Models and libraries in ocr for Arabic language in general and in Android :

**Comparison entre easyocr et tesseract ocr using some images:**

to evaluate OCR engines : metrics used are **Character Error Rate(CER)** and **Word Error Rate(WER)**

* **Good**OCR accuracy: CER **1‐2%**(i.e. 98–99% accurate)
* **Average**OCR accuracy: CER **2-10%**
* **Poor**OCR accuracy: CER **>10%**(i.e. below 90% accurate)

Comparaison avec quelques image en utulisant cer :

1)



dans le cas d’une image qui contient un peu de noise :

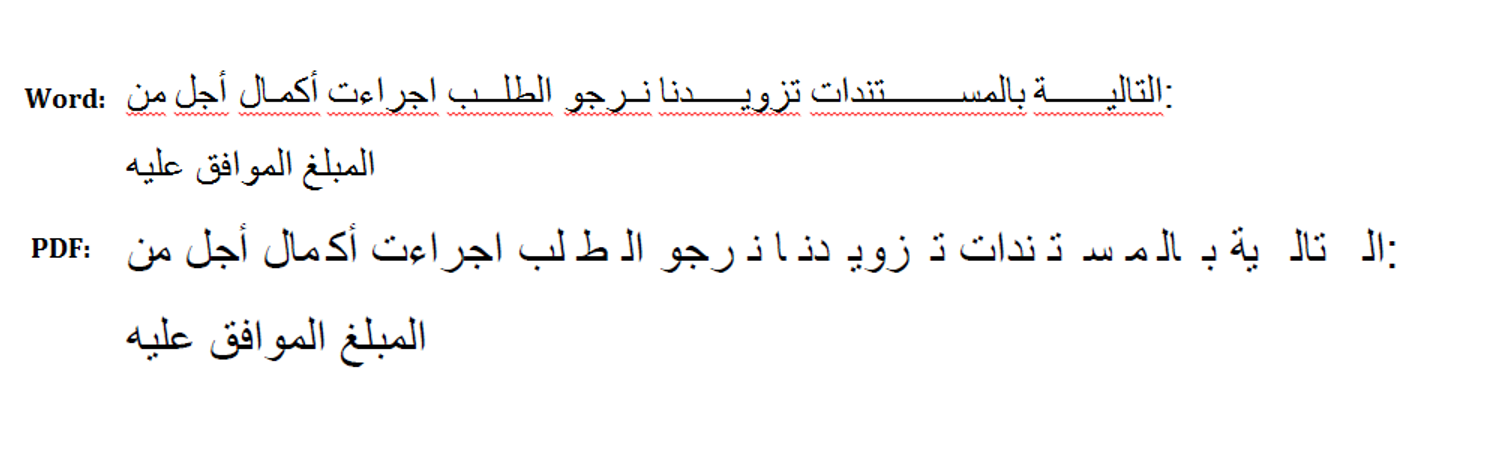
cer est 5% pour easy ocr mais 100% pour tesseract, Alors le texte n’est pas reconnu avec tesseract mais il est reconnu avec easyocr ,easyocr plus pratique dans les images avec un peu de noise .

2)



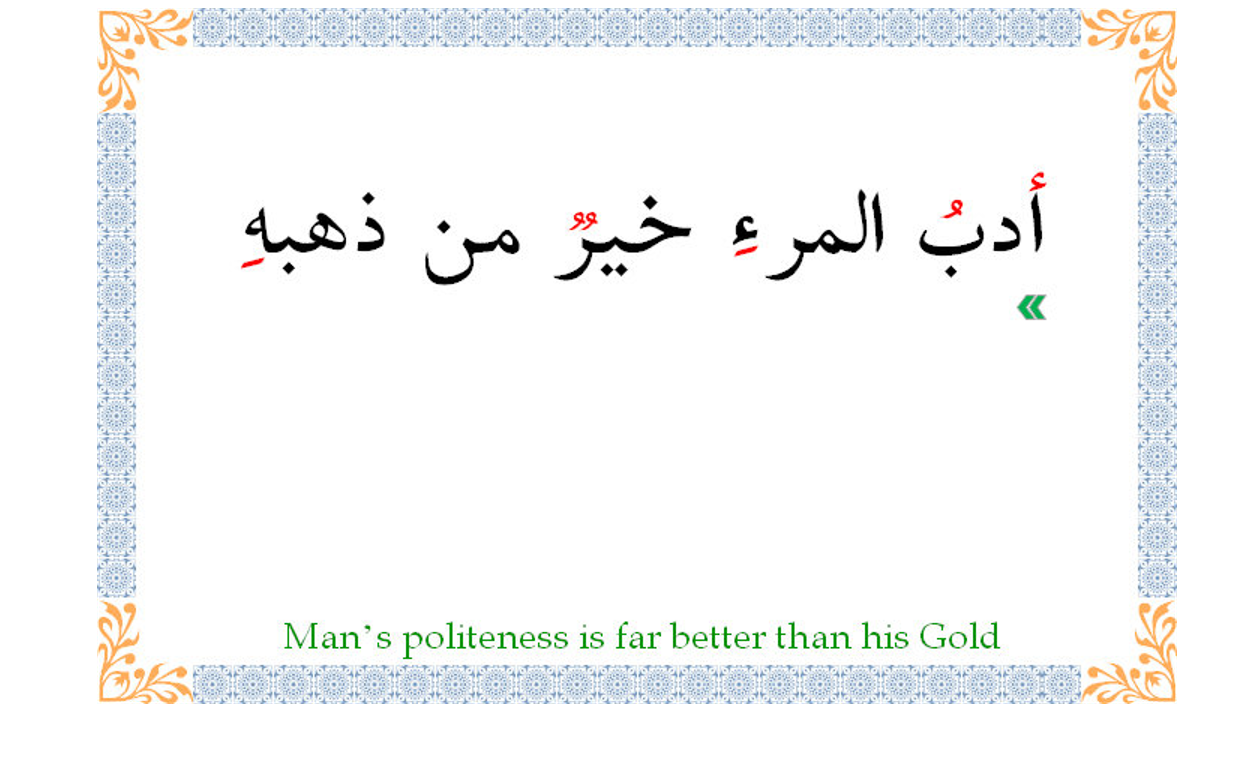
dans cette image : Cer : 0.75% pour tesseract et 3.92% pour easyocr .les deux donne une meilleur performance dans le cas d’un texte claire avec une background blanche.

3)



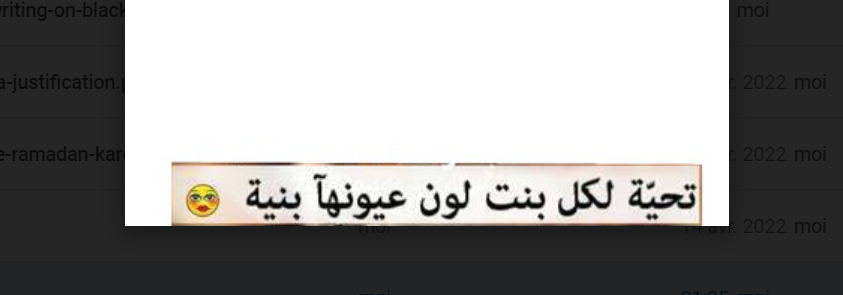
Dans cette image tesseract et plus performant que easy ocr ,ce dernier a un probleme dans les mots qui contient des longues lettres arabe comme dans l’image .donc tesseract est plus pratique dans les images avec une background blanche

4)



dans ces deux image : cer : 100% pour tesseract et 0% pour easyocr .alors easyocr apu bien detecter le texte arabe pas comme tesseract n’as pas arriver a le detecter car il y a un peu de noise

5)



dans cette image : cer : 15.09% pour tesseract et 9.40% pour easyocr .alors easyocr a pu bien detecter le texte arabe mieux que tesseract

