester la robustesse du modèle (`cv2.randn`).
- Application d'un flou gaussien pour simuler des images prises dans des conditions de mise au point imparfaite (`cv2.GaussianBlur`).
- Ajustement de la luminosité pour tester la reconnaissance dans différentes conditions d'éclairage.

2. Prétraitement des images :

- Conversion en niveaux de gris pour simplifier l'analyse (`cv2.cvtColor`).
- Seuillage adaptatif pour améliorer le contraste entre le texte et l'arrière-plan (`cv2.adaptiveThreshold`).
- Application d'opérations morphologiques (fermeture) pour améliorer la qualité des caractères (`cv2.morphologyEx`).
- Filtrage pour réduire le bruit tout en préservant les contours des caractères.

3. Segmentation et reconnaissance des caractères (OCR) :

- Segmentation des caractères individuels à l'aide de la détection de contours (`cv2.findContours`).
- Filtrage des contours basé sur la taille et les proportions pour identifier les caractères.
- Prétraitement de chaque caractère segmenté (redimensionnement à 32x32 pixels, normalisation).
- Reconnaissance des caractères à l'aide d'un modèle CNN (MobileNetV2) entraîné.

4. Méthodes de reconnaissance multiples et comparaison :

- Reconnaissance directe sur l'image originale.
- Reconnaissance après prétraitement de l'image.
- Reconnaissance caractère par caractère avec assemblage.
- Evaluer les performances.

5. Interface web moderne et responsive :

- **Backend** : API Flask pour gérer :
 - L'upload d'images.
 - Le traitement (dégradation, prétraitement, segmentation, reconnaissance).
 - La transmission des résultats (images encodées en base64, texte extrait, visualisations).
 - Endpoint `/process_word` pour la reconnaissance de texte caractère par caractère.
- **Frontend** : Interface intuitive développée avec React et Tailwind CSS :

- Interface d'accueil informative présentant l'application et ses fonctionnalités.
- Module de reconnaissance rapide de texte à partir d'images (TextRecognition).
- Processus complet pas à pas pour visualiser chaque étape de l'OCR.
- Affichage des résultats avec visualisations des caractères segmentés.
- Fonctionnalités d'export des résultats (copie dans le presse-papier, téléchargement).
- Support multilingue complet (anglais/français) via système de traduction.
- Interface responsive adaptée aux différents appareils.
- Internationalisation complète :
 - Support de l'anglais et du français dans toute l'application.
 - Système de traduction contextuel pour tous les éléments de l'interface.
 - Sélecteur de langue accessible depuis l'en-tête.

Cette implémentation offre à la fois une solution d'OCR fonctionnelle et un outil pédagogique permettant de visualiser et comprendre le processus de reconnaissance optique de caractères.

4 Bases Théoriques des Transformations d'Images pour l'Évaluation OCR

4.1 Transformations Basées sur le Bruit

4.1.1 Bruit Gaussien

Le bruit gaussien suit une distribution normale et modélise les imperfections du capteur :

$$I_{\text{bruit}}(x, y) = I_{\text{original}}(x, y) + \mathcal{N}(0, \sigma^2) \quad (1)$$

où $\mathcal{N}(0, \sigma^2)$ représente une distribution normale de moyenne nulle et de variance σ^2 .

Relation avec l'OCR : Simule les conditions réelles de numérisation où le bruit du capteur peut masquer les détails fins des caractères. Un OCR robuste doit maintenir sa précision malgré ce bruit additif.

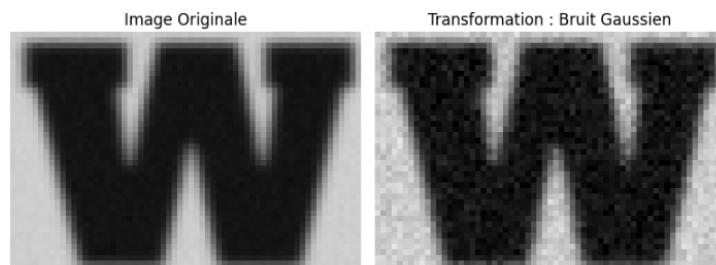


FIGURE 1 – Exemple d'application du bruit gaussien sur un texte : image originale (gauche) et après ajout de bruit gaussien (droite)

4.1.2 Bruit Sel et Poivre

Ce bruit impulsionnel affecte aléatoirement des pixels isolés :

$$I_{sp}(x, y) = \begin{cases} 0 & \text{avec probabilité } p_{poivre} \\ 255 & \text{avec probabilité } p_{sel} \\ I_{original}(x, y) & \text{sinon} \end{cases} \quad (2)$$

Relation avec l'OCR : Modélise les pixels défectueux et les artefacts de compression. Teste la capacité de l'OCR à ignorer les défauts ponctuels tout en préservant la structure globale des caractères.

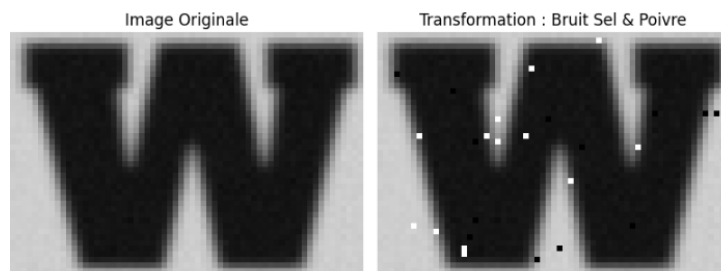


FIGURE 2 – Exemple d'application du bruit sel et poivre sur un texte : comparaison entre l'image originale et l'image dégradée par le bruit impulsionnel

4.2 Dégradations Structurelles

4.2.1 Trous Noirs (Défauts Localisés)

Simulation de zones d'encre manquante ou de détérioration :

$$I_{trous}(x, y) = I_{original}(x, y) \cdot M(x, y) \quad (3)$$

où $M(x, y)$ est un masque binaire définissant les zones affectées.

Relation avec l'OCR : Évalue la capacité de reconstruction contextuelle. Un bon OCR doit pouvoir inférer les parties manquantes d'un caractère à partir du contexte linguistique et de la forme partielle.

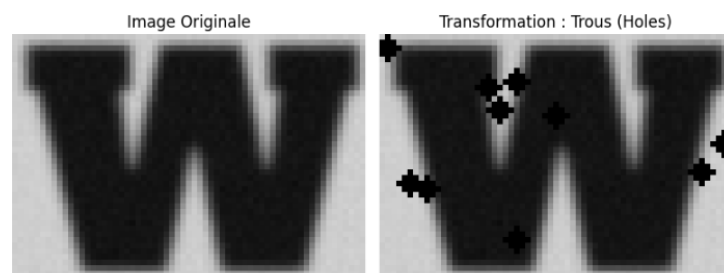


FIGURE 3 – Exemple de création de trous noirs dans un texte : simulation de zones d'encre manquante ou de détérioration du document

4.3 Variations Photométriques

4.3.1 Variations de Luminosité

Modélisation des conditions d'éclairage variables :

$$I_{bright}(x, y) = \min(255, I_{original}(x, y) + \alpha) \quad (4)$$

$$I_{dark}(x, y) = \max(0, I_{original}(x, y) - \beta) \quad (5)$$

Relation avec l'OCR : Teste l'adaptabilité aux conditions d'éclairage réelles. L'OCR doit maintenir sa performance indépendamment des variations globales d'intensité lumineuse.

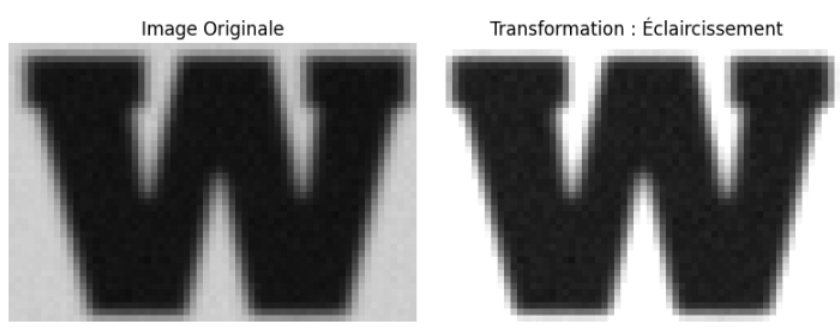


FIGURE 4 – Exemple de variations de luminosité : image originale (centre), version assombrie (gauche) et version éclaircie (droite)

4.3.2 Éblouissement (Glare)

Simulation de reflets localisés :

$$I_{glare}(x, y) = I_{original}(x, y) + G(x, y) \cdot e^{-\frac{(x-x_0)^2 + (y-y_0)^2}{2\sigma^2}} \quad (6)$$

où $G(x, y)$ représente l'intensité du reflet et le terme exponentiel sa distribution spatiale.

Relation avec l'OCR : Évalue la capacité à ignorer les zones surexposées tout en exploitant les informations disponibles dans les régions non affectées.

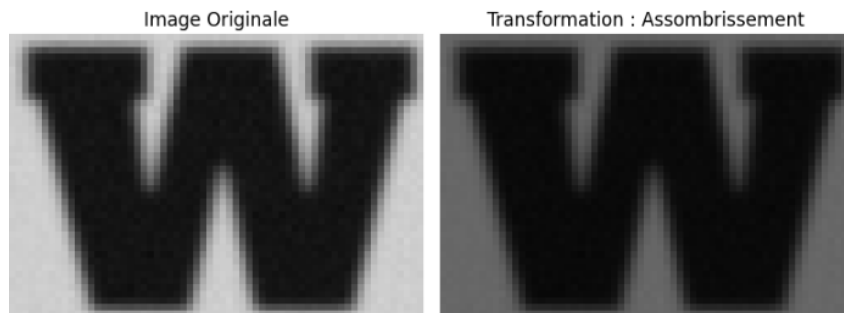


FIGURE 5 – Exemple de simulation d'éblouissement : ajout de reflets localisés sur le texte pour tester la robustesse aux zones surexposées

4.3.3 Amélioration du Contraste

Égalisation d'Histogramme L'égalisation d'histogramme redistribue les niveaux de gris pour maximiser l'utilisation de la dynamique :

$$I_{eq}(x, y) = \text{round} \left(\frac{L-1}{N \cdot M} \sum_{k=0}^{I(x,y)} n_k \right) \quad (7)$$

où n_k est le nombre de pixels de niveau k , $N \times M$ la taille de l'image, et L le nombre de niveaux de gris.

La fonction de transformation cumulative est :

$$T(r) = \frac{L-1}{N \cdot M} \sum_{i=0}^r n_i \quad (8)$$

Étirement de Contraste (Contrast Stretching) L'étirement linéaire du contraste normalise la dynamique des niveaux de gris :

$$I_{stretch}(x, y) = \frac{I(x, y) - I_{min}}{I_{max} - I_{min}} \times (L - 1) \quad (9)$$

où I_{min} et I_{max} sont respectivement les valeurs minimale et maximale de l'image originale.

Pour un étirement avec seuils personnalisés :

$$I_{stretch}(x, y) = \begin{cases} 0 & \text{si } I(x, y) < s_{low} \\ \frac{I(x,y)-s_{low}}{s_{high}-s_{low}} \times (L-1) & \text{si } s_{low} \leq I(x, y) \leq s_{high} \\ L-1 & \text{si } I(x, y) > s_{high} \end{cases} \quad (10)$$

Relations avec l'OCR :

- **Égalisation** : Améliore la visibilité des détails dans les images à faible contraste, teste la capacité de l'OCR à traiter des textes sur des arrière-plans uniformes ou des documents anciens

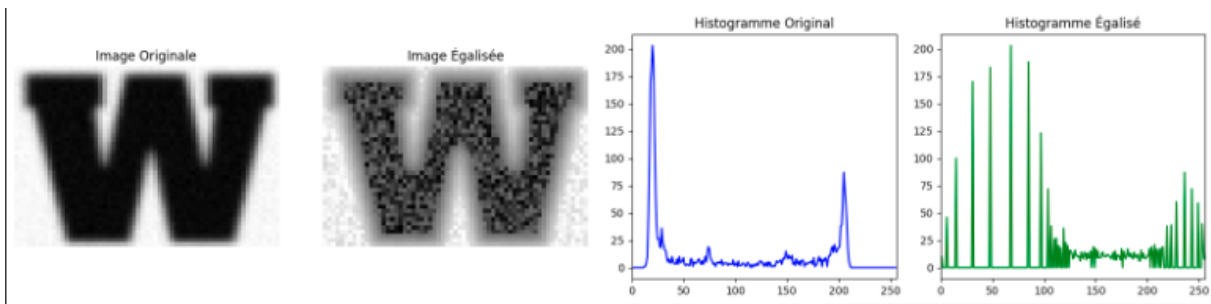


FIGURE 6 – Exemple d'égalisation d'histogramme : redistribution des niveaux de gris pour améliorer le contraste et la visibilité du texte

- **Étirement** : Optimise l'utilisation de la gamme dynamique, évalue la robustesse aux variations de contraste global et la capacité à distinguer le texte de l'arrière-plan

Ces transformations testent la capacité de l'OCR à maintenir sa précision sur des documents présentant des problèmes de contraste, situation fréquente avec les documents numérisés, les photographies de texte, ou les documents dégradés par le temps.

