Université Hassan Premier
Faculté des Sciences et Techniques de Settat
Département d'Informatique

Rapport Technique

Reconnaissance Optique et Intelligente de Caractères

OCR avec Tesseract & ICR avec Réseaux de Neurones Convolutifs

Réalisé par :
Aoulichak Mohamed
Oussama Ait Mendil
Mohammed Abdelkhalek
Fouad Ayyad

Encadré par :
Hicham Ben Alla

Code Source GitHub

Année Universitaire : 2024 - 2025

Décembre 2025

Table des matières

Chapitre 1

Introduction et Contexte

1.1 Problématique Générale

La reconnaissance automatique de texte dans les images représente un défi majeur en vision par ordinateur. Ce projet aborde deux paradigmes complémentaires :

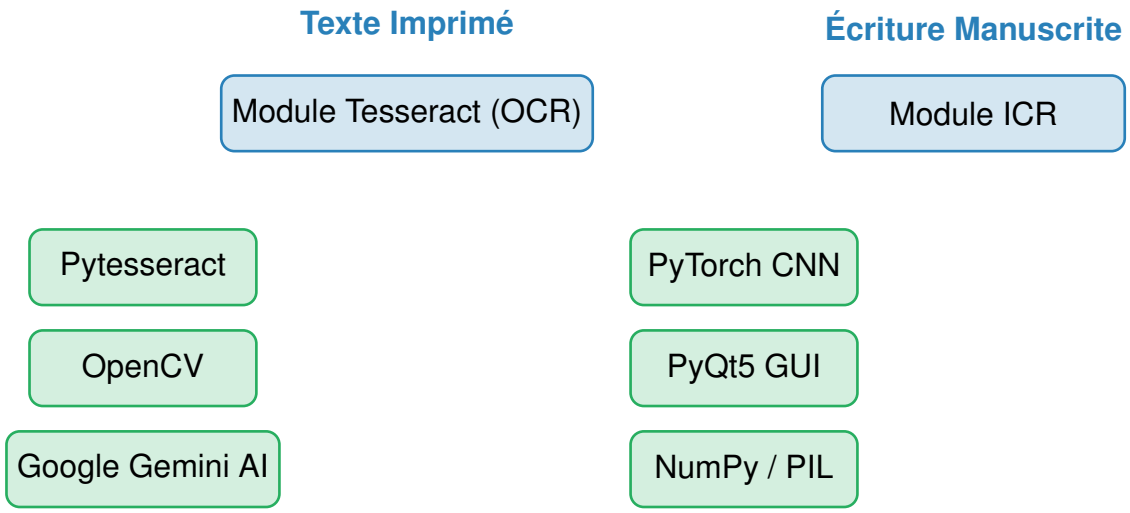
- **OCR (Optical Character Recognition)** : Reconnaissance de texte imprimé via Tesseract
- **ICR (Intelligent Character Recognition)** : Reconnaissance de caractères manuscrits via CNN

Définition

OCR vs ICR : L'OCR se concentre sur le texte standardisé (typographie régulière), tandis que l'ICR doit gérer la variabilité inhérente à l'écriture humaine (styles, inclinaisons, épaisseurs de trait variables).

1.2 Architecture du Projet

Le projet est structuré en deux modules indépendants mais complémentaires :



1.3 Objectifs du Rapport

Ce rapport technique vise à :

1. Documenter l'architecture logicielle des deux modules
2. Expliquer les fondamentaux théoriques de l'IA utilisée
3. Analyser le code source et les choix d'implémentation
4. Illustrer les pipelines de traitement via des diagrammes

Chapitre 2

Fondamentaux de l'IA et Bibliothèques

2.1 Vision par Ordinateur

Définition

La **vision par ordinateur** est un domaine de l'intelligence artificielle qui permet aux machines d'interpréter et de comprendre le contenu visuel d'images et de vidéos numériques.