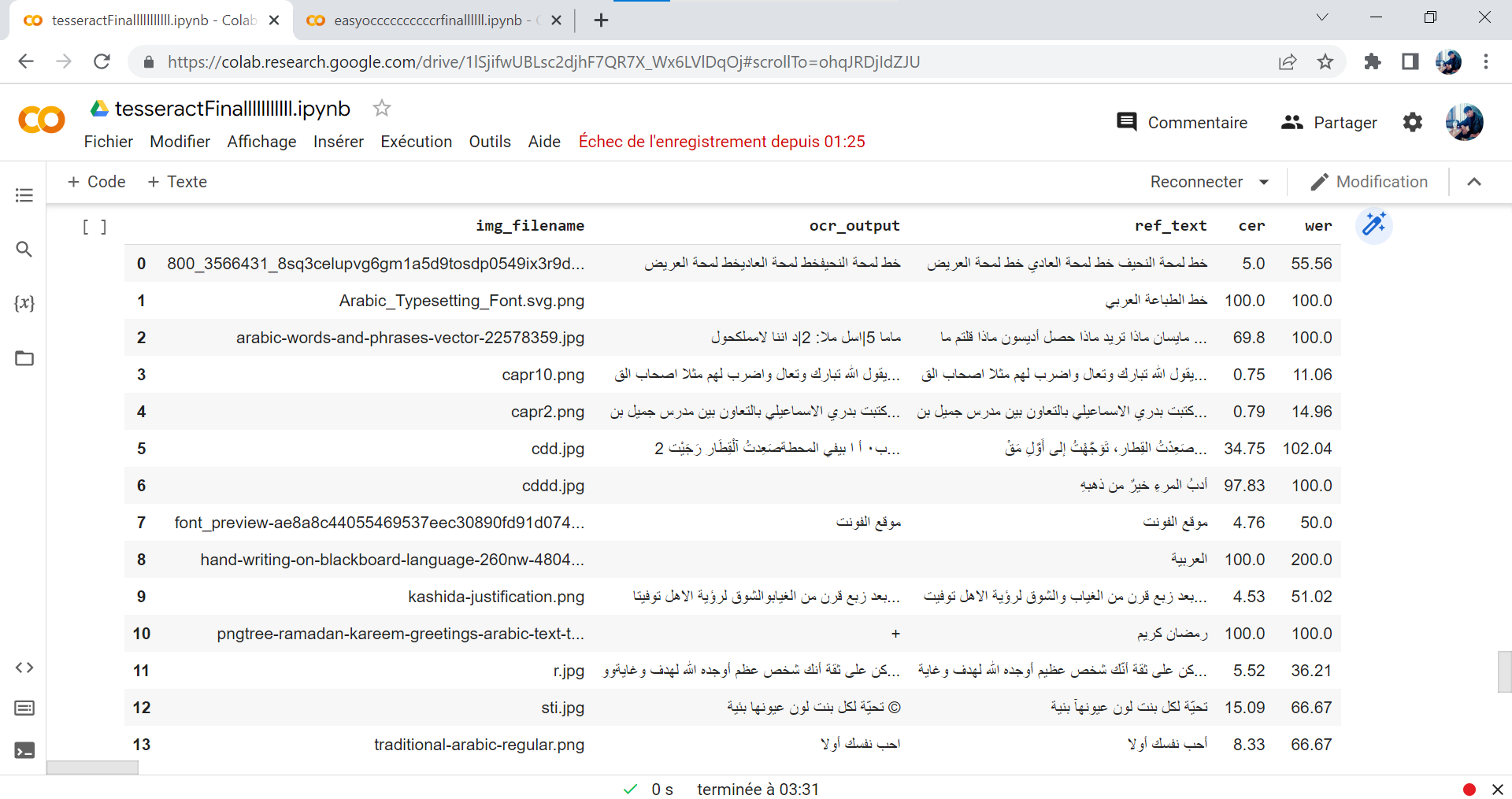
6)



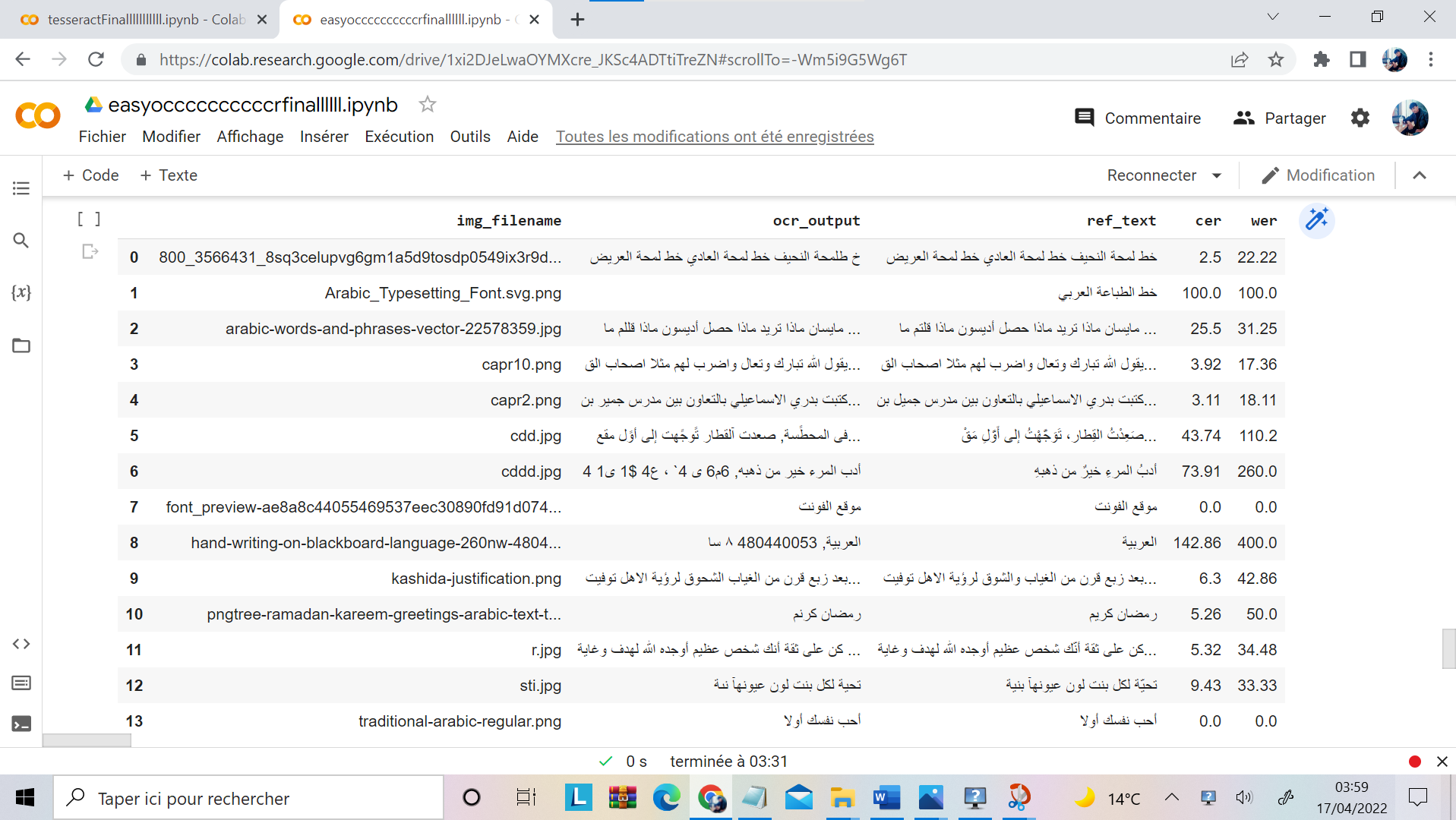
dans cette image : cer :0% pour tesseract et pour easyocr .les deux n ont pas arriver a detecter le texte .