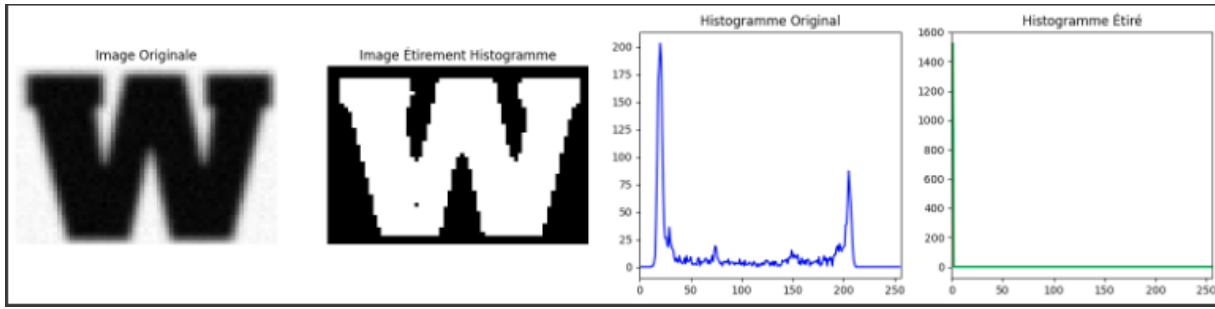


FIGURE 7 – Exemple d'étirement de contraste : normalisation de la dynamique des niveaux de gris pour optimiser la distinction texte/arrière-plan

4.4 Dégradations par Compression et Échantillonnage

4.4.1 Compression JPEG

La compression JPEG introduit des artefacts par quantification des coefficients DCT :

$$I_{jpeg} = \text{IDCT}(\text{Quantize}(\text{DCT}(I_{original}), Q)) \quad (11)$$

où Q est la matrice de quantification dépendant du facteur de qualité.

Relation avec l'OCR : Simule les conditions de documents numérisés avec compression. Teste la tolérance aux artefacts de blocs et à la perte d'information haute fréquence.

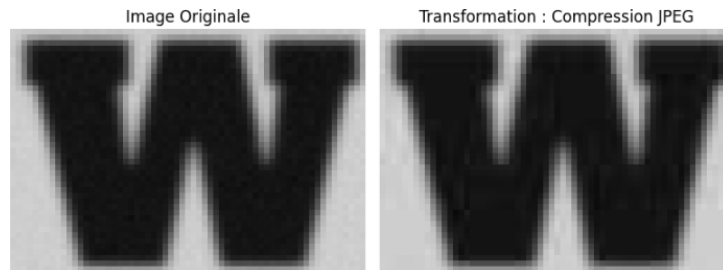


FIGURE 8 – Exemple de compression JPEG : comparaison entre image originale et image compressée avec artefacts de quantification

4.4.2 Redimensionnement avec Flou

Perte de résolution par sous-échantillonnage puis sur-échantillonnage :

$$I_{resize} = \text{Upsample}(\text{Downsample}(I_{original}, f), f) \quad (12)$$

Relation avec l'OCR : Modélise la perte de netteté due à l'optique ou au processus de numérisation. Évalue la robustesse aux caractères de faible résolution.

4.5 Transformations Morphologiques

Les opérations morphologiques utilisent un élément structurant S :

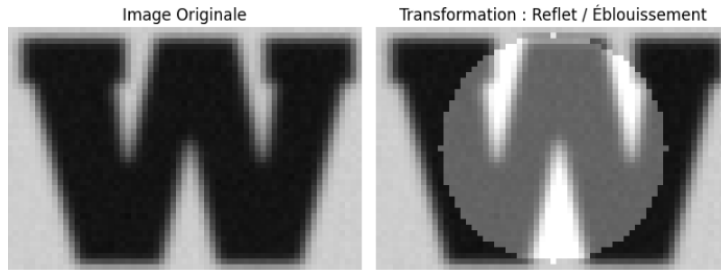


FIGURE 9 – Exemple de redimensionnement avec flou : dégradation de la résolution par sous-échantillonnage suivi d'un sur-échantillonnage

4.5.1 Érosion et Dilatation

$$\text{Érosion : } (I \ominus S)(x, y) = \min_{(s,t) \in S} I(x + s, y + t) \quad (13)$$

$$\text{Dilatation : } (I \oplus S)(x, y) = \max_{(s,t) \in S} I(x + s, y + t) \quad (14)$$

4.5.2 Ouverture et Fermeture

$$\text{Ouverture : } I \circ S = (I \ominus S) \oplus S \quad (15)$$

$$\text{Fermeture : } I \bullet S = (I \oplus S) \ominus S \quad (16)$$

Relation avec l'OCR :

- **Érosion** : Simule l'amincissement des traits, teste la reconnaissance de caractères fins
- **Dilatation** : Modélise l'épaississement, évalue la gestion des caractères gras

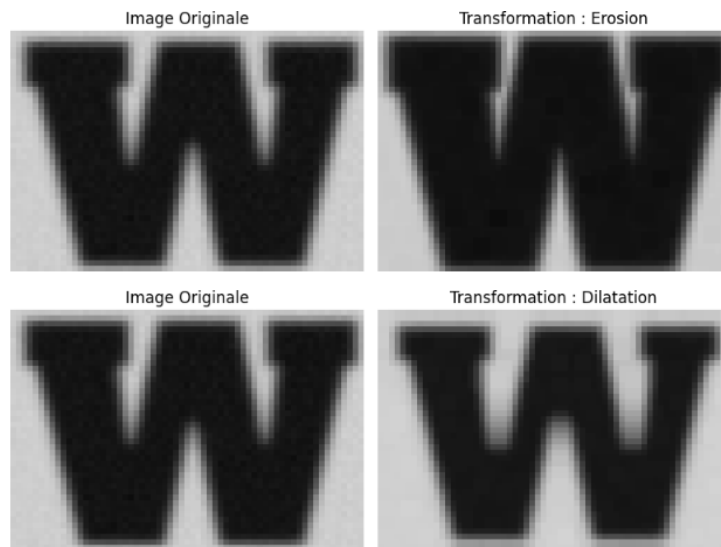


FIGURE 10 – Exemple d'opérations morphologiques d'érosion et dilatation : modification de l'épaisseur des traits de caractères

- **Ouverture** : Élimine les petites connexions, teste la segmentation
- **Fermeture** : Comble les petites coupures, évalue la reconstruction de caractères

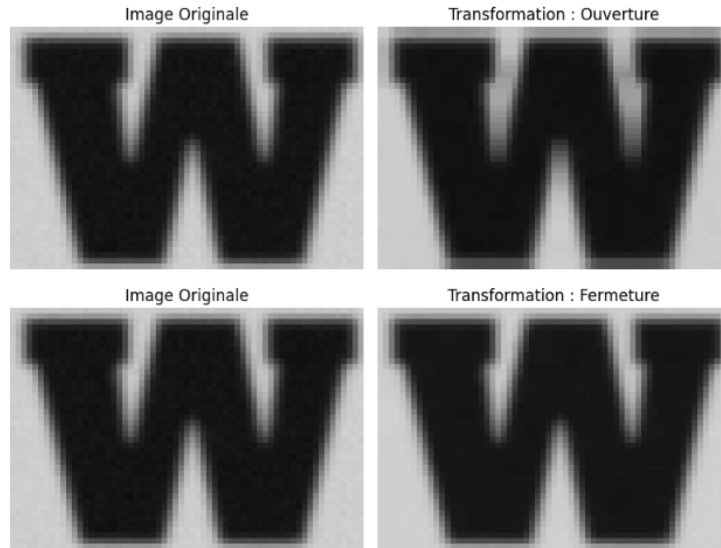


FIGURE 11 – Exemple d'opérations morphologiques d'ouverture et fermeture : traitement des connexions et coupures dans les caractères

4.6 Filtres de Flou

4.6.1 Flou Moyenneur

Convolution avec un noyau uniforme :

$$I_{mean}(x, y) = \frac{1}{n^2} \sum_{i=-k}^k \sum_{j=-k}^k I(x + i, y + j) \quad (17)$$

4.6.2 Flou Gaussien

Application d'un filtre gaussien :

$$I_{gauss}(x, y) = I(x, y) * G(x, y, \sigma) \quad (18)$$

où $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

4.6.3 Filtre Médian

Remplacement par la valeur médiane du voisinage :

$$I_{median}(x, y) = \text{median}\{I(x + i, y + j) : (i, j) \in W\} \quad (19)$$

Relations avec l'OCR :

- **Flou moyenneur** : Simule le mouvement, teste la tolérance au flou uniforme
- **Flou gaussien** : Modélise la mise au point imparfaite, évalue la robustesse optique
- **Filtre médian** : Préserve les contours tout en réduisant le bruit, teste la cohérence des formes

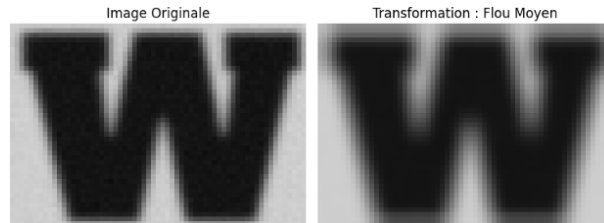


FIGURE 12 – Exemple de flou moyenneur : application d'un filtre moyenneur pour simuler les effets de mouvement

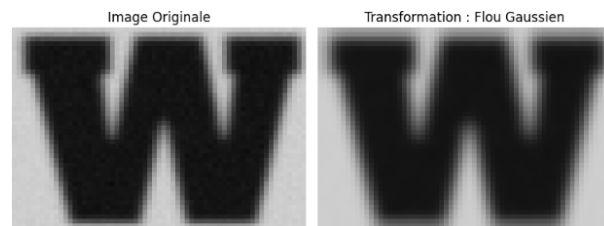


FIGURE 13 – Exemple de flou gaussien : simulation d'une mise au point imparfaite avec un filtre gaussien

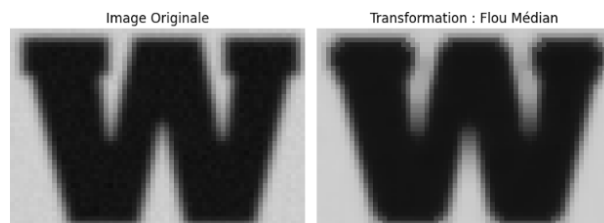


FIGURE 14 – Exemple de filtre médian : réduction du bruit tout en préservant les contours des caractères

4.7 Objectifs de Robustesse pour l'OCR

Ces transformations permettent d'évaluer différents aspects de la robustesse :

1. **Tolérance au bruit** : Capacité à maintenir la précision en présence de perturbations
2. **Adaptabilité photométrique** : Invariance aux variations d'éclairage
3. **Robustesse géométrique** : Reconnaissance malgré les déformations
4. **Résistance à la compression** : Performance sur des images de qualité réduite
5. **Reconstruction contextuelle** : Capacité à compenser les informations manquantes

L'ensemble de ces tests forme un protocole d'évaluation complet pour valider la fiabilité d'un système OCR dans des conditions réelles d'utilisation.

5 Approche proposée

La figure 19 illustre l'approche globale adoptée pour la reconnaissance optique de caractères (OCR) basée sur un réseau de neurones convolutifs optimisé. Cette méthode s'articule autour de plusieurs étapes clés, allant de la préparation des données jusqu'à la prédiction finale. Les images contenant des lettres et chiffres subissent d'abord une série de transformations afin d'améliorer leur qualité et leur robustesse. Ces transformations incluent notamment la réduction du bruit, l'augmentation du contraste, et la modification des structures et conditions d'éclairage. Une fois les images standardisées et converties en vecteurs numériques, elles sont équilibrées via la technique de suréchantillonnage SMOTE pour corriger les déséquilibres de classes. Les données sont ensuite encodées et séparées en ensembles d'entraînement 80% et de validation 20%. L'architecture MobileNetV2, une variante légère et efficace de CNN, est utilisée pour l'apprentissage. Un processus de recherche d'hyperparamètres est intégré pour optimiser les performances du modèle en maximisant la précision. Le modèle est évalué à l'aide de métriques comme l'accuracy, la précision, le rappel et le F1-score afin d'éviter le surapprentissage et de garantir une bonne généralisation. Finalement, la solution est capable de prédire efficacement des caractères à partir d'images en niveaux de gris ou en couleur (RGB), démontrant ainsi sa robustesse et sa polyvalence.