2.1.1 Prétraitement d'Images

Le prétraitement est crucial pour améliorer la qualité de la reconnaissance :

Technique	Description
Conversion Grayscale	Réduction de 3 canaux RGB à 1 canal d'intensité
Débruitage (Median Blur)	Suppression du bruit impulsionnel
Seuillage Otsu	Binarisation adaptative pour séparer texte/-fond
Normalisation [0,1]	Mise à l'échelle des pixels pour stabilité numérique

TABLE 2.1 – Techniques de prétraitement d'images utilisées

2.1.2 Formulation Mathématique

Le seuillage d'Otsu minimise la variance intra-classe :

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{2.1}$$

où ω_0, ω_1 sont les probabilités des deux classes séparées par le seuil t.

2.2 Réseaux de Neurones Convolutifs (CNN)

Architecture CNN

Les CNN sont des réseaux de neurones spécialisés dans le traitement d'images. Ils utilisent des **convolutions** pour extraire des caractéristiques hiérarchiques (edges $\rightarrow$ textures $\rightarrow$ formes).

2.2.1 Composants Fondamentaux

Remarque

L'architecture utilisée dans le module ICR contient **3 blocs convolutionnels** avec progression des filtres : 32 $\rightarrow$ 64 $\rightarrow$ 128. Cette augmentation progressive permet de capturer des caractéristiques de plus en plus abstraites.

2.2.2 Fonctions d'Activation et Régularisation

— **ReLU** : $f(x) = \max(0, x)$ — Résout le problème du gradient évanescent
— **BatchNorm** : Normalise les activations, accélère la convergence
— **Dropout (p=0.5)** : Désactive aléatoirement 50% des neurones
— **Softmax** : Convertit les logits en probabilités : $\sigma(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$

2.3 Choix Techniques

2.3.1 Pourquoi Tesseract pour l'OCR ?

Information

Tesseract est un moteur OCR open-source développé par HP puis Google :

- Support multilingue (français, arabe, anglais)
- Intégration facile via Pytesseract
- Possibilité de combiner avec l'IA générative (Gemini)

2.3.2 Pourquoi PyTorch pour l'ICR ?

Avantage	Détail
Graphe dynamique	Débogage facilité, flexibilité
Écosystème riche	TorchVision, modèles pré-entraînés
GPU natif	Accélération CUDA transparente
Production ready	Export ONNX, TorchScript

TABLE 2.2 – Justification du choix de PyTorch

2.4 Bibliothèques Utilisées

Bibliothèque	Module	Usage
`pytesseract`	Tesseract	Interface pour Tesseract-OCR
`opencv-python`	Tesseract	Traitement d'image
`google-generativeai`	Tesseract	Correction IA via Gemini
`PySide6`	Tesseract	Interface graphique Qt
`torch`	ICR	Framework deep learning
`torchvision`	ICR	Datasets et transformations
`PyQt5`	ICR	Interface graphique
`numpy`	Les deux	Calcul matriciel

TABLE 2.3 – Dépendances du projet par module

Chapitre 3

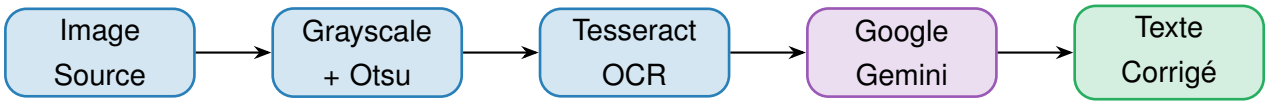
Analyse du Module Tesseract (OCR)

3.1 Vue d'Ensemble

Le module Tesseract combine l'OCR classique avec l'intelligence artificielle générative :

- Extraction de texte multilingue (FR + AR + EN)
- Amélioration automatique des images
- Correction intelligente via Google Gemini
- Détection automatique du type de document

3.2 Pipeline de Traitement



3.3 Extraits de Code Clés

3.3.1 Optimisation de l'Image

Listing 3.1 – Prétraitement d'image pour l'OCR

```
def optimize_img(self, img_cv2):
    """Ameliore l'image pour l'OCR."""
    # Conversion en niveaux de gris
    gray = cv2.cvtColor(img_cv2, cv2.COLOR_BGR2GRAY)

    # Debruitage par filtre median
    denoised = cv2.medianBlur(gray, 3)
```

```
8
9       # Binarisation avec seuillage Otsu
10      _, thresholded = cv2.threshold(
11          denoised, 0, 255,
12          cv2.THRESH_BINARY | cv2.THRESH_OTSU
13      )
14      return thresholded
```