MES RESULTATS POUR TESSERACT OCR ET EASYOCR AVEC WER ET CER metrics of evaluation :

Pour tesseract :

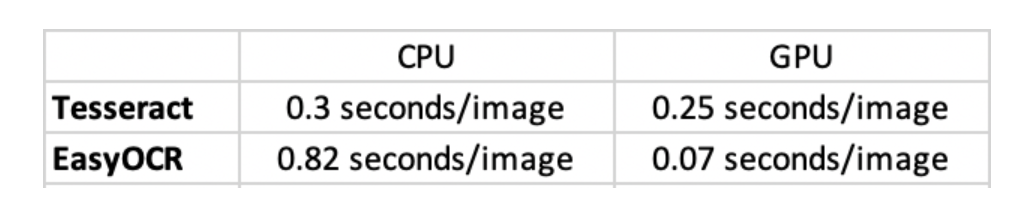


Pour easy ocr :



Vitesse des models:

CPU uses :Tesseract.

GPU uses :EasyOcr

**Advantages of Tesseract:**

* Tesseract work pretty faster with multiple images.
* Tesseract has the configuration to extract only the digits.
* Works better on clean images with wight background
* better NER extraction and conversion time with better accuracy .
* have [tessdata\_best](https://github.com/tesseract-ocr/tessdata_best): Best trained models of tesseract OCR and acts as the base models for fine-tuning (new)
* **Advantages of EasyOCR:**
* EasyOCR supports the GPU and performance is good on GPU.
* EasyOCR provides the confidence of the extracted text
* EasyOCR works better with noisy images when compared with tesseract.
* Easy OCR is better on characters,

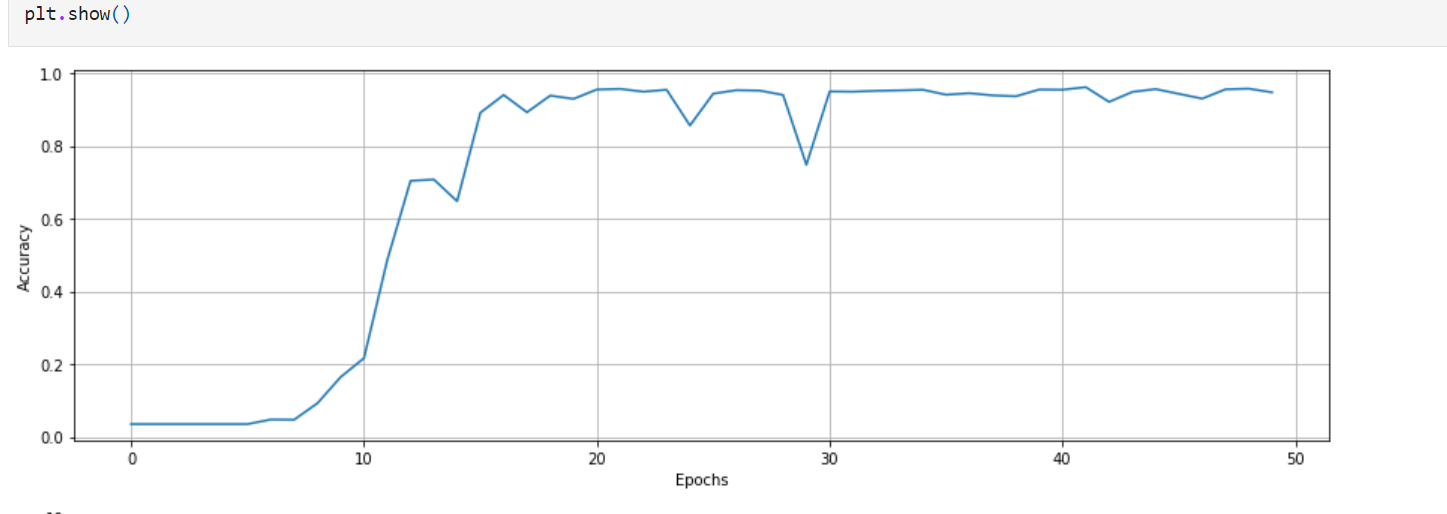
Limitations Tesseract and EasyOCR:

* If a document contains languages outside of those given in the LANG arguments, results may be poor.
* On handwritten text both would give low results.
* Doesn’t do well with images affected by artifacts including partial occlusion, distorted perspective, and complex background
* accuracy can be increased by filtering the noise images.

OTHER STUDIES FOR TESSERACT :

-Patel and friends produced 70% accuracy using 20 sample images in 2012 . Kumar and friends produced 97% accuracy for small scanned bill documents and 83% accuracy for small scanned bill documents using Tesseract OCR on 25 scanned bills in 2020 . Akinbade and friends produced 81.9% character accuracy and 69.7% word accuracy on 11 sample images in 2020 .

model achieves test accuracy above 95% for [Arabic Handwritten Digits Dataset on Kaggle](https://www.kaggle.com/mloey1/ahdd1)



<https://github.com/elnomrosy66/ocr-arabic>

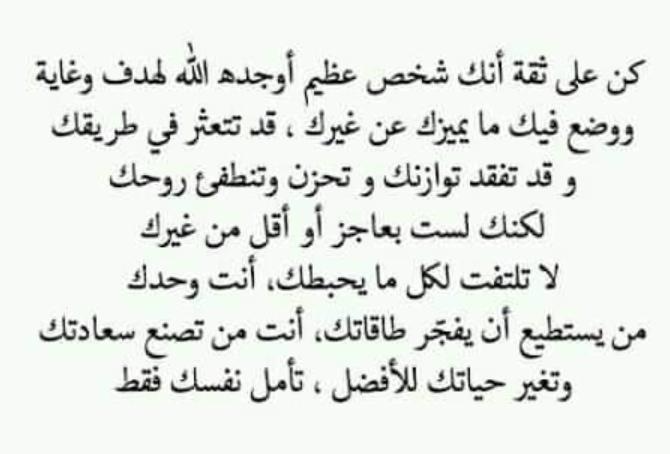
ArabicOcr LIBRARY  :

Library in python .

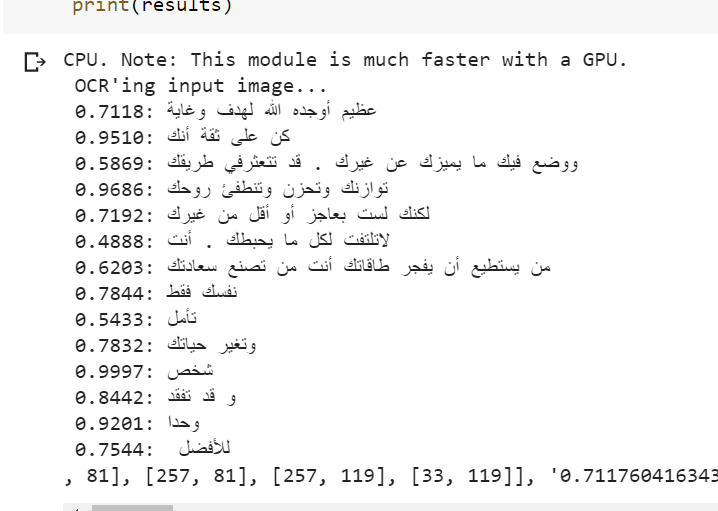
Pip install ArabicOcr

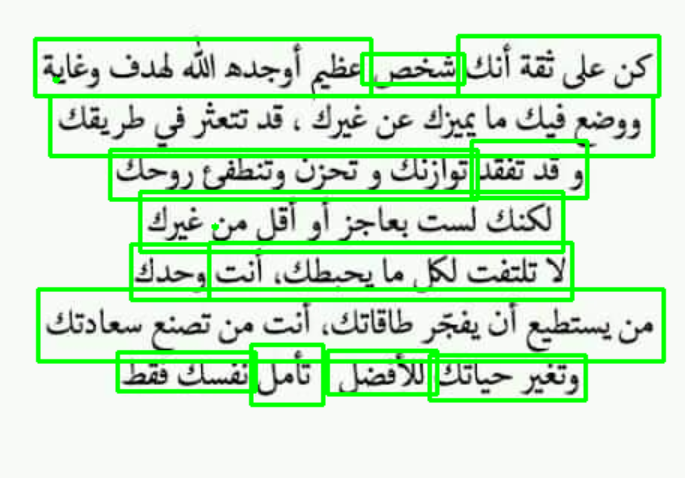
Donne des bonnes performances mais il detecte par mot et donne a chaque detection sa performance, probleme quand je fusionne les mots ils ont des positions differentes :