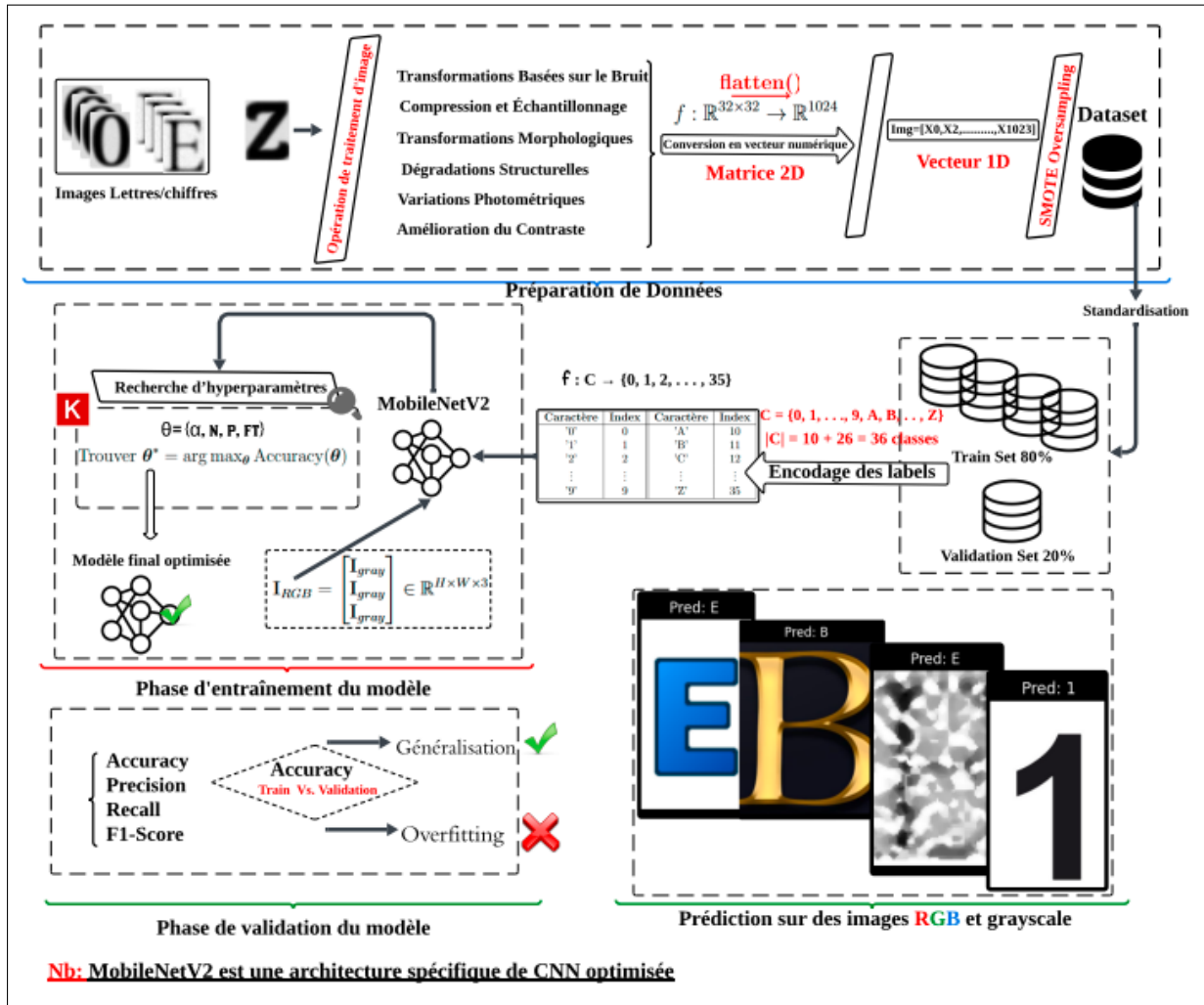


FIGURE 15 – Schéma global de l'approche proposée pour l'OCR

Remarque : *MobileNetV2 est une architecture optimisée de réseau de neurones convolutifs (CNN), conçue pour un bon compromis entre précision et efficacité computationnelle.*

5.1 Préparation des données

Les données utilisées sont des images de caractères (lettres et chiffres) extraites du dataset OCR de Kaggle. Chaque image est soumise à une série de transformations d'augmentation pour améliorer la robustesse du modèle :

- **Transformations basées sur le bruit** (bruit gaussien, sel et poivre),
- **Transformations morphologiques** (érosion, dilatation, ouverture, fermeture),
- **Variations photométriques** (luminosité, contraste),
- **Compression et échantillonnage** (JPEG, flou),
- **Dégradations structurales** (trous noirs, artefacts simulés).

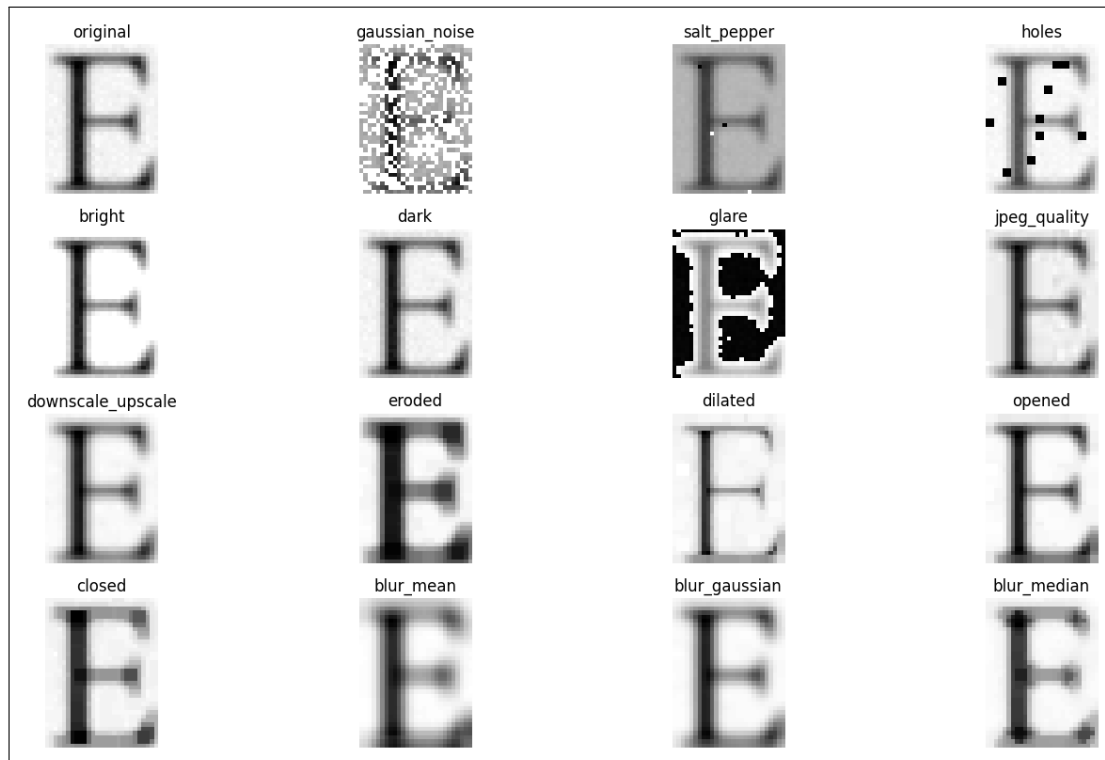


FIGURE 16 – Transformations d’Images pour l’Évaluation OCR exemple sur une observation



FIGURE 17 – visualiser la distribution et la qualité des données

Affichage d'exemples par label Pour visualiser la distribution et la qualité des données, un exemple d'image a été affiché pour chaque classe présente dans le jeu de données. Cette visualisation permet de s'assurer de la cohérence des labels, de repérer d'éventuelles anomalies et d'observer la diversité des caractères.

5.2 Processus de Conversion : Aplatissement (Flattening) Principe Mathématique

L'aplatissement consiste à transformer la matrice bidimensionnelle $I_{32 \times 32}$ en un vecteur unidimensionnel \mathbf{v}_{1024} :

$$f : R^{32 \times 32} \rightarrow R^{1024} \quad (20)$$

$$I_{i,j} \rightarrow \mathbf{v}_k \quad \text{où} \quad k = i \times n + j \quad (21)$$

Ce parcours suit un ordre ****ligne par ligne**** (row-major order), ce qui signifie que chaque ligne est lue intégralement avant de passer à la suivante.

5.3 Exemple Détaillé avec une Matrice 4×4

Considérons une image simplifiée de 4×4 pixels :

Image 4×4

Ligne 0	45	78	123	200
Ligne 1	12	156	89	234
Ligne 2	67	34	178	91
Ligne 3	145	203	56	112

flatten()

Représentation matricielle :

$$I = \begin{pmatrix} 45 & 78 & 123 & 200 \\ 12 & 156 & 89 & 234 \\ 67 & 34 & 178 & 91 \\ 145 & 203 & 56 & 112 \end{pmatrix} \quad (22)$$

Après aplatissement :

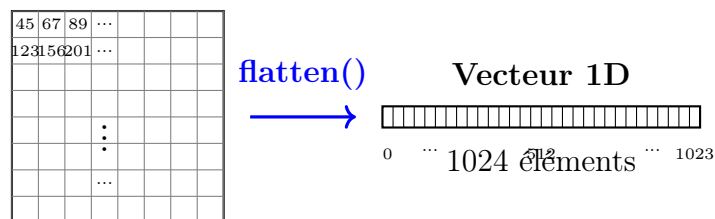
Vecteur 1D (16 éléments)

45	78	123	200	12	156	89	234	67	34	178	91	145	203	56	112
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Vecteur résultant :

$$\mathbf{v} = [45, 78, 123, 200, 12, 156, 89, 234, 67, 34, 178, 91, 145, 203, 56, 112] \quad (23)$$

5.4 Schéma Conceptuel pour une Image 32×32

Image 32×32 

1024 pixels au total

Les images sont ensuite converties en niveaux de gris, redimensionnées à 32×32 pixels, puis transformées en vecteurs 1D de dimension 1024 par la fonction `flatten()`. Ce processus permet de standardiser les entrées du modèle.

Index	0	1	2	3	4	5	6	7	8	9	...	1015	1016	1017	1018	1019	1020	1021	1022	1023	Label
0	178	178	178	171	157	140	135	131	131	131	...	65	65	64	64	64	65	74	98	98	Z
1	217	217	217	217	216	215	214	213	212	210	...	169	169	169	170	175	182	195	203	208	2
2	94	94	94	94	93	93	94	95	94	95	...	67	68	66	66	67	71	74	82	90	M
3	220	220	221	221	219	216	211	201	190	178	...	185	195	203	210	215	218	219	220	221	5
4	198	198	191	191	186	184	184	183	185	186	...	151	150	148	148	148	158	171	185	185	E

TABLE 1 – Extraits de vecteurs de caractéristiques normalisés associés à différentes étiquettes

5.5 Équilibrage du dataset

Pourquoi utiliser SMOTE ? SMOTE (Synthetic Minority Over-sampling Technique) est une technique d'augmentation de données qui permet de corriger le déséquilibre entre classes en générant de nouveaux exemples synthétiques pour les classes minoritaires. Contrairement à un simple sur-échantillonnage par duplication, SMOTE crée des exemples intermédiaires en interpolant linéairement entre une instance minoritaire existante x_i et l'un de ses k plus proches voisins x_{voisin} :

$$\text{Exemple synthétique} = x_i + \delta \cdot (x_{\text{voisin}} - x_i), \quad \delta \in [0, 1]$$

Exemple : Soit $x_i = (2, 3)$ et un voisin $x_{\text{voisin}} = (4, 5)$. Si on choisit $\delta = 0.5$, alors l'exemple généré est :

$$x_{\text{synt}} = (2, 3) + 0.5 \cdot ((4, 5) - (2, 3)) = (3, 4)$$

Ainsi, SMOTE enrichit l'espace des classes rares avec des points cohérents et variés, ce qui aide le modèle à mieux les apprendre sans perdre d'information (contrairement à l'undersampling).

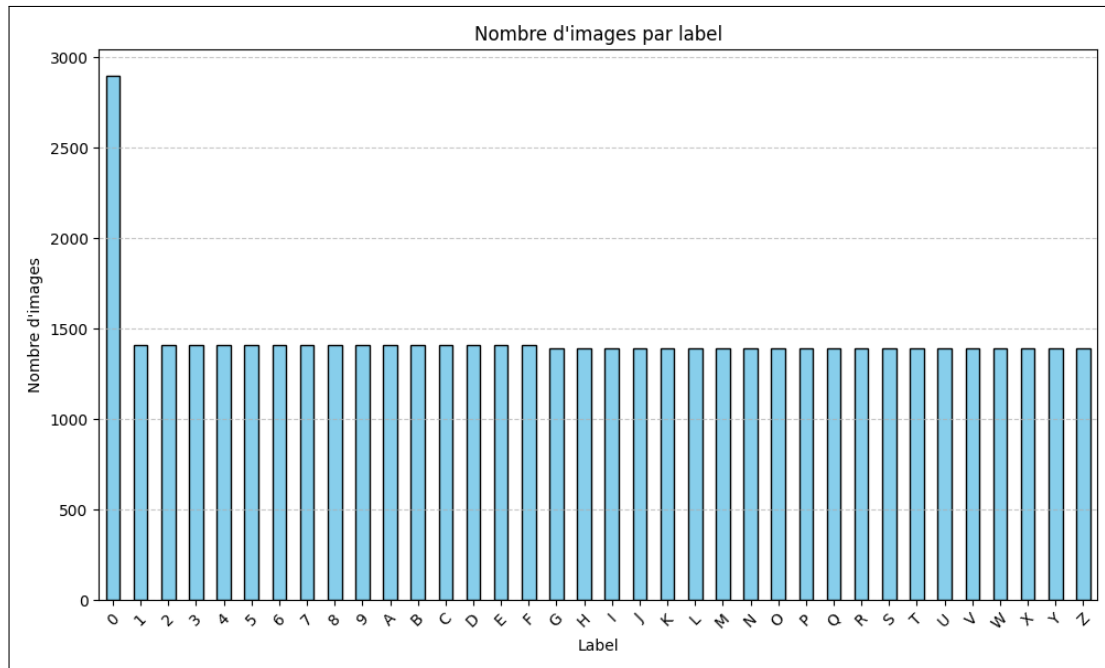


FIGURE 18 – Dataset déséquilibré avant le Over-sampling

Le dataset présente un déséquilibre important entre les classes. Pour pallier ce problème, la technique SMOTE (*Synthetic Minority Over-sampling Technique*) est appliquée

afin de générer des exemples synthétiques pour les classes sous-représentées, améliorant ainsi l'apprentissage sans recourir à une réduction du nombre d'exemples majoritaires.