3.3.2 Extraction de Texte

Listing 3.2 – Extraction OCR multilingue

```
1  def extract_txt(self, processed_img) -> str:
2      """Extrait le texte via Tesseract."""
3      # PSM 6 = bloc de texte uniforme
4      # Langues: francais + arabe + anglais
5      custom_config = r'--psm 6 -l fra+ara+eng'
6
7      text = pytesseract.image_to_string(
8          processed_img,
9          config=custom_config
10     )
11     return text.strip()
```

3.3.3 Intégration Google Gemini

Listing 3.3 – Traitement IA avec Google Gemini

```
1  def process_text(self, raw_text: str) -> dict:
2      """Envoie le texte a Gemini pour correction."""
3      prompt = f"""Tu es un assistant expert OCR.
4
5      TEXTE BRUT: "{raw_text}"
6
7      INSTRUCTIONS:
8      1. Corrige les erreurs OCR
9      2. Detecte le type de document
10
11     FORMAT:
12     <TYPE_DOCUMENT>...</TYPE_DOCUMENT>
13     <TEXTE_CORRIGE>...</TEXTE_CORRIGE>
```

```
14      """
15
16      response = self.model.generate_content(prompt)
17      return self._parse_response(response.text)
```

Chapitre 4

Analyse du Module ICR

4.1 Vue d'Ensemble

Le module ICR utilise un CNN pour reconnaître les lettres manuscrites (A-Z) :

— Dessin de caractères à la souris
— Chargement d'images externes
— Prédiction en temps réel avec Top-5
— Visualisation de la confiance

4.2 Architecture du Modèle CNN

Couche	Type	Output	Params
Input	—	(1, 28, 28)	0
Conv Block 1	Conv+BN+ReLU+Pool	(32, 14, 14)	320
Conv Block 2	Conv+BN+ReLU+Pool	(64, 7, 7)	18,496
Conv Block 3	Conv+BN+ReLU+Pool	(128, 3, 3)	73,856
Flatten	—	(1152,)	0
FC1	Linear+ReLU+Dropout	(256,)	295,168
FC2	Linear	(26,)	6,682
Total			**394,522**

TABLE 4.1 – Architecture détaillée du CNN

4.3 Implémentation PyTorch

Listing 4.1 – Architecture CNN pour ICR

```python
class AdvancedCNN(nn.Module):
    """CNN pour reconnaissance de caracteres."""

    def __init__(self, num_classes=26):
        super(AdvancedCNN, self).__init__()

        # Bloc Convolutionnel 1
        self.conv_block1 = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),
            nn.BatchNorm2d(32),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )

        # Bloc Convolutionnel 2
        self.conv_block2 = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )

        # Bloc Convolutionnel 3
        self.conv_block3 = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )

        # Classificateur
        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(128 * 3 * 3, 256),
            nn.ReLU(inplace=True),
            nn.Dropout(p=0.5),
            nn.Linear(256, num_classes)
        )
```

```
39
40      def forward(self, x):
41          x = self.conv_block1(x)
42          x = self.conv_block2(x)
43          x = self.conv_block3(x)
44          return self.classifier(x)
```

4.4 Datasets et Entraînement

Information

Le modèle est entraîné sur la **fusion de deux datasets** :

- **A-Z Handwritten Alphabets** (Kaggle) : 370,000 échantillons
- **EMNIST Letters** (TorchVision) : 145,000 échantillons

Total : ∼**515,000 images** au format 28×28 pixels.

