Université des sciences et de technologies Houari Boumedien

Département d'Informatique

Reconnaissance du manuscrit arabe en utilisant le Deep learning

Master 2 SII -Projet du module TAD

Réalisé par :

- ABDELALI Asma Nihad (Groupe: 1)
- DJEBROUNI Radhia (Groupe: 2)

Responsable du Module: Mme Zahia TAMEN

Introduction

Les tâches de reconnaissance visuelle faciles pour l'être humain sont en réalité bien compliquées pour la machine. L'être humain voit la forme de l'objet en 3d alors que l'ordinateur n'a qu'une matrice de pixels qu'il doit comprendre.
Détecter les couleurs, capturer et afficher des images ne nécessite pas un grand effort à la machine cependant, reconnaître ce qu'une image représente est d'une plus grande difficulté vu qu'un ordinateur n'a pas les mêmes expériences ni la compréhension nécessaires pour accomplir la tâche.

Computer vision est le domaine des sciences informatiques et de l'intelligence artificielle qui se base à reproduire la complexité de la vision humaine et extraire des informations à partir d'une image. Ce domaine est utilisé lors des reconnaissances faciales, reconnaissance des manuscrits, la réalité augmenté...etc
Un exemple de l'utilisation de ce domaine, c'est les voitures autonomes qui évitent les obstacles et reconnaissent les routes.
Ce qui a énormément contribué à l'avancement de la Computer vision et d'améliorer les résultats obtenus c'est les réseaux de neurones et le deep learning.

Les réseaux de neurones

En référence aux systèmes de neurones trouvés en nature et spécialement chez l'être humain, les réseaux de neurones sont des algorithmes qui ont pour but de reconnaître les relations entre des données. En s'entraînant en prenant plusieurs exemples d'entrées et des résultats prédéfinis les réseaux de neurones peuvent s'adapter à donner un résultat pour des différentes données en entrée.
L'entraînement d'un réseau de neurones se fait en lui donnant un ensemble d'apprentissage ainsi qu'une supervision humaine où on doit changer les paramètres jusqu'à avoir un résultat avec le maximum de précision.
Les réseaux de neurones peuvent avoir plusieurs couches de neurones, à un certain niveau de couches on arrive à ce qu'on appelle le deep learning. Chaque neurone du réseau traite une partie de l'information, les résultats donnés par tous les neurones sont influencés par des poids et envoyés à la couche suivante de neurones jusqu'à la dernière couche qui retourne un résultat.

Le Deep learning

Le deep learning est basé sur les réseaux de neurones et utilise des exemples pour apprendre et résoudre des problèmes, il utilise les modèles fréquents entre tous les exemples qui lui sont présentés et les transforme en formule mathématique, et il s'améliore avec le temps.

Vocabulaire

Train , Test , Validation set: Train set est l'ensemble de données le modèle apprend avec, Validation set est principalement utilisé pour décrire l'évaluation des modèles lors du réglage des hyperparamètres et de la préparation des données, et le Test set est principalement utilisé pour décrire l'évaluation d'un modèle final réglé lors de sa comparaison avec d'autres modèles finaux.

Les réseaux de convolution neurale: c'est des réseaux de neurones de convolution appelés CNN qui procèdent en traitant des des petites quantités d'information.

conv2D: c'est de convoluer une matrice en de plus petites matrices, en sachant qu'en informatique les images sont considérés comme des matrices, elle sert à trouver les ressemblances entre deux images partie par partie, conv1D est la fonction qui traite les vecteurs unidimensionnels.

max pool 2D: le max pooling 2D est l'opération de prendre des données 2 dimensionnelles et de prendre le maximum en se limitant à une plage donnée. Une matrice 4x4 va donner une plus petite matrice de dimension 2x2 contenant le maximum de la région.

Learning rate: le taux d'apprentissage d'un réseau de neurones est le paramètre qui contrôle le niveau de changement des poids des neurones en se basant sur le taux d'erreurs.

Activation functions: Ce sont les fonctions mathématiques appliquées au signal de sortie d'un neurone et qui permettent l'apprentissage de données complexes.