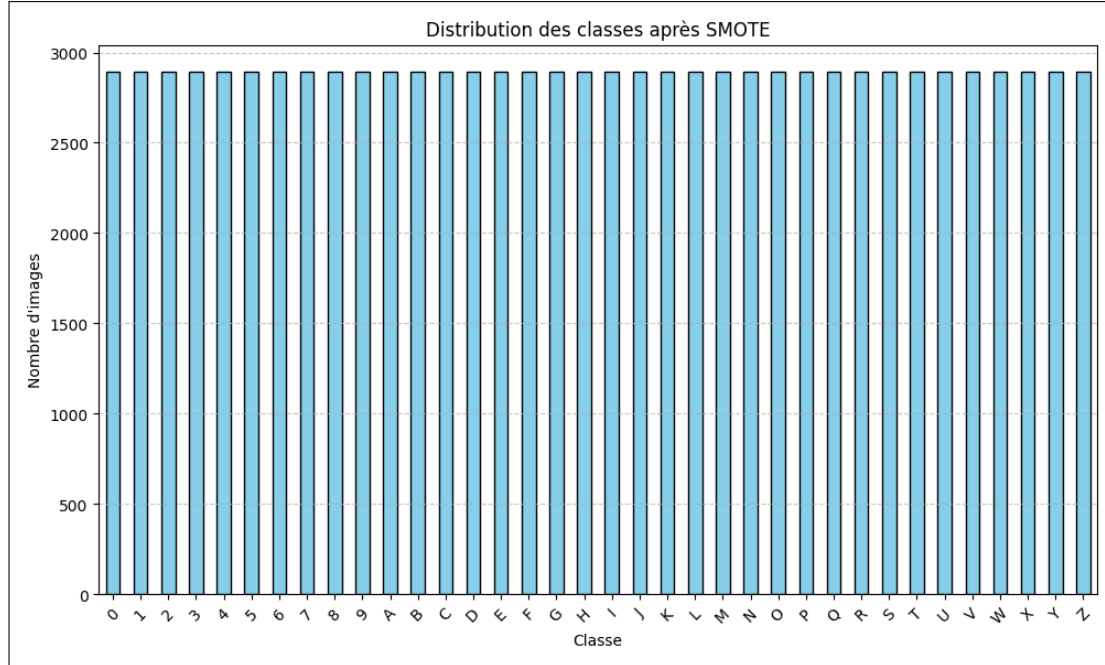


FIGURE 19 – Dataset équilibre après le Over-sampling par la méthode SMOTE

5.6 Encodage One-Hot OCR : Chiffres et Lettres

Dans un système de reconnaissance optique de caractères (OCR), chaque caractère (chiffre ou lettre) représente une classe distincte.

5.6.1 Ensemble des classes

Pour un système OCR reconnaissant chiffres et lettres majuscules :

$$\mathcal{C} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, \dots, Z\} \quad (24)$$

$$|\mathcal{C}| = 10 + 26 = 36 \text{ classes} \quad (25)$$

5.6.2 Mapping des classes

$$f : \mathcal{C} \rightarrow \{0, 1, 2, \dots, 35\} \quad (26)$$

Caractère	Index	Caractère	Index
'0'	0	'A'	10
'1'	1	'B'	11
'2'	2	'C'	12
⋮	⋮	⋮	⋮
'9'	9	'Z'	35

Les caractères sont codés sur un ensemble de 36 classes correspondant à :

$$C = \{0, 1, \dots, 9, A, B, \dots, Z\}, \quad |C| = 10 + 26 = 36$$

Chaque caractère est associé à un indice numérique unique utilisé pour l'entraînement du modèle.

5.7 Architecture du modèle

L'architecture utilisée est **MobileNetV2**, un CNN léger et optimisé, particulièrement adapté aux environnements contraints. Le modèle accepte aussi bien des images RGB que des images en niveaux de gris (converties en format compatible $R^{H \times W \times 3}$). Une phase de recherche d'hyperparamètres est réalisée (nombre de neurones, taux d'apprentissage, fonction de perte, etc.) pour optimiser la performance du modèle.

5.7.1 MobileNetV2 et transfert d'apprentissage pour OCR

Relation entre CNN et MobileNetV2

Architecture CNN générale Un CNN traite une image $\mathbf{I} \in R^{H \times W \times C}$ où :

- H = hauteur, W = largeur, C = nombre de canaux
- Opération de convolution : $\mathbf{Y} = \mathbf{I} * \mathbf{K} + b$
- \mathbf{K} = noyau de convolution, b = biais

MobileNetV2 comme CNN optimisé MobileNetV2 utilise des **convolutions séparables en profondeur** :

$$\text{Convolution standard : } \mathcal{O}(H \times W \times C_{in} \times C_{out} \times K^2) \quad (27)$$

$$\text{Convolution séparable : } \mathcal{O}(H \times W \times C_{in} \times K^2 + C_{in} \times C_{out}) \quad (28)$$

Réduction de complexité : $\frac{1}{C_{out}} + \frac{1}{K^2} \approx 8 - 9$ fois moins d'opérations.

Problème des canaux : Grayscale vs RGB

Incompatibilité dimensionnelle

$$\text{Image grayscale : } \mathbf{I}_{gray} \in R^{H \times W \times 1} \quad (29)$$

$$\text{MobileNetV2 attend : } \mathbf{I}_{RGB} \in R^{H \times W \times 3} \quad (30)$$

Problème : Incompatibilité des dimensions d'entrée !

Solution : Duplication de canaux Transformation mathématique :

$$\mathbf{I}_{RGB} = \begin{bmatrix} \mathbf{I}_{gray} \\ \mathbf{I}_{gray} \\ \mathbf{I}_{gray} \end{bmatrix} \in R^{H \times W \times 3} \quad (31)$$

Formellement, pour chaque pixel (i, j) :

$$\mathbf{I}_{RGB}(i, j, :) = [\mathbf{I}_{gray}(i, j), \mathbf{I}_{gray}(i, j), \mathbf{I}_{gray}(i, j)] \quad (32)$$

Avantages du transfert d'apprentissage

Extraction de caractéristiques pré-apprises MobileNetV2 pré-entraîné fournit des filtres $\mathbf{W}_{pre} \in R^{K \times K \times C \times F}$ optimisés pour :

- Détection de contours
- Reconnaissance de textures
- Identification de formes géométriques

Architecture finale

$$\text{MobileNetV2 (base)} : R^{224 \times 224 \times 3} \rightarrow R^{1280} \quad (33)$$

$$\text{Dense(36)} : R^{1280} \rightarrow R^{36} \quad (34)$$

$$\text{Softmax} : R^{36} \rightarrow [0, 1]^{36} \quad (35)$$

Fonction de perte pour OCR

Entropie croisée catégorielle avec 36 classes :

$$\mathcal{L} = -\frac{1}{N} \sum_{n=1}^N \sum_{i=0}^{35} y_{n,i} \log(\hat{y}_{n,i}) \quad (36)$$

où $\mathbf{y}_n \in \{0, 1\}^{36}$ est le vecteur one-hot et $\hat{\mathbf{y}}_n$ la prédiction.

Avantages de cette approche

1. **Efficacité computationnelle** : MobileNetV2 = 53% moins de paramètres que ResNet-50
2. **Réutilisation des features** : Caractéristiques visuelles pré-apprises sur ImageNet
3. **Convergence rapide** : Moins d'époques nécessaires grâce au pré-entraînement
4. **Performance OCR** : Excellente reconnaissance de caractères malgré la conversion grayscale→RGB

Résumé : La duplication de canaux permet d'utiliser MobileNetV2 (CNN optimisé) sur images grayscale, combinant efficacité computationnelle et performance OCR via le transfert d'apprentissage.

5.7.2 Recherche d'hyperparamètres avec Keras Tuner

Concept de la recherche automatique

Problème d'optimisation L'entraînement d'un CNN nécessite de choisir plusieurs hyperparamètres $\theta = \{\alpha, n, p, ft\}$:

$$\alpha : \text{Learning rate} \in [10^{-5}, 10^{-1}] \quad (37)$$

$$n : \text{Dense units} \in [32, 512] \quad (38)$$

$$p : \text{Dropout rate} \in [0.0, 0.7] \quad (39)$$

$$ft : \text{Fine-tune} \in \{\text{True}, \text{False}\} \quad (40)$$

Objectif : Trouver $\theta^* = \arg \max_{\theta} \text{Accuracy}(\theta)$

Espace de recherche L'espace total des combinaisons possibles est :

$$|\Theta| = |\mathcal{A}| \times |\mathcal{N}| \times |\mathcal{P}| \times |\mathcal{FT}| \approx 10^6 \text{ combinaisons} \quad (41)$$

Fonctionnement de Keras Tuner

Algorithme de recherche Keras Tuner utilise différentes stratégies :

1. **Random Search** : Échantillonnage aléatoire

$$\theta_i \sim \mathcal{U}(\Theta) \quad \forall i \in \{1, 2, \dots, N\} \quad (42)$$

2. **Bayesian Optimization** : Modélisation probabiliste

$$P(\text{Accuracy}|\theta) \sim \mathcal{GP}(\mu(\theta), k(\theta, \theta')) \quad (43)$$

3. **Hyperband** : Allocation adaptative des ressources

Processus itératif Pour chaque combinaison θ_i :

$$1. \text{ Construire le modèle : } M_i = \text{build_model}(\theta_i) \quad (44)$$

$$2. \text{ Entraîner : } M_i^* = \text{train}(M_i, \mathcal{D}_{\text{train}}) \quad (45)$$

$$3. \text{ Évaluer : } s_i = \text{evaluate}(M_i^*, \mathcal{D}_{\text{val}}) \quad (46)$$

$$4. \text{ Stocker : } \text{scores}[\theta_i] = s_i \quad (47)$$

Impact des hyperparamètres sur la performance

Learning Rate (α) Influence sur la convergence :

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}_t) \quad (48)$$

- α trop grand \Rightarrow Oscillations, divergence
- α trop petit \Rightarrow Convergence lente
- α optimal \Rightarrow Convergence stable et rapide

Dense Units (n) Capacité du modèle :

$$\text{Dense}(n) : R^{1280} \rightarrow R^n \rightarrow R^{36} \quad (49)$$

Trade-off : Plus de neurones = plus de capacité mais risque de surapprentissage.

Dropout Rate (p) Régularisation stochastique :

$$\mathbf{h}_{\text{dropout}} = \mathbf{h} \odot \mathbf{m}, \quad \mathbf{m} \sim \text{Bernoulli}(1 - p) \quad (50)$$

$p = 0$: Pas de régularisation, $p = 0.5$: Régularisation forte.

Fine-tuning (ft)

$$ft = \text{False} : \text{Geler MobileNetV2, entraîner seulement la tête} \quad (51)$$

$$ft = \text{True} : \text{Entraîner tout le réseau avec } \alpha_{\text{base}} < \alpha_{\text{head}} \quad (52)$$

Algorithme simplifié de Keras Tuner

Input : Espace de recherche Θ , Budget N **Output** : Meilleurs hyperparamètres θ^*
 $\text{best_score} \leftarrow 0$ $\theta^* \leftarrow \emptyset$ $i = 1$ to N $\theta_i \leftarrow \text{sample}(\Theta)$ $\text{model}_i \leftarrow \text{build_model}(\theta_i)$ $\text{model}_i \leftarrow \text{train}(\text{model}_i)$ $\text{score}_i \leftarrow \text{evaluate}(\text{model}_i)$ $\text{score}_i > \text{best_score}$ $\text{best_score} \leftarrow \text{score}_i$ $\theta^* \leftarrow \theta_i$ θ^*

Avantages de cette approche

1. **Automatisation** : Évite le réglage manuel fastidieux
2. **Exploration systématique** : Couvre efficacement l'espace des hyperparamètres

3. **Optimisation objective** : Maximise la métrique de validation

4. **Reproductibilité** : Résultats consistants et traçables

Pour optimiser les performances du modèle MobileNetV2, une recherche d'hyperparamètres a été effectuée à l'aide de Keras Tuner. L'objectif visé était de maximiser la précision sur le jeu de validation (`val_accuracy`). Deux configurations ont été testées.