4.4.1 Hyperparamètres

Paramètre	Valeur	Justification
Optimiseur	Adam	Convergence rapide
Learning Rate	0.001	Standard pour Adam
Batch Size	64	Compromis vitesse/généralisation
Dropout	0.5	Régularisation forte
Loss	CrossEntropy	Classification multi-classe

TABLE 4.2 – Hyperparamètres d'entraînement

4.5 Prédiction

Listing 4.2 – Processus de prédiction

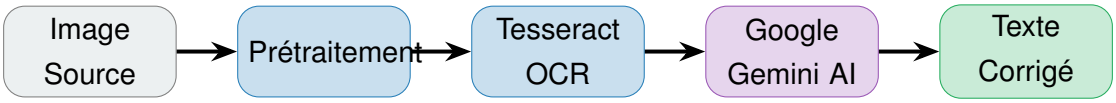
```
1 def predict_drawing(self):
2     """Prediction sur le dessin."""
3     # Image preprocessee (28x28, normalisee)
4     img_array = self.canvas.get_image()
5
6     # Conversion en tenseur PyTorch
```

```
    img_tensor = torch.FloatTensor(img_array)\
        .unsqueeze(0).unsqueeze(0).to(self.device)

    # Prediction
    self.model.eval()
    with torch.no_grad():
        outputs = self.model(img_tensor)
        probs = F.softmax(outputs, dim=1)

        # Top-5 predictions
        top5_probs, top5_idx = torch.topk(probs, 5)
        predicted_letter = chr(65 + top5_idx[0][0])

    return predicted_letter
```

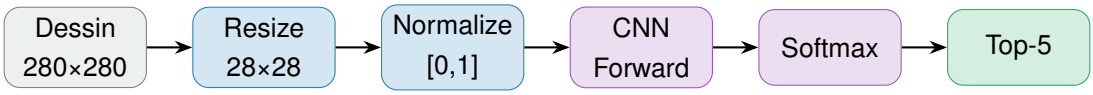
Chapitre 5

Visualisations et Diagrammes

5.1 Pipeline Complet OCR + IA



5.2 Pipeline ICR (Deep Learning)



5.3 Comparaison OCR vs ICR

Caractéristique	OCR (Tesseract)	ICR (CNN)
Type de texte	Imprimé	Manuscrit
Langues	FR, AR, EN	A-Z uniquement
Correction IA	Oui (Gemini)	Non
Temps réel	Non	Oui
Entraînement	Pré-entraîné	Custom

TABLE 5.1 – Comparaison des deux modules

Chapitre 6

Conclusion et Perspectives

6.1 Synthèse

Ce projet démontre une approche complète de la reconnaissance de caractères :

— **OCR Classique** : Tesseract avec prétraitement OpenCV
— **IA Générative** : Correction via Google Gemini
— **Deep Learning** : CNN PyTorch pour manuscrit
— **Interfaces Modernes** : GUI Qt professionnelles

6.2 Performances

Module	Métrique	Résultat
Tesseract + Gemini	Qualité correction	Élevée
ICR (CNN)	Accuracy (test)	$>$ 90%
ICR (CNN)	Top-5 Accuracy	$>$ 98%

TABLE 6.1 – Résumé des performances

6.3 Perspectives

Améliorations Futures

1. Ajouter les chiffres (0-9) et caractères spéciaux
2. Intégrer un LSTM pour séquences de caractères
3. Utiliser Transfer Learning (ResNet, EfficientNet)
4. Déploiement Web avec FastAPI + React
5. Application mobile (TensorFlow Lite)

Rapport rédigé dans le cadre du projet ICR

A. Mohamed — Décembre 2025
Université Hassan Premier - FSTS

Annexe A

Fichiers de Configuration

A.1 Requirements - Module Tesseract

Listing A.1 – requirements.txt - Tesseract

```
PySide6>=6.5.0
opencv-python>=4.8.0
numpy>=1.24.0
pytesseract>=0.3.10
google-generativeai>=0.8.0
python-dotenv>=1.0.0
```

A.2 Requirements - Module ICR

Listing A.2 – requirements.txt - ICR

```
PyQt5>=5.15.0
torch>=2.0.0
torchvision>=0.15.0
numpy>=1.21.0
Pillow>=9.0.0
```