- **Rectified Linear Units (ReLu):** ayant la formule $ReLu = max(0, z)$ élimine les valeurs négatives du signal.
- **Exponential Linear Unit (Elu):** est une fonction qui converge le signal à 0. sa formule est :

$$Elu(z) = z \quad \text{si } z \geq 0$$

$$Elu(z) = \alpha \times (e^{z} - 1) \quad \text{Sinon}$$

  avec α une constante positive.
- **Softmax**: Est une fonction qui calcule les probabilités de distribution d'une classe sur les autres cibles possibles, peu importe l'entrée de Softmax le résultat va être compris entre 0 et 1 et la somme de toutes les sorties va être égale à 1. Le résultat de cette fonction représente la probabilité de la classe. Sa formule est la suivante:

$$\sigma(z)_j = \frac{e^{zj}}{\sum_{k=1}^{K} e^{zk}} \; pour\; j = 1, \dots K$$

Tensorflow: est un outil Open-Source créé par google pour le développement des modèles d'apprentissages intelligents.

Keras Tuner: c’est un outil de traitement d'hyper paramètres (paramètres utilisés pour contrôler le processus d'apprentissage), Keras Tuner trouve lui-même les meilleurs paramètres pour minimiser les erreurs.

Dropout: c'est une technique de régularisation pour contrer l'over fitting dans un réseau de neurones il consiste à abandonner des neurones et les supprimer dans le but d'optimiser l'apprentissage. à chaque étape de l'entraînement chaque neurone a une probabilité d'être ignoré.

Loss function 'categorical cross entropy': c'est une fonction du module Keras qui sert à classifier les données, elle retourne un vecteur où chaque case représente une classe et contient la probabilité que notre entrée appartient à cette classe.

1- Problématique :

Dans ce projet , on aura de partie, la première consiste à trouver un modèle de deep learning pour la reconnaissance du manuscrit arabe, en faisant des tests sur les différents hyperparamètres possibles, et en définissant ensuite le meilleur modèle trouvé.
Pour la 2eme partie, on va exploiter des differents techniques d'augmentation de données, et étudie leurs effets sur les modèles de la reconnaissance du manuscrit arabe

1- Environnement de travail :

Langage : Python
Editor : Notebook Collaboratory, Vscode
Libraries : os, zipfile, random, tensorflow, shutil, matplotlib-pyplot, keras_tuner, pyqt5, numpy
Environnement matériel:
Machine virtuelle fournis par Google colabs, avec 12GB de RAM , 60gb pour le disk, et GPU gratuit.

2- Analyse de données :

Les données fournies sont présentées sous forme d'un dossier contenant 28 dossiers représentant les classes qu'on va traiter, chaque classe contient 34/35 images des mots représentés en langue arabe de la même classe, 926 images.

3- Etape de Réalisation de la premier partie:

a- Prétraitement des données :

-Ajouter le document AHDB.zip à l'nvironnement de travail, l'extraire dans un nouveau dossier
-Importer les différentes librairies dont on aura besoin
-Créer trois dossiers Train, validation, test n on copie un pourcentage de données de chaque classe , on l'insert dans le dossier de classe dans le dossier train, on fait la même chose pour les deux autres dossiers, et ceci en utilisant la méthode :

```
split_data(SOURCE, TRAINING,VALIDATION, TESTING, TRAIN_SIZE,
VAL_SIZE):
```

source : le dossier source extrait

Training, validation, testing, les chemin reliés à chaque dossier

Train_size, val_size : le pourcentage des données transformées pour le dossier de training, validation respectivement, et le reste pour les images tests

- Séparer Labels et les images, dans notre cas, le target de chaque image est le nom de sa classe (son dossier) , en utilisant ImageDataGenerator.flow_from_directory, on va traiter les labels des dossiers automatiquement , tel que le nom du premier dossier "AAN" représente le label 0, le dernier dossier "yakon" représente le label 27, est toute ses images auront cette valeurs comme label, et ainsi de suite .

exemple pour le dossier validation

```
VALIDATION_DIR ='/tmp/recognize/validation/'
validation_datagen = ImageDataGenerator( rescale = 1.0/255. )
validation_generator =
validation_datagen.flow_from_directory(VALIDATION_DIR,batch_size=10,
class_mode  = 'categorical',target_size = (150, 150))
```

b- Apprentissage

- On va traiter les données en utilisant 3 couches de réseaux de neurones convolutionnels, une couche cachée, et une couche de sortie qui contient 28 neurones selon le nombres de classes, on parcourant les hyper paramètres suivants afin de trouver les meilleurs paramètres :

  fonction d'activation =["relu","elu"]
  nombre de neurones dans la couche caché =[32, 64,96,....,512]
  learning rate = [1e-2, 1e-3, 1e-4]
  loss function ='categorical_crossentropy'
  (car on a plusieurs classes)
  70% données train, 15% validation, 15% test