Le meilleur résultat a été obtenu lors de l'essai **Trial 1**, avec une **précision de validation de 95.35%**, en activant le *fine-tuning*, et avec une couche dense de 256 neurones, un taux de dropout de 0.5 et un taux d'apprentissage de 0.0005. En comparaison, l'essai **Trial 0** (sans fine-tuning et avec un taux de dropout plus élevé) a abouti à une performance significativement inférieure (56.00%).

Essai	Fine-tuning	Unités Dense	Dropout	Taux d'apprentissage	val_accuracy
Trial 1	Oui	256	0.5	0.0005	0.9535
Trial 0	Non	256	0.7	0.001	0.5600

TABLE 2 – Résultats de la recherche d'hyperparamètres avec MobileNetV2

Résumé : Keras Tuner automatise la recherche des meilleurs hyperparamètres en testant systématiquement différentes combinaisons et en sélectionnant celle qui maximise la performance du modèle CNN.

5.8 Entraînement et Validation

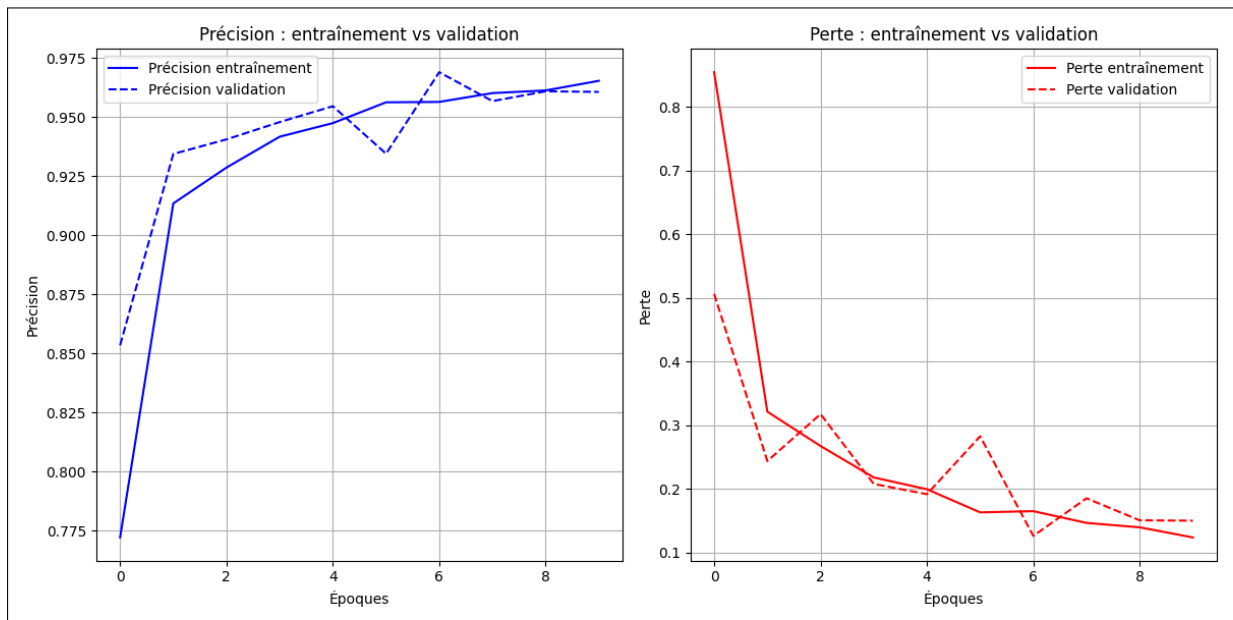


FIGURE 20 – Courbes d'entraînement et de validation du modèle OCR (précision et perte).

Les courbes de précision et de perte montrent que le modèle OCR apprend efficacement sans tomber dans l'overfitting.

Précision

Les courbes d'entraînement et de validation sont proches et augmentent régulièrement. Cela indique un apprentissage progressif et une bonne capacité de généralisation du modèle.

Perte

Les pertes diminuent pour les deux ensembles. Une légère instabilité est observée du côté validation, probablement en raison de données bruitées, mais les courbes restent globalement proches.

Conclusion

Le modèle est robuste et généralise bien. L'augmentation de données a permis d'améliorer sa résistance aux imperfections visuelles, notamment dans le cas d'images à faible résolution, sans détériorer ses performances globales.

5.9 Évaluation sur Données de Validation

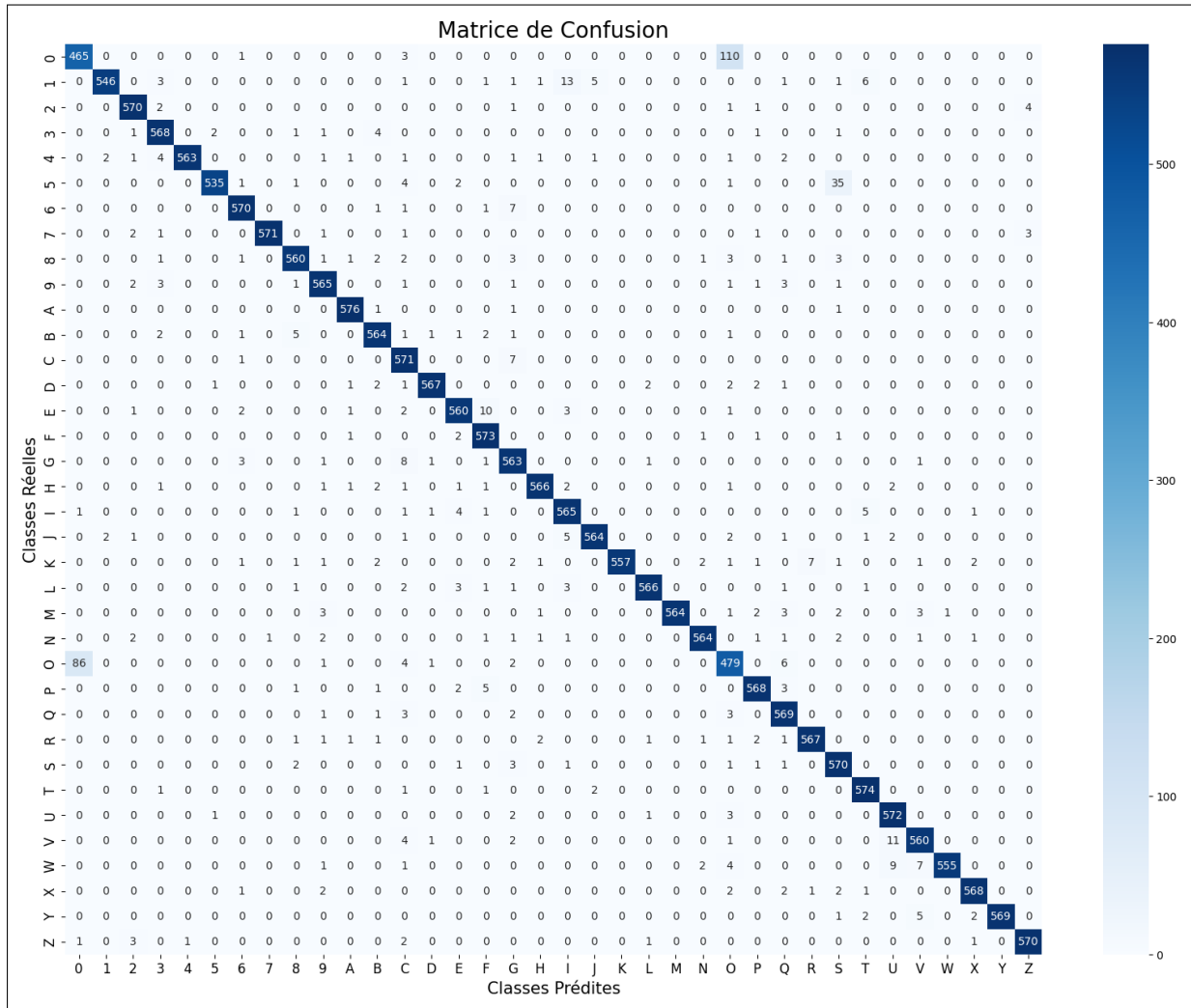


FIGURE 21 – Matrice de confusion sur l'ensemble de validation du modèle OCR.

Analyse de la matrice de confusion La matrice de confusion montre que le modèle atteint une excellente précision globale. La majorité des classes présentent un fort taux de prédictions correctes (valeurs diagonales supérieures à 550). Toutefois, quelques confusions subsistent, notamment : - $N \rightarrow O$: 110 erreurs - $Q \rightarrow O$: 86 erreurs

Ces confusions sont probablement dues à des similarités visuelles entre les lettres. Des techniques comme la data augmentation ciblée ou un meilleur prétraitement des images pourraient améliorer la différenciation de ces classes.

Classe	Précision	Rappel (Recall)	Spécificité	F1-score	Taux d'erreur
0	0.841	0.803	0.996	0.822	0.010
1	0.993	0.943	1.000	0.967	0.002
2	0.978	0.984	0.999	0.981	0.001
3	0.969	0.981	0.999	0.975	0.001
4	0.998	0.972	1.000	0.985	0.001
5	0.993	0.924	1.000	0.957	0.002
6	0.979	0.983	0.999	0.981	0.001
7	0.998	0.984	1.000	0.991	0.000
8	0.974	0.967	0.999	0.971	0.002
9	0.969	0.976	0.999	0.972	0.002
10 (A)	0.988	0.995	1.000	0.991	0.000
11 (B)	0.971	0.974	0.999	0.972	0.002
12 (C)	0.925	0.986	0.998	0.955	0.003
13 (D)	0.991	0.979	1.000	0.985	0.001
14 (E)	0.972	0.966	0.999	0.969	0.002
15 (F)	0.958	0.990	0.999	0.974	0.001
16 (G)	0.937	0.972	0.998	0.954	0.003
17 (H)	0.988	0.978	1.000	0.983	0.001
18 (I)	0.953	0.974	0.999	0.963	0.002
19 (J)	0.986	0.974	1.000	0.980	0.001
20 (K)	1.000	0.960	1.000	0.980	0.001
21 (L)	0.990	0.978	1.000	0.983	0.001
22 (M)	1.000	0.972	1.000	0.986	0.001
23 (N)	0.988	0.974	1.000	0.981	0.001
24 (O)	0.773	0.827	0.993	0.799	0.012
25 (P)	0.976	0.979	0.999	0.978	0.001
26 (Q)	0.955	0.983	0.999	0.969	0.002
27 (R)	0.986	0.979	1.000	0.983	0.001
28 (S)	0.918	0.983	0.997	0.949	0.003
29 (T)	0.973	0.991	0.999	0.982	0.001
30 (U)	0.960	0.988	0.999	0.974	0.001
31 (V)	0.969	0.967	0.999	0.968	0.002
32 (W)	0.998	0.959	1.000	0.978	0.001
33 (X)	0.988	0.981	1.000	0.984	0.001
34 (Y)	1.000	0.983	1.000	0.991	0.000
35 (Z)	0.988	0.984	1.000	0.986	0.001

TABLE 3 – Métriques de performance par classe : précision, rappel, spécificité, F1-score et taux d'erreur.

Exactitude globale : 0.9665

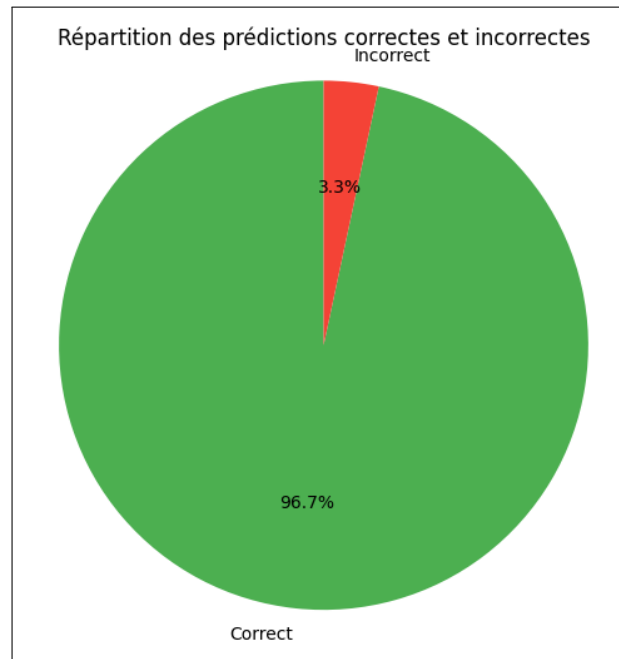


FIGURE 22 – Répartition des prédictions correctes et incorrectes .

Analyse de la répartition des prédictions Le diagramme circulaire montre que le modèle réalise **96,7 %** de prédictions correctes contre seulement **3,3 %** d'erreurs. Ces résultats témoignent d'une **excellente précision globale** et d'une bonne capacité de généralisation du modèle sur les données de test. Le faible taux d'erreur confirme son efficacité pour la tâche de classification.

5.10 Phase de Test et Prédiction

Une fois le modèle entraîné et validé, il a été testé sur des images de caractères inconnus, en couleur (RGB) et en niveaux de gris. Les prédictions montrent une reconnaissance efficace, même en présence de variations de style, de police ou d'altération visuelle.



FIGURE 23 – Exemple de prédiction sur un caractère en niveaux de gris .