Image :



Resultat obtenu pour cette image:



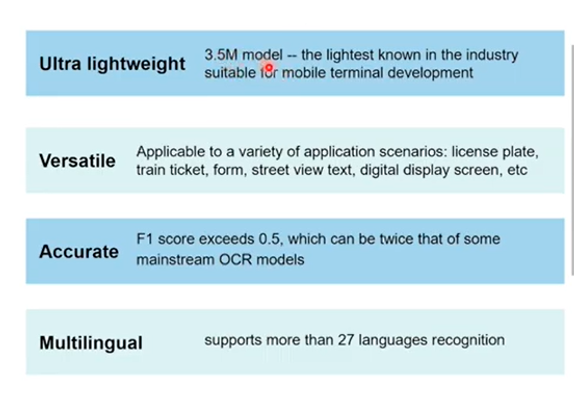


# PaddleOCR:

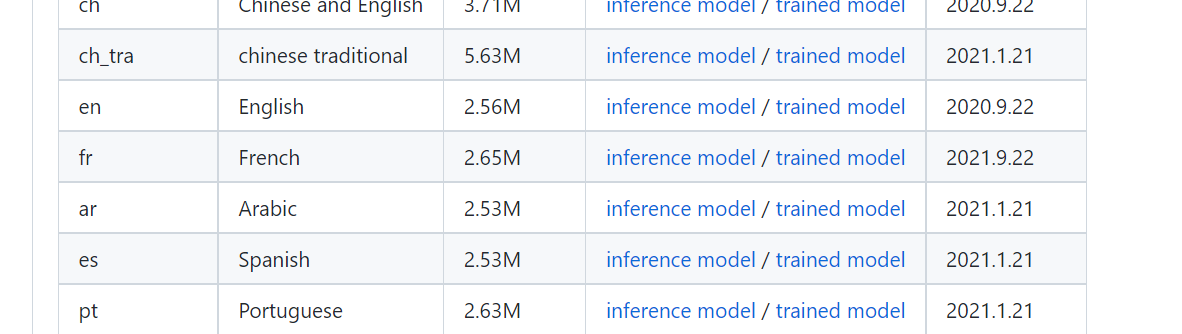
# The latest lightweight OCR system, provides an easy-to-use ultra lightweight OCR system.

PaddleOCR For mobile

<https://medium.com/axinc-ai/paddleocr-the-latest-lightweight-ocr-system-a13171d7ea3e>

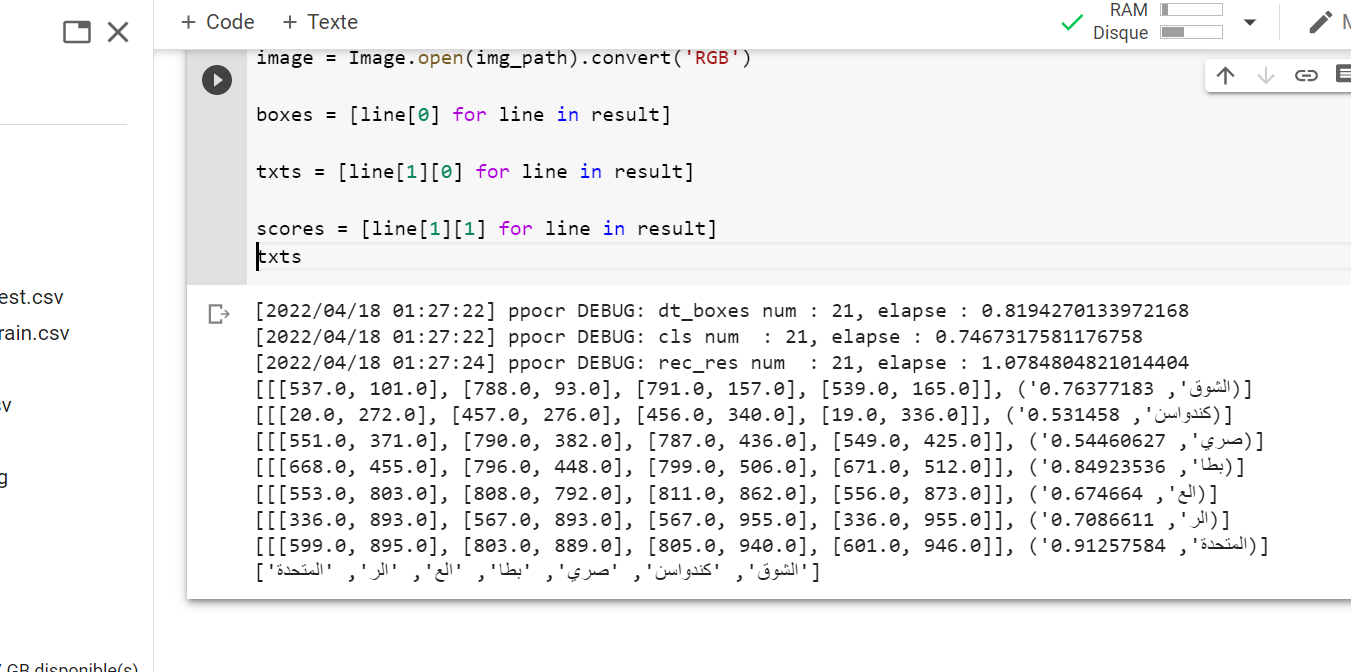
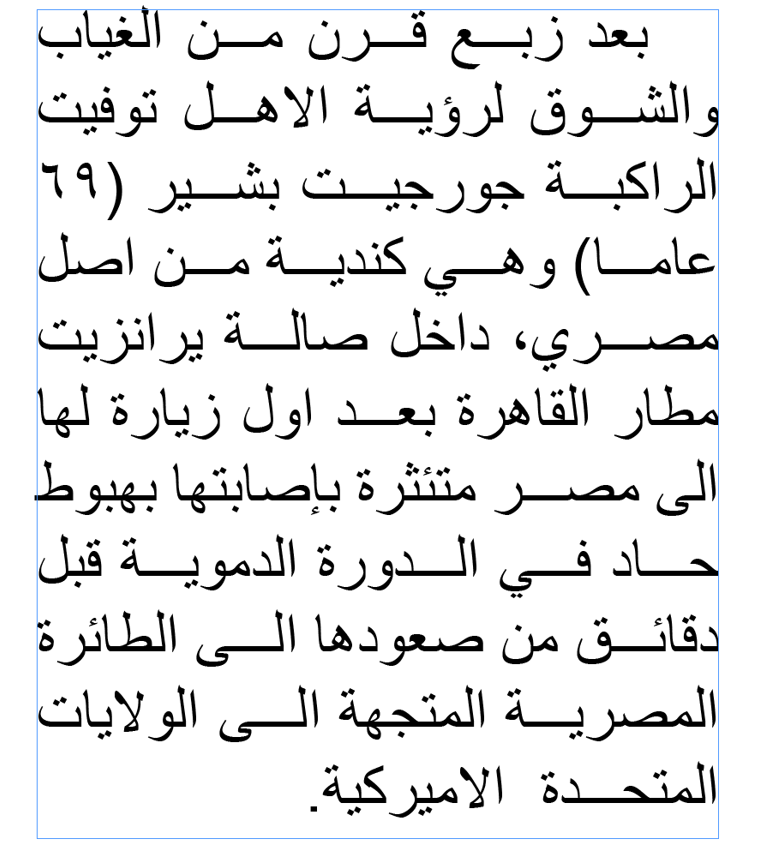