Voila la simulation du modèle en utilisant la librairie tenserflow.keras :

```
def model_builder(hp):

 hp_units = hp.Int('units', min_value=32, max_value=512, step=32)
 hp_activation=hp.Choice('activation',values=["relu","elu"])
 model = tf.keras.models.Sequential([

     tf.keras.layers.Conv2D(16, (3,3), activation=hp_activation,
input_shape=(150, 150, 3)),
     tf.keras.layers.MaxPooling2D(2,2),
     tf.keras.layers.Conv2D(32, (3,3), activation=hp_activation),
     tf.keras.layers.MaxPooling2D(2,2),
     tf.keras.layers.Conv2D(64, (3,3), activation=hp_activation),
     tf.keras.layers.MaxPooling2D(2,2),
     tf.keras.layers.Conv2D(128, (3,3), activation=hp_activation),
     tf.keras.layers.MaxPooling2D(2,2),
     tf.keras.layers.Flatten(),
     tf.keras.layers.Dense(hp_units, activation=hp_activation),
     tf.keras.layers.Dense(28, activation='softmax')
 ])

 # Choose an optimal value from 0.01, 0.001, or 0.0001
 hp_learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])
 model.compile(optimizer=RMSprop(lr=hp_learning_rate),
loss='categorical_crossentropy', metrics=['acc'])

 return model
```

- on utilisant la librarie `keras_tuner.Hyperband, on lance l'apprentissage` du modèle précédent et en parcourant les différents paramètres afin de trouver la meilleur validation accuracy, 15 epochs chacun

```
Total elapsed time: 00h 58m 42s

Search: Running Trial #29

Hyperparameter    |Value             |Best Value So Far
units             |160               |448
activation        |elu               |relu
learning_rate     |0.0001            |0.001
tuner/epochs      |15                |15
tuner/initial_e...|0                 |5
tuner/bracket     |0                 |2
tuner/round       |0                 |2

Epoch 1/15
88/88 [==============================] - 24s 258ms/step - loss: 2.8125 - a
Epoch 2/15
88/88 [==============================] - 23s 257ms/step - loss: 1.4512 - a
```

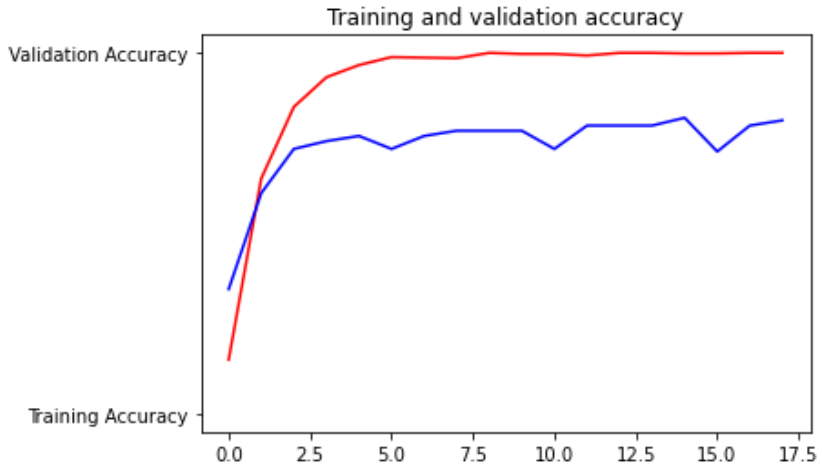
Meilleur modèle trouvé

```
Trial 30 Complete [00h 03m 24s]
val_acc: 0.6906474828720093

Best val_acc So Far: 0.7769784331321716
Total elapsed time: 00h 51m 06s
INFO:tensorflow:Oracle triggered exit

The hyperparameter search is complete. The optimal number
of units in the first densely-connected
layer is 384 and the optimal learning rate for the
optimizer is 0.001.
activation_func : relu
```

-Affichages des diagrammes avec la librairie matplotlib :



temps d'exécution : 1h 23min

c -Evaluation du modèle trouvé sur le données de test

Voila le resultat trouvé :

```
x_test,y_test=testing_generator.next()
eval_result = hypermodel.evaluate(x_test, y_test)
print("[test loss, test accuracy]:", eval_result)
```

```
1/1#evaluate the model with test data
1/1 [==============================] - 0s 250ms/step - loss: 0.2481 -
acc: 0.9000
[test loss, test accuracy]: [0.24806885421276093, 0.8999999761581421]
```

On remarque que test_accuracy et test_loss ne sont pas parfait, signifiant que le dernier modèle peut ne pas reconnaître des images similaires au données traitées.