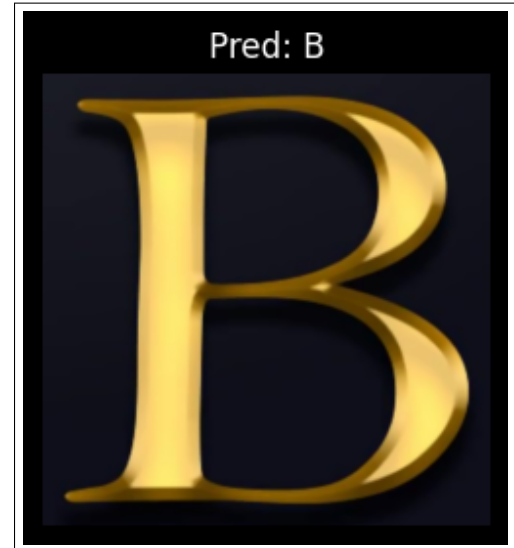


FIGURE 24 – Exemple de prédiction sur un caractère en RGB.

6 Conception et Architecture

6.1 Introduction

La phase de conception représente une étape cruciale dans le développement de notre solution mobile de détection d'incendie. Elle permet de traduire les besoins fonctionnels et techniques identifiés lors de l'analyse en modèles visuels concrets, facilitant la compréhension, la communication et l'implémentation.

6.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation décrit les interactions principales entre l'utilisateur et le système. Il permet d'identifier les fonctionnalités accessibles par l'utilisateur via l'application mobile, tout en mettant en évidence les dépendances fonctionnelles internes du système.

6.2.1 Acteur

- **Utilisateur** : Représente toute personne qui interagit avec le système, qu'il s'agisse d'un chercheur, d'un professionnel ou d'un utilisateur non technique.

6.2.2 Cas d'utilisation

1. **Charger une image** : L'utilisateur peut téléverser une image au système via l'interface web. Cette fonctionnalité constitue le point d'entrée principal du système.
2. **Dégrader une image** : L'utilisateur peut appliquer des dégradations contrôlées à l'image originale pour simuler des conditions difficiles comme :
 - Redimensionnement à basse résolution

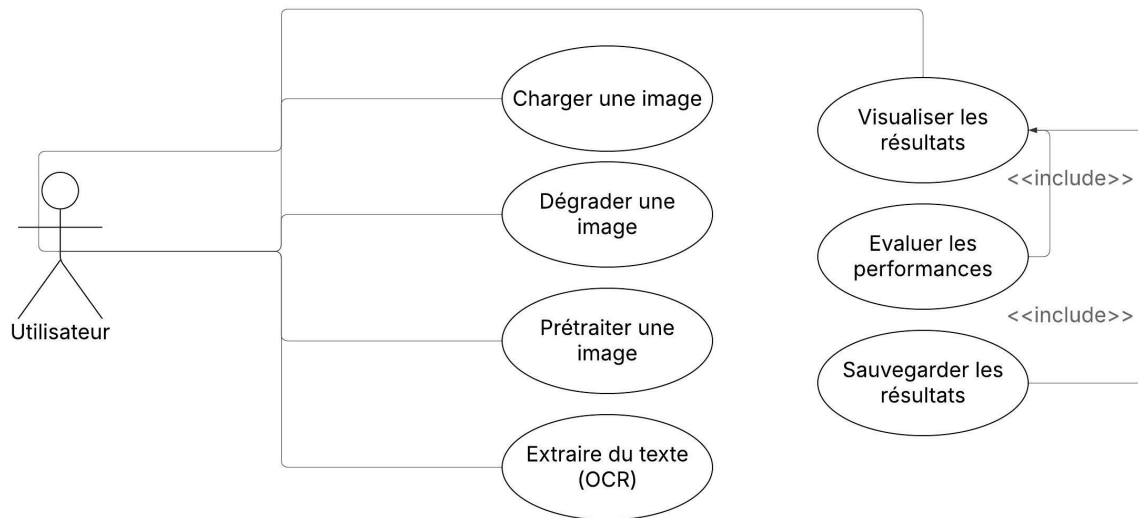


FIGURE 25 – Diagramme de cas d'utilisation – Fonctionnalités principales

- Ajout de bruit gaussien ou poivre et sel
 - Application de flou gaussien
 - Compression JPEG avec artéfacts
3. **Prétraiter une image** : L'utilisateur peut appliquer diverses techniques de pré-traitement pour améliorer la qualité de l'image avant l'OCR :
 - Conversion en niveaux de gris
 - Seuillage adaptatif
 - Débruitage avec filtres bilatéraux ou médians
 - Amélioration de la netteté
 4. **Extraire du texte (OCR)** : Le système utilise Tesseract OCR pour extraire le texte de l'image prétraitée ou non.
 5. **Visualiser les résultats** : L'utilisateur peut voir à l'écran :
 - L'image originale
 - L'image dégradée (si applicable)
 - L'image prétraitée (si applicable)
 - Le texte extrait
 6. **Comparer les performances** : Le système permet à l'utilisateur de comparer les résultats d'OCR obtenus avec différentes techniques de prétraitement en utilisant des métriques.
 7. **Sauvegarder les résultats** : L'utilisateur peut enregistrer les résultats obtenus.

Ce diagramme met en évidence l'étendue fonctionnelle du système et permet de comprendre rapidement les interactions possibles entre l'utilisateur et l'application.

6.3 Diagramme d'activité

Le diagramme d'activité modélise le flux de travail complet du système d'extraction de texte, depuis le chargement d'une image jusqu'à la sauvegarde des résultats. Il illustre la séquence des opérations et les points de décision dans le processus.

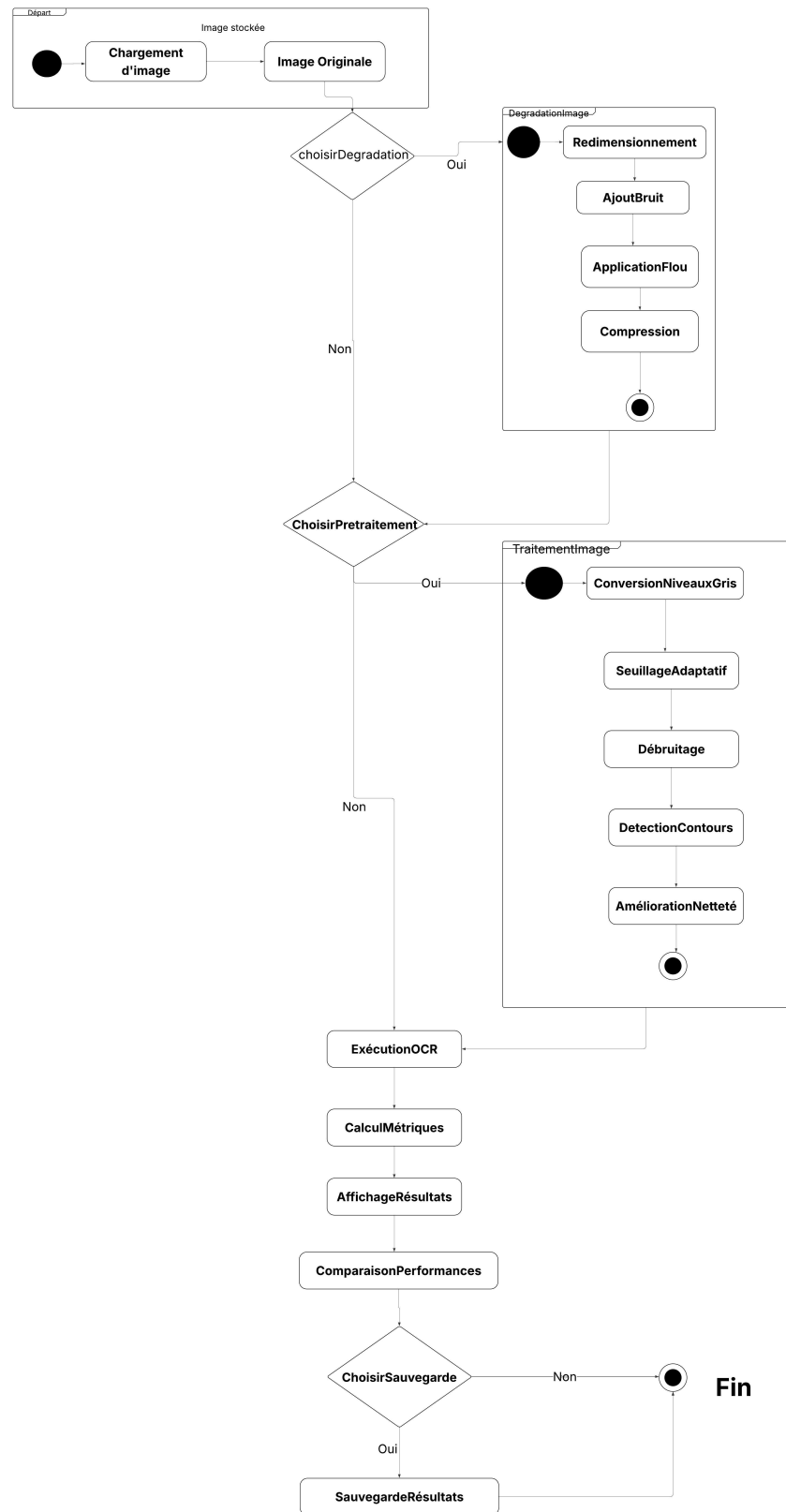


FIGURE 26 – Diagramme d'activité

6.3.1 Flux Principal :

1. Chargement d'image

- Point de départ du processus où l'utilisateur téléverse une image via l'interface web.
- L'image originale est stockée temporairement sur le serveur.

2. Choix de dégradation

- Premier point de décision où l'utilisateur choisit d'appliquer ou non une dégradation contrôlée à l'image.
- Si "Oui", l'image passe par le processus de dégradation.
- Si "Non", le flux passe directement à l'étape du choix de prétraitement.

3. Dégradation d'image (sous-état)

- Redimensionnement standardisé (typiquement 32x32 pixels).
- Ajout de bruit gaussien et sel et poivre pour simuler des conditions réelles.
- Application d'un flou gaussien pour simuler des images prises dans des conditions de mise au point imparfaite.
- Ajustement de la luminosité pour tester la reconnaissance dans différentes conditions d'éclairage.

4. Choix de prétraitement

- Deuxième point de décision où l'utilisateur choisit d'appliquer ou non des techniques de prétraitement.
- Si "Oui", l'image passe par le processus de prétraitement.
- Si "Non", l'image (originale ou dégradée) passe directement à l'OCR.

5. Prétraitement d'image (sous-état)

- Conversion en niveaux de gris pour simplifier l'analyse.
- Seuillage adaptatif pour améliorer le contraste entre le texte et l'arrière-plan.
- Débruitage avec application d'opérations morphologiques (fermeture).
- Détection des contours pour identifier les caractères.
- Segmentation des caractères individuels basée sur l'analyse des contours.

6. Exécution OCR

- Pour chaque caractère segmenté :
 - Redimensionnement à 32x32 pixels.
 - Normalisation des valeurs de pixels.
 - Traitement par le modèle CNN (MobileNetV2).
 - Obtention des prédictions et niveaux de confiance.
- Combinaison des caractères prédits en texte complet.
- Comparaison avec d'autres méthodes (comme Tesseract OCR).

7. Calcul des métriques

- Évaluation de la confiance de l'extraction.

8. Affichage des résultats

- Présentation à l'utilisateur :

- Image originale.
- Caractères segmentés avec visualisation.
- Texte extrait avec indicateurs de confiance.
- Comparaison entre différentes approches de reconnaissance.

9. Comparaison des performances

- Analyse des résultats obtenus avec différentes configurations :
 - Reconnaissance directe sur l'image originale.
 - Reconnaissance après prétraitement.
 - Reconnaissance caractère par caractère.
 - Comparaison avec Tesseract OCR.

10. Actions sur les résultats

- L'utilisateur peut :
 - Copier le texte reconnu dans le presse-papier.
 - Télécharger le texte sous forme de fichier texte.
- Option de traiter une nouvelle image (recommencer le processus).

Ce flux d'activité représente fidèlement l'enchaînement des opérations tel qu'il est implémenté dans l'application, tout en respectant la structure du diagramme d'activité initial. L'interface utilisateur offre une expérience guidée permettant de comprendre visuellement chaque étape du processus OCR, depuis le chargement de l'image jusqu'à la reconnaissance finale et l'export des résultats.

6.4 Architecture du projet

7 Démonstration et fonctionnement de l'application

Cette section présente l'interface utilisateur de notre application OCR Vision et détaille son fonctionnement à travers les différentes étapes du processus de reconnaissance de texte.

7.1 Page d'accueil

La page d'accueil (Figure 27) offre une introduction attrayante à l'application avec un design moderne dominé par un dégradé de couleur violette. Elle comprend :

- Un en-tête avec le logo OCRVision, une navigation claire (Accueil, Fonctionnalités, Comment ça marche) et un sélecteur de langue (FR/EN).
- Un titre principal "Extraction intelligente de texte" qui met en avant la fonction principale de l'application.
- Une description concise expliquant que l'application utilise des techniques avancées de traitement d'image et l'IA pour extraire du texte même à partir d'images à faible résolution.
- Un bouton d'appel à l'action "Commencer maintenant" pour débiter le processus.
- Une animation visuelle montrant une notification de succès "Texte extrait avec succès".