Paddle ocr model size :

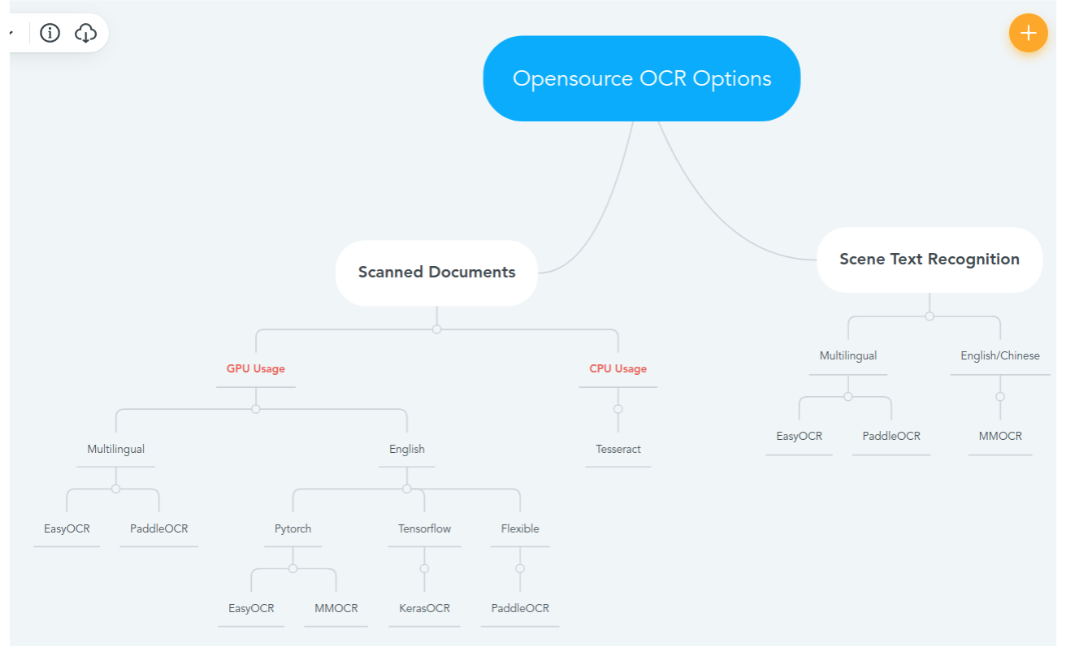


* + Ultra lightweight PP-OCRv2 series models: detection (3.1M) + direction classifier (1.4M) + recognition 8.5M) = 13.0M
  + Ultra lightweight PP-OCR mobile series models: detection (3.0M) + direction classifier (1.4M) + recognition (5.0M) = 9.4M
  + Support multi-lingual recognition: about 80 languages

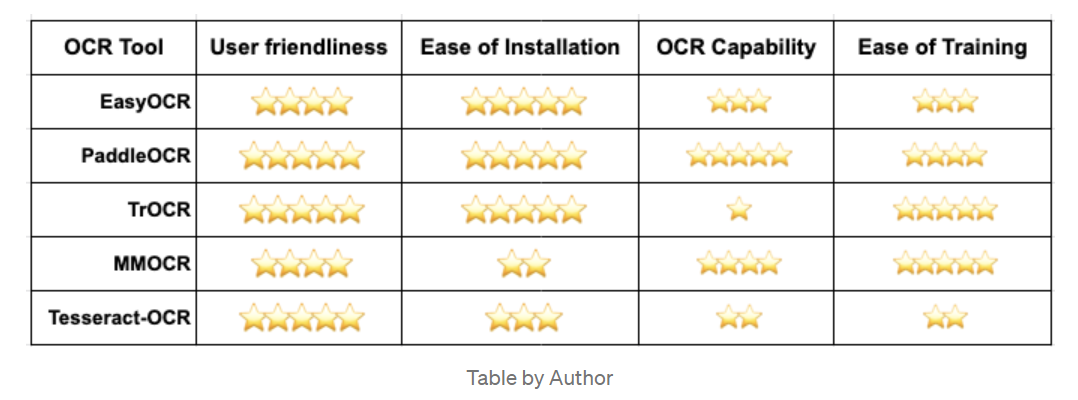
Il est performant pour chinese and english pour l’arabe comme elle est nouvelle elle demande plus d’amélioration exemple que j’ai tester :



Choose your OpenSource OCR :



comparaison entre quelque opensource ocr engines :



<https://towardsdatascience.com/5-open-source-tools-you-can-use-to-train-and-deploy-an-ocr-project-8f204dec862b>

ARABIC OCR TEXT DETECTION FOR MOBILE  :

# 1)Android OCR Application Based on Tesseract :

# build android app for OCR using tess two with android studio and tesseract

[Arabic OCR requires the installation of all Cube data files](https://github.com/tesseract-ocr/tesseract/wiki/Data-Files#cube-data-files-for-version-304305)

You can find the cube files for the arabic language dataset from github

Procedure :

1. Create android project

2. Add dependency into build.gradle “compile 'com.rmtheis:tess-two:6.3.0”

'tess-two:9.0.0'' - it contains a fork of Tesseract Tools for Android ([tesseract-android-tools](https://github.com/alanv/tesseract-android-tools)) that adds some additional functions. Also, we need camera and write permissions, so add it to AndroidManifest.xml.

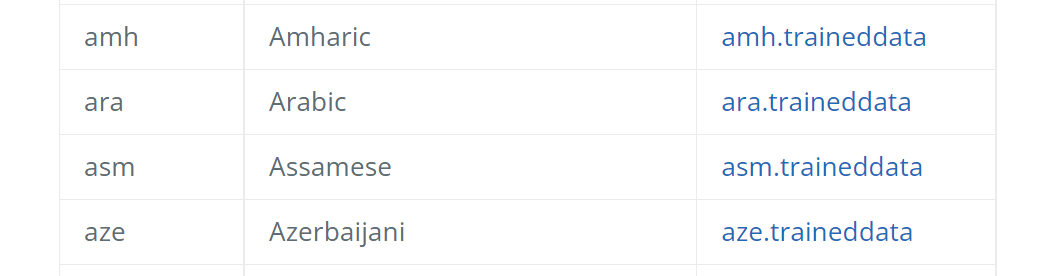
3. Download trained data - choose language: if vietnamese - vie.traineddata if japanese - jpn.traineddata ,ara.traineddata if arabic……