4 - Etude de l'effet de quelque techniques d'augmentation de données

Pour cette phase , on a ajouté les techniques d'augmentation suivantes au code précédent (en ajoutant un Dropout avant la couche cachée :

1-Augmentation avec les paramètres suivants :

```
training_datagen = ImageDataGenerator(
      rescale = 1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      zoom_range=0.2,
      featurewise_center=True,
      featurewise_std_normalization=True,
      zca_whitening=True,
      brightness_range=[0.4,1.5],
      fill_mode='nearest')
```

Définition des paramètres:

- **rescale**: le facteur de redimensionnement (rescaling), si la valeur est nulle un redimensionnement est appliqué sinon on multiplie les données par la valeur fournie.

- **rotation range**: Le degré de l'angle des rotations

- **width shift range**: peut être un nombre flottant, un vecteur unidimensionnel ou un entier, dans notre cas un nombre flottant qui représente la fraction du total de la largeur.

- **height shift range**: peut être un nombre flottant, un vecteur unidimensionnel ou un entier, dans notre cas un nombre flottant qui représente la fraction du total de la hauteur.

- **zoom range**: l'intervalle de zoom (agrandissement)

- **featurewise center**: un booléen qui met les moyennes des échantillons à 0.

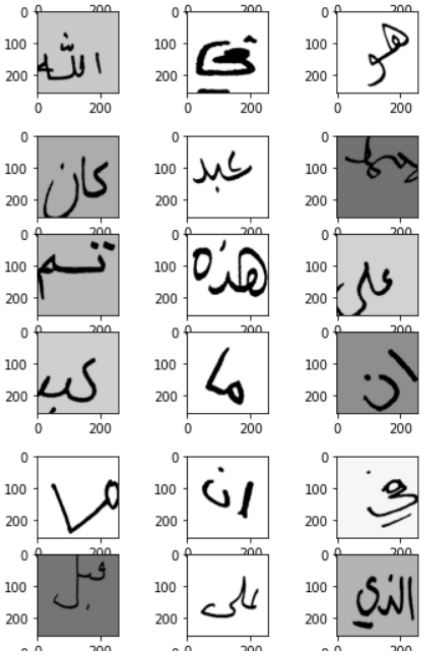
- **featurewise std normalization**: un booléen qui divise les entrées par std du jeu de données par fonctionnalité.

- **zca whitening**: Une transformation de blanchiment est une transformation linéaire qui transforme un vecteur de variables aléatoires avec une matrice de covariance connue en un ensemble de nouvelles variables dont la covariance est la matrice d'identité.

- **brightness range**: une liste de valeurs ou intervalle de la luminosité

- **fill mode**: peut être l'un des valeurs suivantes {"constant", "nearest", "reflect" or "wrap"}, "nearest" est la valeur par défaut, c'est le mode de remplissage. Les points en dehors des limites de l'entrée sont remplis selon le mode donné.
  pour nearest le mode est le suivant: 'nearest': aaaaaaaa|abcd|dddddddd

Exemples des données augmentés :