Fonctionnalités principales

FIGURE 27 – Page d'accueil de l'application OCR Vision

7.2 Section des fonctionnalités principales



FIGURE 28 – Section des fonctionnalités principales

La section des fonctionnalités principales (Figure 28) présente les trois capacités clés de l'application :

- **Traitement d'images avancé** : Conversion en niveaux de gris, seuillage adaptatif, et débruitage pour optimiser la qualité de l'image avant extraction.
- **OCR haute précision** : Utilisation de Tesseract OCR pour une reconnaissance de texte fiable, même sur des documents de qualité médiocre.
- **Analyse comparative** : Visualisation de la différence entre l'extraction directe et l'extraction avec prétraitement pour mesurer l'amélioration.

7.3 Section "Comment ça marche"

La section "Comment ça marche" (Figure 29) explique le processus en trois étapes principales :



FIGURE 29 – Section explicative du fonctionnement

1. **Acquisition d'image** : L'utilisateur télécharge une image contenant du texte. Le système analyse les propriétés de l'image : résolution, profondeur de couleur, et rapport signal/bruit.
2. **Prétraitement avancé** : Application de techniques avancées comme l'égalisation adaptative d'histogramme, la réduction de bruit et le seuillage adaptatif pour améliorer la visibilité du texte.
3. **Extraction OCR** : Le moteur OCR analyse l'image prétraitée pour identifier et extraire le texte, avec comparaison des résultats avant et après prétraitement.

Cette présentation étape par étape aide l'utilisateur à comprendre le flux de travail complet avant même de commencer à utiliser l'application.

7.4 Choix de la méthode de reconnaissance

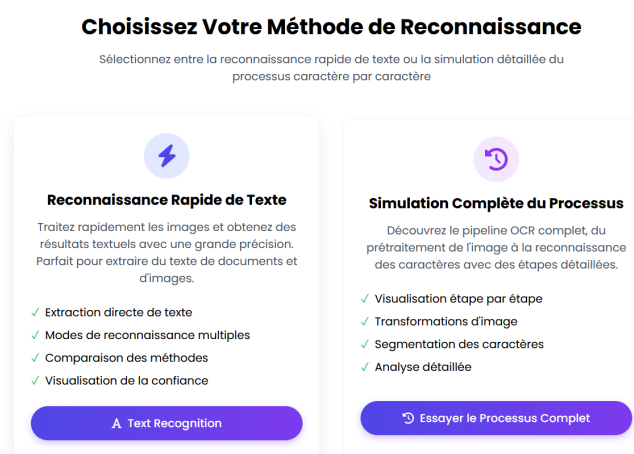


FIGURE 30 – Interface de sélection de la méthode de reconnaissance

Après avoir accédé à l'application, l'utilisateur est invité à choisir entre deux méthodes de reconnaissance (Figure 30) :

- **Reconnaissance Rapide de Texte** : Option optimisée pour obtenir rapidement des résultats textuels. Elle offre :
 - Extraction directe de texte
 - Modes de reconnaissance multiples
 - Comparaison des méthodes
 - Visualisation de la confiance
- **Simulation Complète du Processus** : Option pédagogique détaillant chaque étape du pipeline OCR. Elle propose :
 - Visualisation étape par étape
 - Transformations d'image
 - Segmentation des caractères
 - Analyse détaillée

Cette approche à deux voies permet de satisfaire à la fois les utilisateurs cherchant des résultats rapides et ceux souhaitant comprendre en profondeur le processus OCR.

7.5 Module de reconnaissance rapide de texte



FIGURE 31 – Interface de téléchargement pour la reconnaissance rapide

Le module de reconnaissance rapide (Figure 31) commence par une interface simple de téléchargement :

- Titre clair "Reconnaissance de Texte"
- Instructions indiquant à l'utilisateur de télécharger une image contenant du texte
- Zone de glisser-déposer avec icône de téléchargement
- Indication des formats supportés (JPG, PNG, BMP, TIFF)
- Bouton "Reconnaître le Texte" pour lancer l'analyse

Après le traitement, l'application affiche les résultats détaillés (Figures 32 et 33) :

- Le texte reconnu (dans l'exemple : "FST MARRAKECH MASTER IAI 2025")

Texte Reconnu

FST MARRAKECH MASTER IAI 2025

Confiance : 85.00%

Copier dans le Presse-papier

Télécharger en fichier texte

Image Originale

FST MARRAKECH MASTER IAI 2025

Visualisation des Caractères

F S T M A R U K E C H M A S T E R I A I 2 0 2 5

FST MARRAKECH MASTER IAI 2025

Visualisation du processus de reconnaissance des caractères

Processus de Reconnaissance des Caractères

1. **Prétraitement d'image** : L'image téléchargée est convertie en niveaux de gris, redimensionnée, et améliorée avec un seuillage adaptatif pour améliorer la visibilité du texte.
2. **Segmentation des Caractères** : L'image prétraitée est analysée pour identifier et isoler les caractères individuels à l'aide d'algorithmes de détection de contours.

FIGURE 32 – Résultats de la reconnaissance de texte rapide (partie 1)

détection de contours.

3. **Prétraitement des Caractères** :
 - Conversion en niveaux de gris
 - Redimensionnement à 32x32 pixels
 - Standardisation avec des valeurs de pixels entre 0 et 1
 - Restructuration pour correspondre aux exigences d'entrée du modèle
4. **Prédiction des Caractères** : Chaque caractère prétraité est soumis au modèle de réseau neuronal (MobileNetV2) pour prédire le caractère.
5. **Ordonnement des Caractères** : Les caractères prédits sont organisés en fonction de leurs positions spatiales dans l'image originale (triés par coordonnées x).
6. **Construction du Texte** : Les caractères ordonnés sont combinés pour former le texte final reconnu.

Comparaison des Méthodes

Méthode	Texte Reconnu	Confiance :
Direct (Original)	FST MARRAKECH MASTER IAI 2025	75.00%
Direct (Prétraité)	FST MARRAKECH MASTER IAI 2025	85.00%
Caractère par Caractère	FSTMARUKECHMASTERIAI2025	-
Segmentation de Mots	FST MARRAKECH MASTER IAI 2025	-
Tesseract OCR	FST MARRAKECH MASTER IAI 2025	-

Essayer une Autre Image

FIGURE 33 – Résultats de la reconnaissance de texte rapide (partie 2)

- Le niveau de confiance de la reconnaissance (85.00%)
- L'image originale à côté d'une visualisation colorée des caractères reconnus
- Une explication détaillée du processus de reconnaissance en 6 étapes
- Un tableau comparatif des différentes méthodes de reconnaissance utilisées :
 - Direct (Original)
 - Direct (Prétraité)
 - Caractère par Caractère
 - Segmentation de Mots
 - Tesseract OCR
- Des boutons pour copier le texte dans le presse-papier ou le télécharger en fichier texte
- Une option pour essayer une autre image

7.6 Processus complet pas à pas

Le processus complet (Figure ??) propose une expérience guidée en 6 étapes :

1. **Télécharger l'image** : Fournir l'image à traiter
2. **Valider et traiter** : Vérifier que l'image répond aux exigences
3. **Transformations** : Techniques de dégradation d'image
4. **Segmentation** : Segmentation des caractères
5. **Reconnaissance** : Reconnaissance des caractères par CNN
6. **Résultats** : Voir le texte extrait et l'analyse

Une barre de progression en haut de la page permet à l'utilisateur de suivre son avancement.

7.6.1 Étape 1 : Téléchargement de l'image

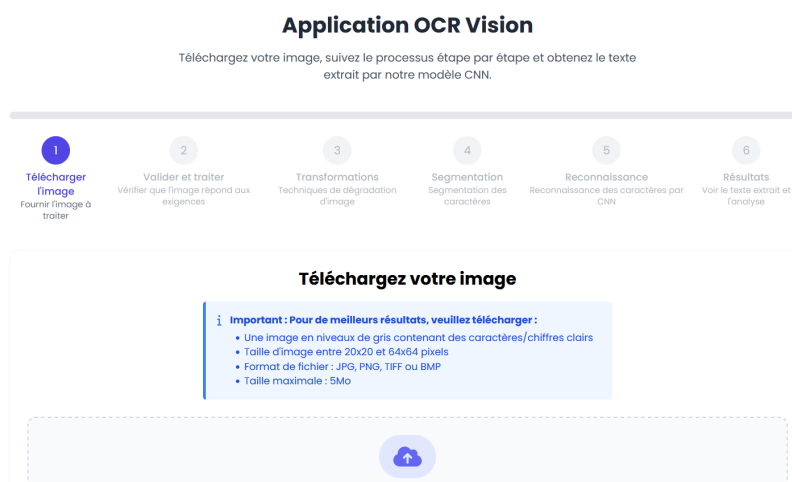


FIGURE 34 – Interface de téléchargement d'image dans le processus complet

L'étape de téléchargement (Figure 34) présente :

- Un encadré pour glisser-déposer une image
- Des consignes claires pour de meilleurs résultats :
 - Une image en niveaux de gris contenant des caractères/chiffres clairs
 - Taille d'image entre 20x20 et 64x64 pixels
 - Format de fichier : JPG, PNG, TIFF ou BMP
 - Taille maximale : 5Mo
- Un aperçu de l'image téléchargée (dans l'exemple, un caractère "K")
- Information sur le fichier sélectionné, sa taille et ses dimensions

7.6.2 Étape 2 : Validation et traitement

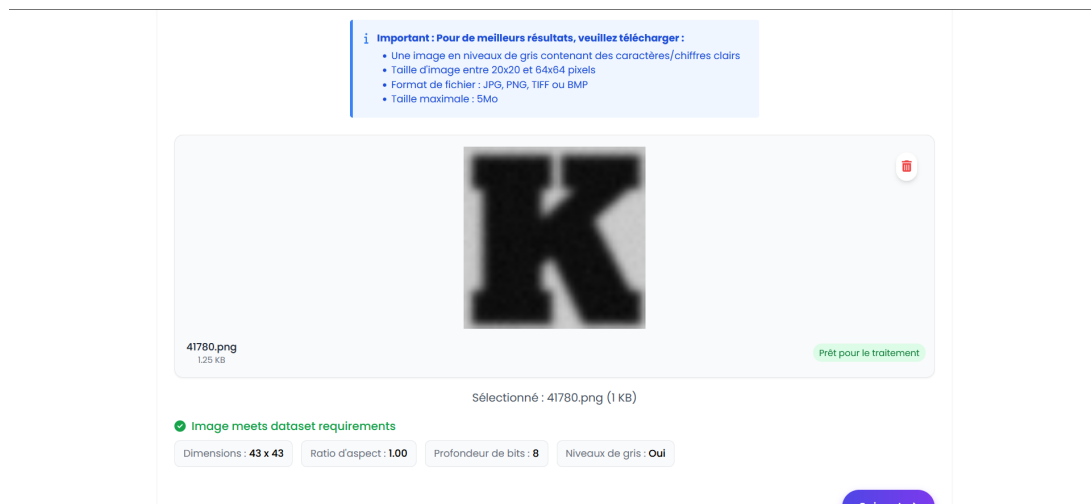


FIGURE 35 – Validation et analyse de l'image

L'étape de validation (Figure 35) affiche :

- Un message de confirmation si l'image répond aux exigences du dataset
- Les caractéristiques de l'image :
 - Dimensions : 43 x 43 pixels
 - Ratio d'aspect : 1.00
 - Profondeur de bits : 8
 - Niveaux de gris : Oui
- Une indication visuelle (coche verte) confirmant la compatibilité

7.6.3 Étape 3 : Transformations

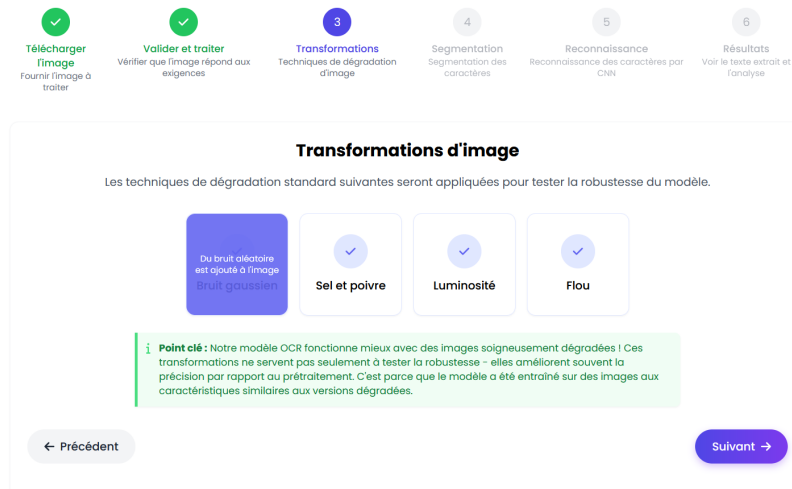


FIGURE 36 – Transformations appliquées à l'image

L'étape des transformations (Figure 36) montre les techniques de dégradation standard appliquées :

- Bruit gaussien : Du bruit aléatoire est ajouté à l'image
- Sel et poivre : Ajout de pixels blancs et noirs aléatoires
- Luminosité : Ajustement de la luminosité pour simuler différentes conditions
- Flou : Application d'un flou gaussien

Un point clé est mis en évidence : "Notre modèle OCR fonctionne mieux avec des images soigneusement dégradées ! Ces transformations ne servent pas seulement à tester la robustesse - elles améliorent souvent la précision par rapport au prétraitement."