4. Import trained data to your project

5. start Coding

## Traineddata Files for Version 4.00 + :

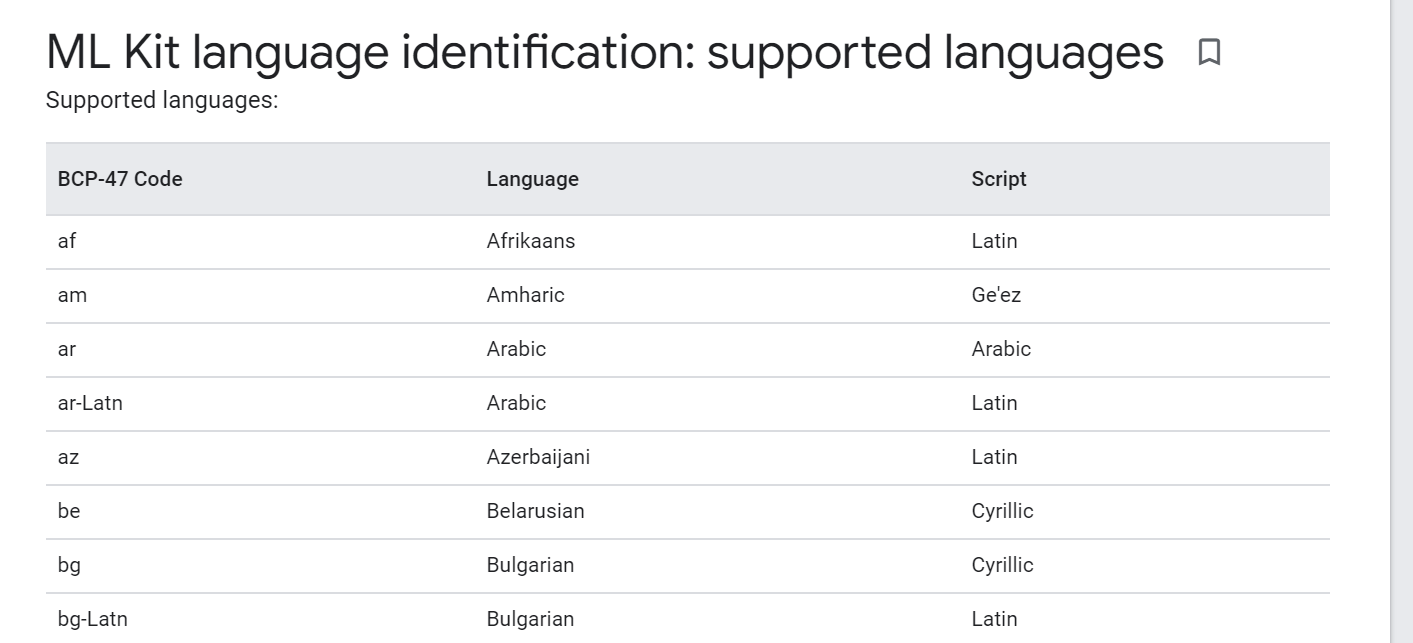
There is also arabic :

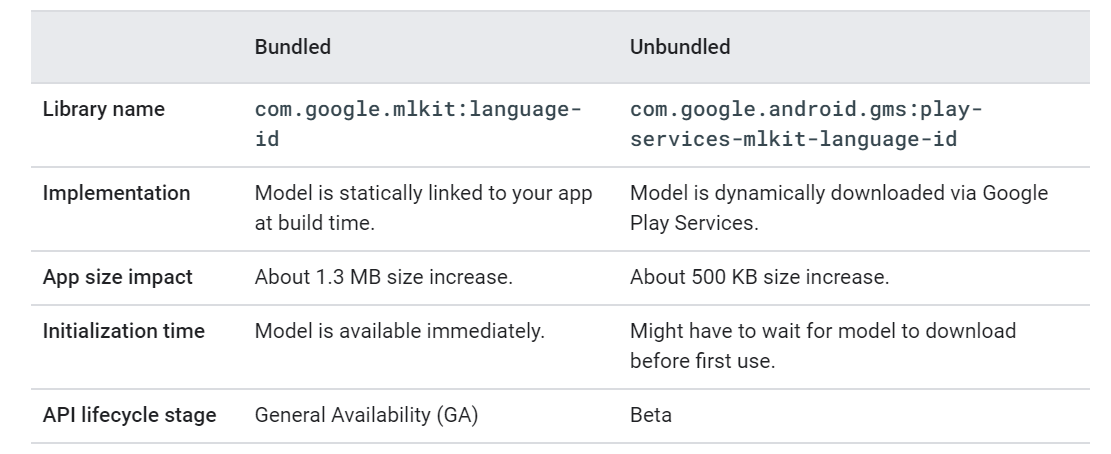


2)ML Kit language identification(Firebase):

ML Kit is a library from Google include computer vision-related tasks, such as text recognition API.

ML Kit recognizes text in more than 100 different languages in their native scripts. In addition, romanized text can be recognized for Arabic, Bulgarian, Chinese, Greek, Hindi, Japanese, and Russian.





3)convert pretrained model to tflite  :

Convert easy ocr (craft and crnn for detection and recognition) to tflite using tensorflow and use it on android studio

4) google vision API :

<https://cloud.google.com/vision/> This API supports many languages including Arabic.

5)Openalpr library :

Pour detecter le texte dans les plaques des voitures

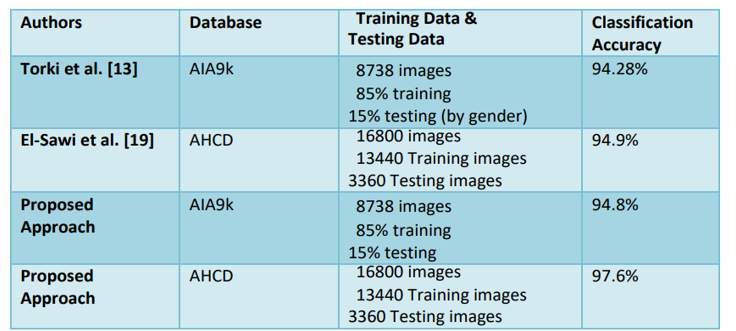
OpenALPR is an open source  library written in C++ with bindings in C#, Java, Node.js, Go, and Python. The library analyzes images and extract texts

<https://github.com/SandroMachado/openalpr-android>

AUTRES MODELS ET ETUDES POUR ARABIC OCR :

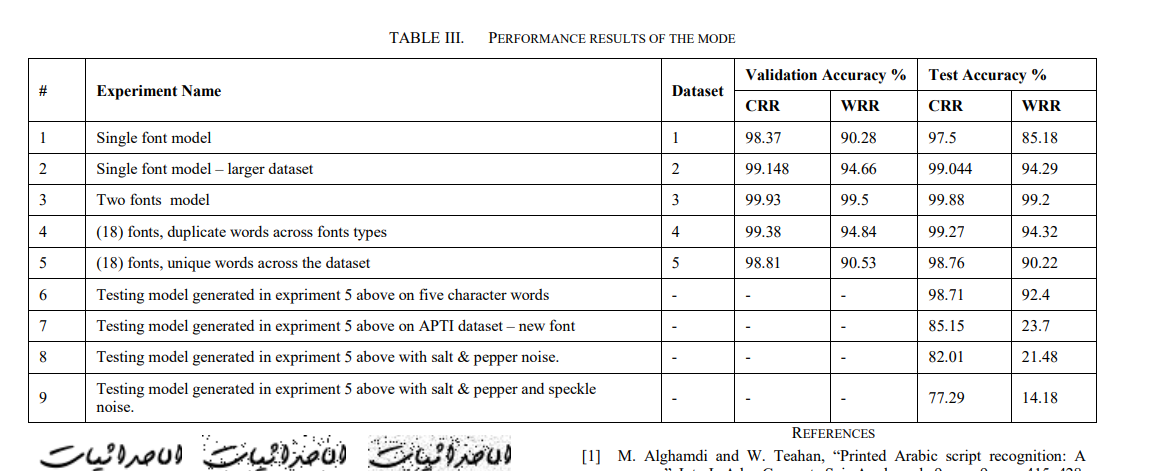
CNN :

capable of classifying Arabic handwritten characters with a stateof-the-art classification accuracy of 94.8% and 97.6% on the AIA9k and AHDC datasets :



Evaluating CNN model:

To evaluate the model, CRR and Words Recognition Rate (WRR) measures were employed :



The dataset used :

from kaggle kernels which include **[Arabic Digits](https://www.kaggle.com/mloey1/ahdd1)** and **[Arabic Letters](https://www.kaggle.com/mloey1/ahcd1)**.