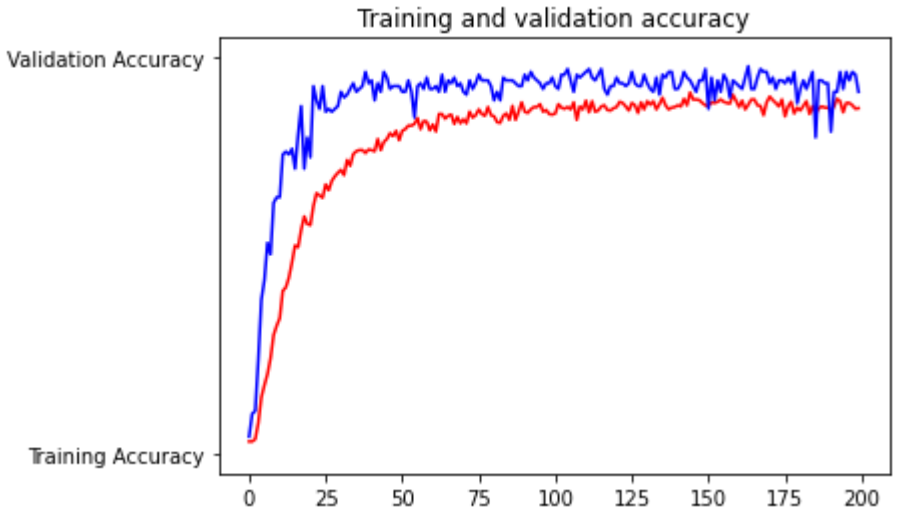
Résultats des meilleurs hyperparamètres trouvés en 15 epochs:

```
Trial 30 Complete [00h 01m 31s]
val_acc: 0.7338129281997681

Best val_acc So Far: 0.7338129281997681
Total elapsed time: 00h 16m 40s
INFO:tensorflow:Oracle triggered exit

The hyperparameter search is complete. The optimal number of units in
the first densely-connected
layer is 288 and the optimal learning rate for the optimizer
is 0.001, the optimal activation func is relu,
the optimal dropout value is 0.2.
```

Les résultats après apprentissage avec 200 epoch :
temps d'exécution = plus de 2h, qui est beaucoup plus que les resultats precedents