7.6.4 Étape 4 : Segmentation des caractères

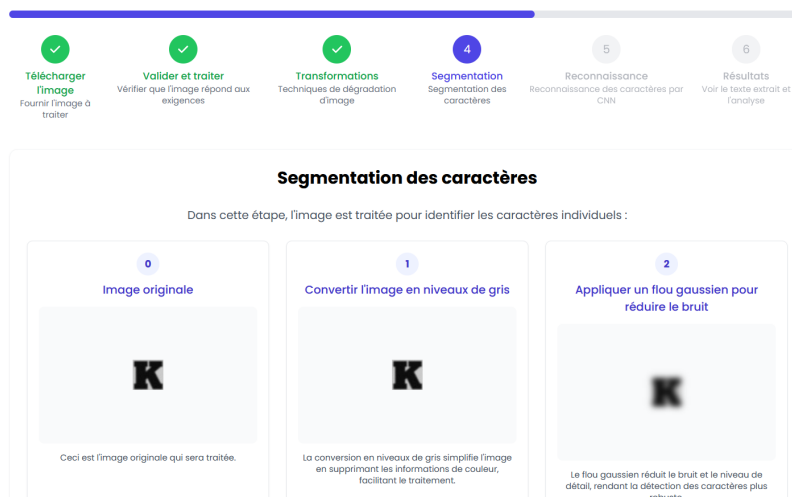


FIGURE 37 – Processus de segmentation des caractères (partie 1)

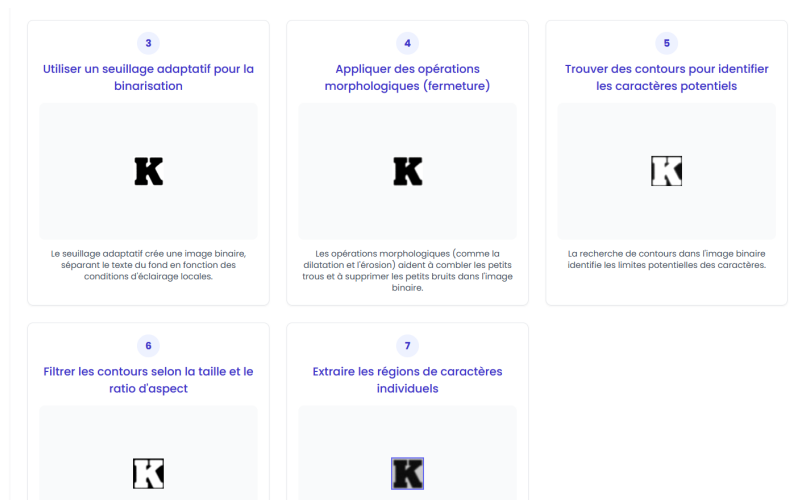


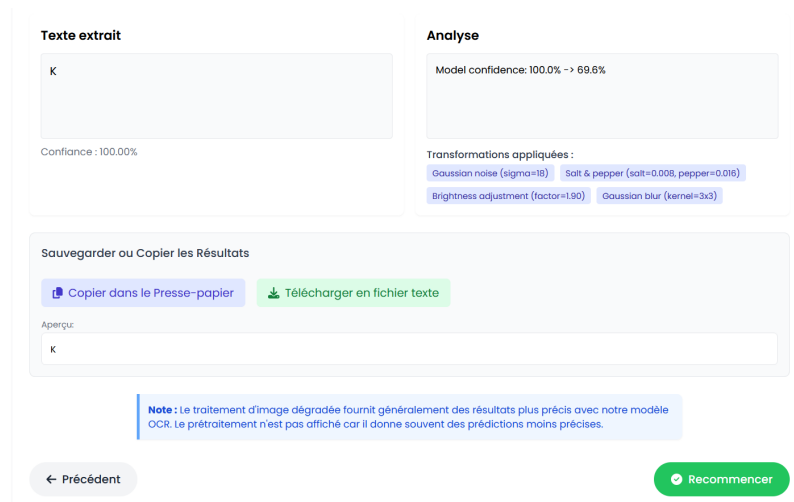
FIGURE 38 – Processus de segmentation des caractères (partie 2)

L'étape de segmentation (Figures 37 et 38) présente le processus d'identification des caractères à travers plusieurs sous-étapes :

1. Image originale
2. Conversion en niveaux de gris
3. Application d'un flou gaussien pour réduire le bruit
4. Utilisation d'un seuillage adaptatif pour la binarisation
5. Application d'opérations morphologiques (fermeture)
6. Recherche de contours pour identifier les caractères potentiels
7. Filtrage des contours selon la taille et le ratio d'aspect
8. Extraction des régions de caractères individuels

Chaque étape est illustrée par une image montrant la transformation progressive du caractère "K", accompagnée d'une explication claire de l'objectif de chaque traitement.

7.6.5 Étape 5 : Résultats de la reconnaissance



The screenshot displays the final results of the OCR process. It is divided into several sections:

- Texte extrait:** A box containing the extracted text "K". Below it, the confidence level is shown as "Confiance : 100.00%".
- Analyse:** A section showing the model's confidence change from 100.0% to 69.6%. It also lists the transformations applied: Gaussian noise (sigma=18), Salt & pepper (salt=0.008, pepper=0.016), Brightness adjustment (factor=1.90), and Gaussian blur (kernel=3x3).
- Sauvegarder ou Copier les Résultats:** Two buttons are present: "Copier dans le Presse-papier" (blue) and "Télécharger en fichier texte" (green).
- Aperçu:** A small preview box showing the text "K".
- Note:** A blue box with text explaining that degraded image processing generally provides more precise results with the model's OCR, but the preprocessing step is not shown as it often yields less precise predictions.
- Navigation:** At the bottom, there are two buttons: "← Précédent" (Previous) and "Recommencer" (Restart).

FIGURE 39 – Résultats de la reconnaissance OCR

L'étape finale des résultats (Figure 39) affiche :

- Le texte extrait ("K" dans cet exemple simple)
- Le niveau de confiance (100%)
- Une section d'analyse avec :
 - Confiance du modèle
 - Liste des transformations appliquées avec leurs paramètres précis :
 - Gaussian noise (sigma=18)
 - Salt & pepper (salt=0.008, pepper=0.016)
 - Brightness adjustment (factor=1.90)
 - Gaussian blur (kernel=3x3)
- Options pour sauvegarder ou copier les résultats :
 - Copier dans le Presse-papier
 - Télécharger en fichier texte
- Un aperçu du texte extrait
- Une note explicative indiquant que le traitement d'image dégradée fournit généralement des résultats plus précis
- Boutons de navigation pour revenir en arrière ou recommencer le processus

7.7 Support multilingue

L'application offre un support complet pour l'anglais et le français, accessible via un sélecteur de langue dans l'en-tête (EN/FR). Toutes les interfaces, instructions, et messages sont traduits dans les deux langues, permettant une expérience utilisateur entièrement localisée.

7.8 Conclusion

L'interface utilisateur d'OCR Vision est conçue avec une attention particulière à l'expérience utilisateur, combinant :

- Un design moderne et attrayant avec une palette de couleurs cohérente
- Une navigation intuitive et des instructions claires à chaque étape
- Une approche pédagogique permettant de comprendre le processus OCR
- Des options flexibles adaptées aux différents besoins des utilisateurs
- Des visualisations détaillées rendant le processus transparent
- Des fonctionnalités pratiques d'export des résultats

Cette interface équilibre efficacement simplicité d'utilisation et richesse fonctionnelle, rendant les techniques avancées de vision par ordinateur accessibles même aux utilisateurs non techniques.

8 Conclusion générale

Au terme de ce projet OCR Vision, nous avons développé une solution complète de reconnaissance optique de caractères combinant des techniques avancées de traitement d'image, de segmentation et de reconnaissance par réseaux de neurones. Ce travail nous a permis non seulement d'atteindre nos objectifs techniques, mais également d'acquérir une compréhension approfondie des défis et des solutions dans le domaine de la vision par ordinateur appliquée à l'OCR.

L'implémentation d'un pipeline OCR complet nous a conduits à explorer et maîtriser diverses techniques de traitement d'image, depuis la conversion en niveaux de gris et le seuillage adaptatif jusqu'aux opérations morphologiques et à la détection de contours. Ces manipulations, appliquées dans un contexte concret, ont transformé des concepts théoriques en compétences pratiques directement applicables. La segmentation précise des caractères, en particulier, s'est révélée être un défi stimulant qui a renforcé notre compréhension des principes fondamentaux de la vision par ordinateur. Une découverte particulièrement intéressante a été l'observation paradoxale que les images intentionnellement dégradées produisaient souvent de meilleurs résultats de reconnaissance que les images originales. Cette constatation contre-intuitive nous a amenés à repenser notre approche du prétraitement d'images et à approfondir notre compréhension de l'alignement nécessaire entre les caractéristiques des données d'entraînement et celles des images à analyser. Ce phénomène souligne l'importance d'une compréhension nuancée des modèles d'apprentissage automatique et de leurs spécificités.

Sur le plan du développement web, la création d'une interface bilingue intuitive avec React et Tailwind CSS nous a permis d'acquérir des compétences précieuses en matière de conception d'interfaces modernes et responsives. L'approche à deux voies que nous avons adoptée – offrant à la fois une reconnaissance rapide et un processus détaillé pas à pas – répond efficacement aux besoins variés des utilisateurs tout en servant de support pédagogique pour comprendre le fonctionnement de l'OCR.

Ce projet ouvre de nombreuses perspectives d'amélioration et d'extension. L'intégration de fonctionnalités de reconnaissance de documents plus complexes, le support de langues supplémentaires, ou encore l'implémentation de techniques adaptatives capables d'ajuster automatiquement les paramètres de prétraitement en fonction des caractéristiques spécifiques de chaque image, constituent des axes de développement prometteurs. Un déploiement cloud de l'application permettrait également d'élargir son accessibilité et faciliterait son intégration avec d'autres services.

En définitive, ce projet OCR Vision représente bien plus qu'une simple application technique – il incarne une approche pédagogique de la vision par ordinateur, rendant visibles et compréhensibles des processus habituellement cachés. En équilibrant rigueur technique et accessibilité, nous avons créé un outil qui démontre le potentiel des technologies modernes de traitement d'image et d'apprentissage automatique pour résoudre des problèmes concrets de reconnaissance de texte, tout en servant de plateforme éducative pour comprendre les principes sous-jacents de ces technologies.

9 Bibliographie

Références

- [1] Ronacher, A. (2023). *Flask : Web Development, One Drop at a Time*. Version 2.3.3. <https://flask.palletsprojects.com/>
- [2] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al. (2023). *TensorFlow : Large-Scale Machine Learning on Heterogeneous Systems*. Version 2.15.0. <https://www.tensorflow.org/>
- [3] Kay, R., Smith, R. (2023). *Tesseract OCR*. Version 5.3.1. <https://github.com/tesseract-ocr/tesseract>
- [4] Bradski, G. (2023). *OpenCV : Open Source Computer Vision Library*. Version 4.8.1. <https://opencv.org/>
- [5] Smith, R. (2007). *An Overview of the Tesseract OCR Engine*. In Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Vol. 2, pp. 629-633.
- [6] Zhang, Y., Gueguen, L., Zharkov, I., Zhang, P., Seifert, K., Kadlec, B. (2020). *From Pixels to Words : Document-level OCR with Visual Language Modeling and Efficient Transformer*. arXiv preprint arXiv :2005.13850.
- [7] Mahdipour Saray, H., Yavuz, Z. (2020). *Offline handwritten character recognition using deep learning methods*. International Journal of Computational Intelligence Systems, 13(1), 501-517.
- [8] Young, J., Cole, R. (2019). *Preprocessing techniques for improved OCR accuracy on degraded documents*. Pattern Recognition Letters, 124, 89-95.
- [9] Sauvola, J., Pietikäinen, M. (2000). *Adaptive document image binarization*. Pattern Recognition, 33(2), 225-236.
- [10] Zuiderveld, K. (1994). *Contrast Limited Adaptive Histogram Equalization*. Graphics Gems IV, 474-485.
- [11] Serra, J. (1983). *Image Analysis and Mathematical Morphology*. Academic Press, London.
- [12] Tan, M., Le, Q. (2019). *EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks*. Proceedings of the 36th International Conference on Machine Learning, PMLR 97 :6105-6114.
- [13] Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H. (2017). *MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv preprint arXiv :1704.04861.
- [14] Tong, W., Evans, D. A. (1996). *A statistical approach to automatic OCR error correction in context*. Proceedings of the fourth workshop on very large corpora, 88-100.
- [15] Meta. (2023). *React : A JavaScript library for building user interfaces*. Version 18.2. <https://react.dev/>
- [16] Wathan, A., Reinink, J. (2023). *Tailwind CSS : A utility-first CSS framework*. Version 3.3. <https://tailwindcss.com/>