**Phases and results :**

Data Preprocessing

Image Normalization

## Designing Model Architecture

As results we can see that best parameters are:

* Optimizer: Adam
* Kernel\_initializer: uniform
* Activation: relu

## training and fitting Model

Result :

**Test accuracy is improved from 98.286% to 98.862% As we train the model on 20 more epochs. so the CNN model is tested on more than 13000 image with all possible classes and got very high accuracy of 98.86%.**

CNN and RNN :

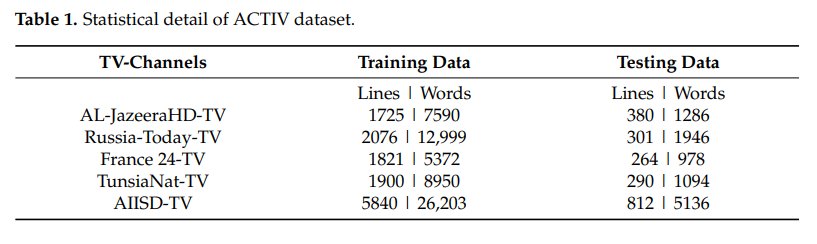
model takes an input image and generates feature sequences through a CNN. These sequences are transferred to a bidirectional RNN to obtain feature sequences in order.

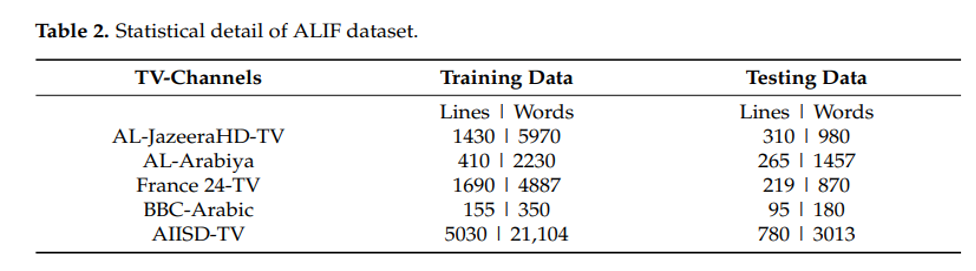
Datasets used :

ACTIV (21520 images)and ALIF(6532 text lines images)

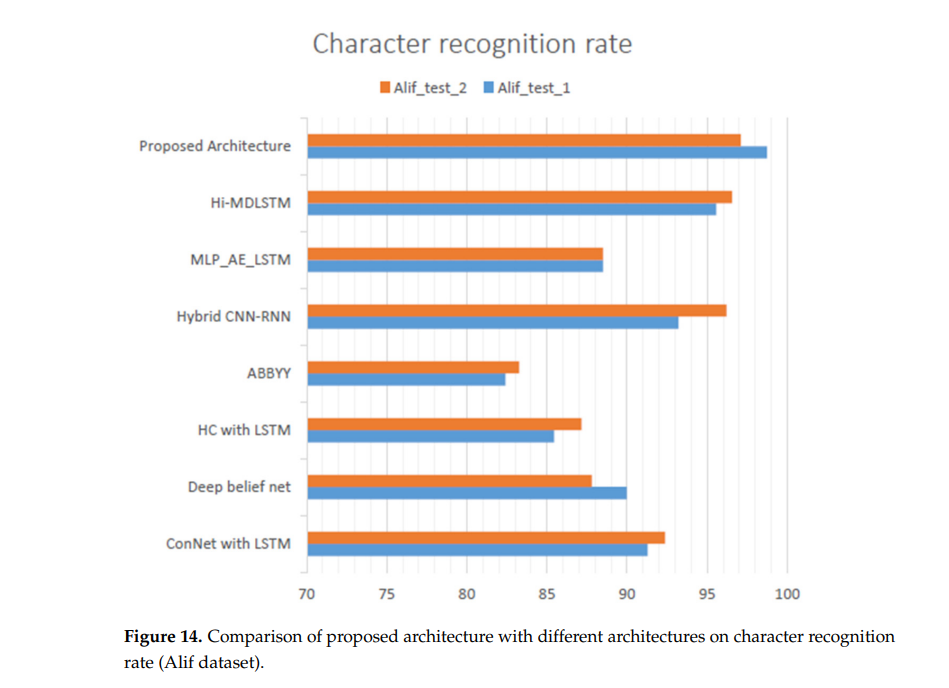
VGG architecture used ;

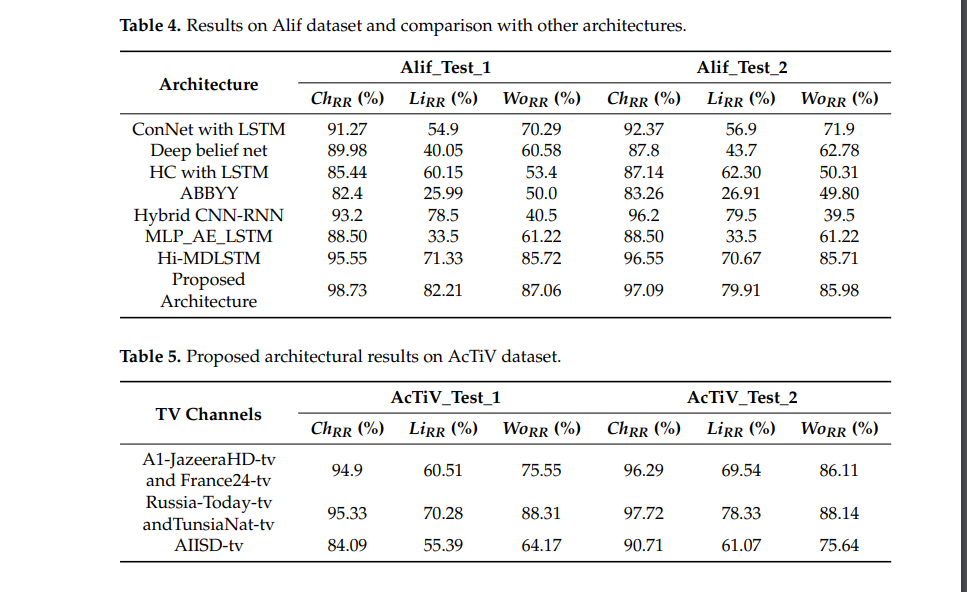
Article de 2021





Results and Performance of character recognation rate on ALIF dataset :





HMMS :

resistant to noise, tolerate variations in writing, and the HMM tools are freely available.

HMM models have obtained a correctness rate of 73.78 % and an accuracy rate of 71.74 % for window features using horizontal and vertical edge derivatives of the image.

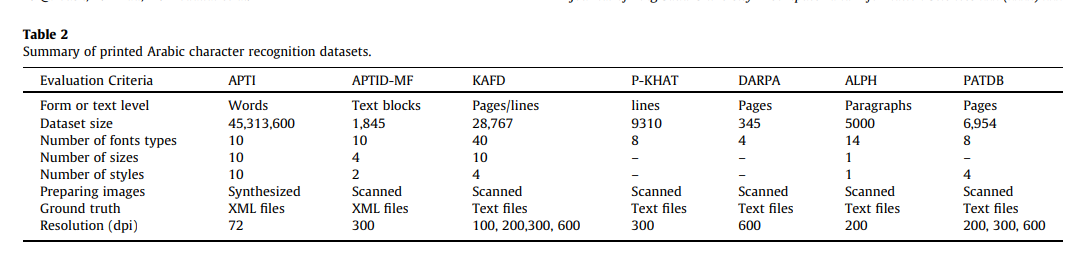
Using the hierarchal window features, the best results obtained are a correctness rate of 81.05 % and an accuracy rate of 78.04 %.

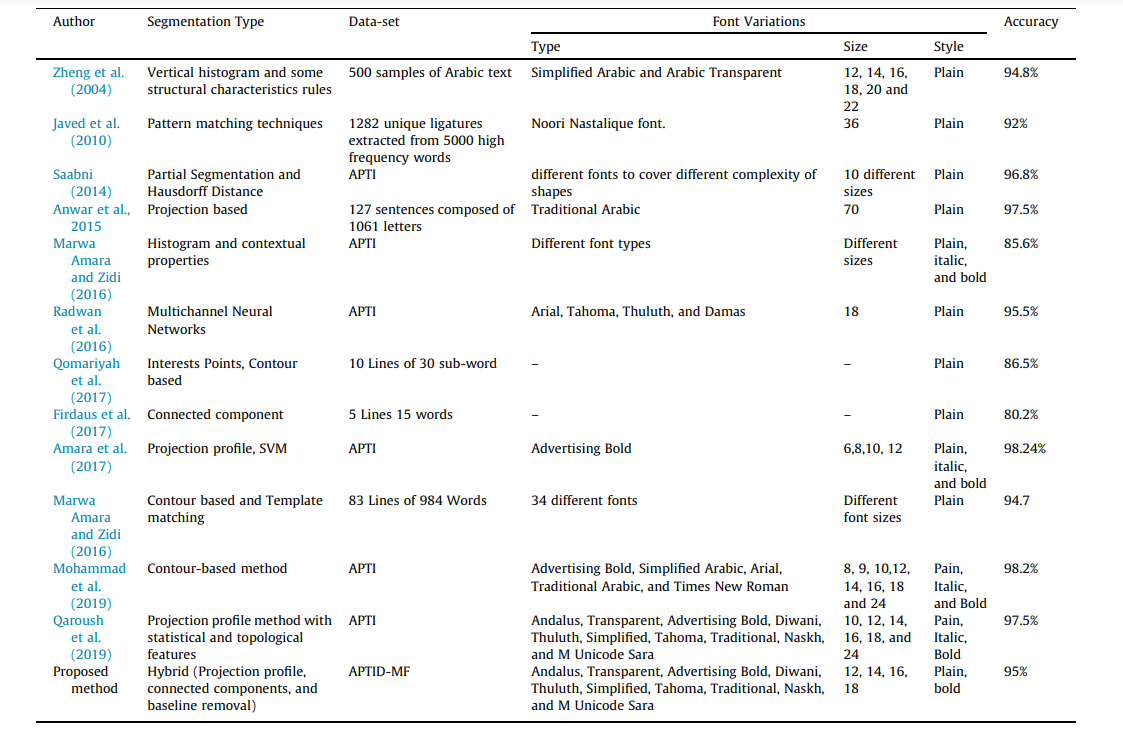
Alphanumeric VGG :

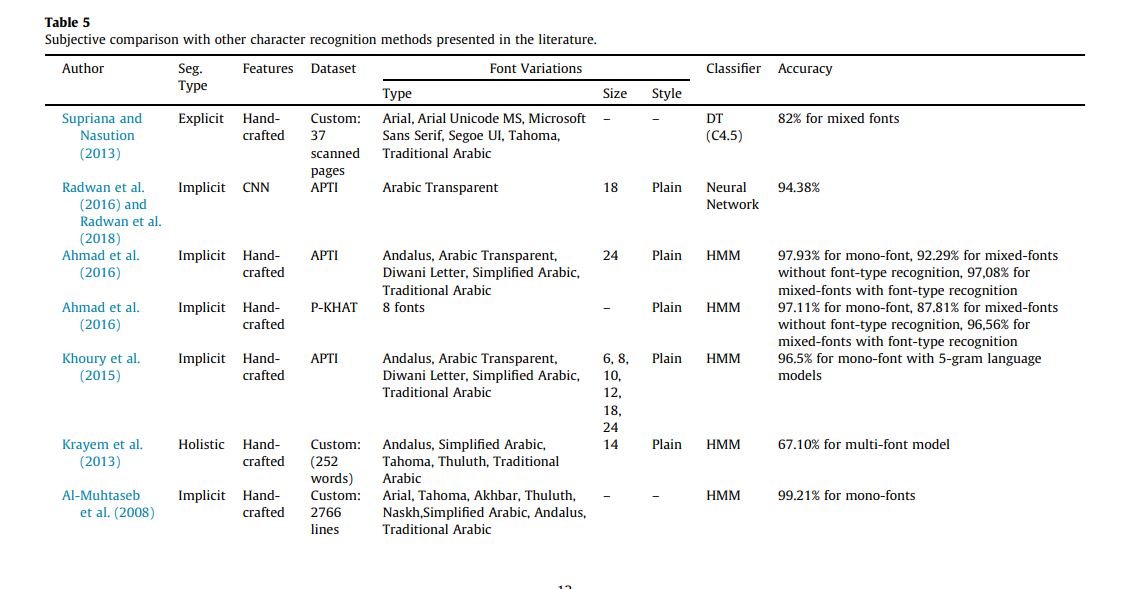
accuracy : 97.32% for the HACDB database and 99.66% for the ADBase database.

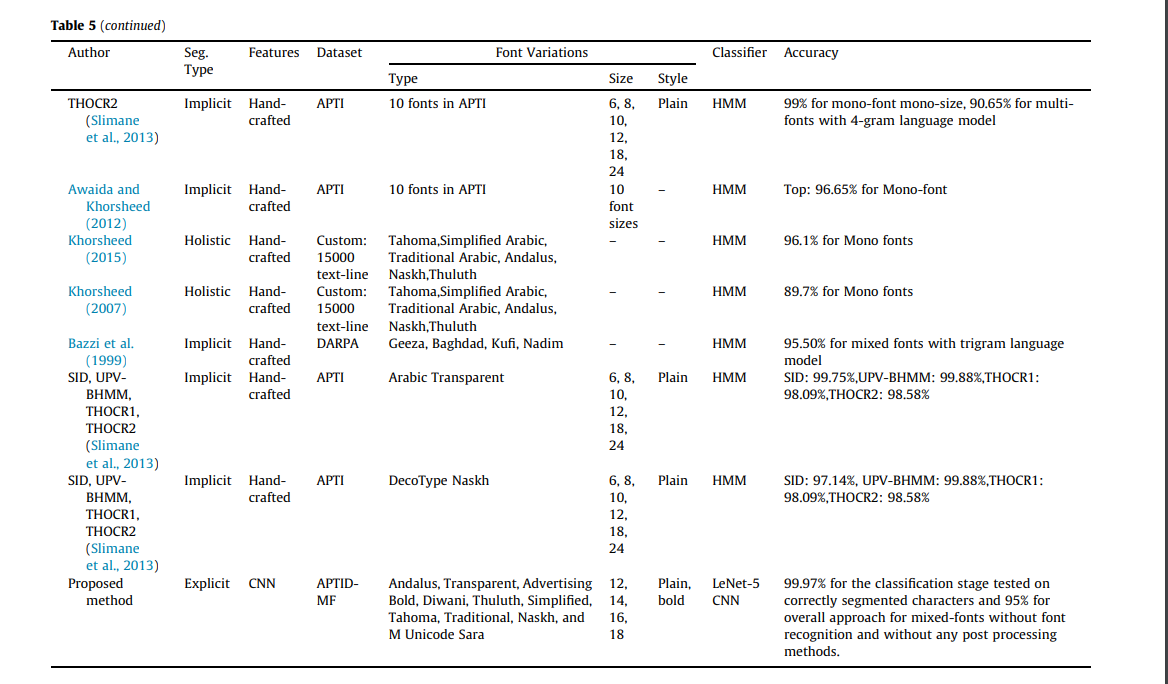
TABLES COMPARATIFS :

 DATASETS :









Quelques autres reference :

file:///C:/Users/pc/Downloads/forecasting-03-00033-v2%20(1).pdf

<https://github.com/cherry247/OCR-bill-detection/blob/master/ocr.ipynb>

<https://francescopochetti.com/easyocr-vs-tesseract-vs-amazon-textract-an-ocr-engine-comparison/>

<https://github.com/tukeyclothespin/scimitar>

<https://github.com/aub-mind/arabert>

<https://github.com/zacharywhitley/awesome-ocr>

<https://github.com/PaddlePaddle/PaddleOCR>