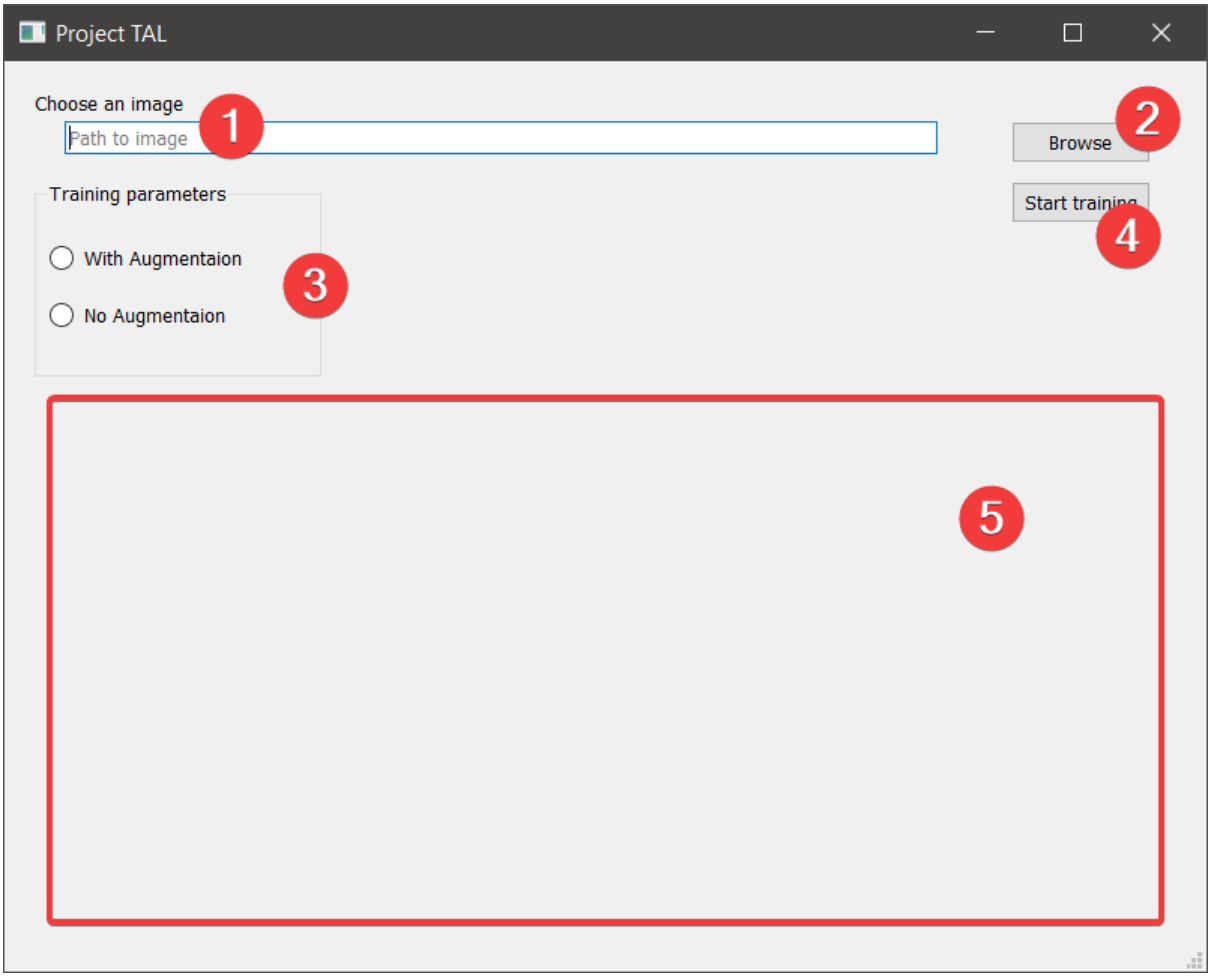


Evaluate the model with test data

```
1/1 [==============================] - 0s 191ms/step - loss:
2.5324e-04 - acc: 1.0000
[test loss, test accuracy]: [0.0002532407233957201, 1.0]
```

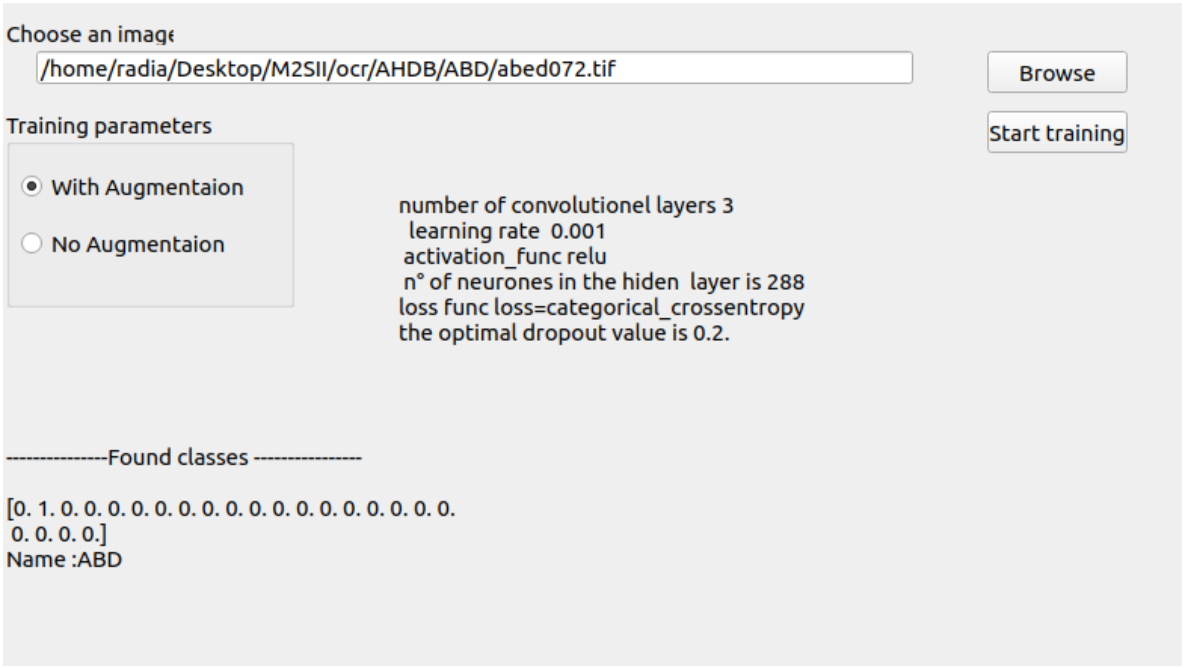
On remarque que les résultats après augmentation sont beaucoup meilleurs que les résultats précédent, les diagrammes convergent sans overfitting , et ceci car le volume de données a augmenté après l'application de ses techniques en ajoutant des copies modifiées pour chaque image, et ce qui a engendré plus de temps pour le temps d'exécution.

Interface

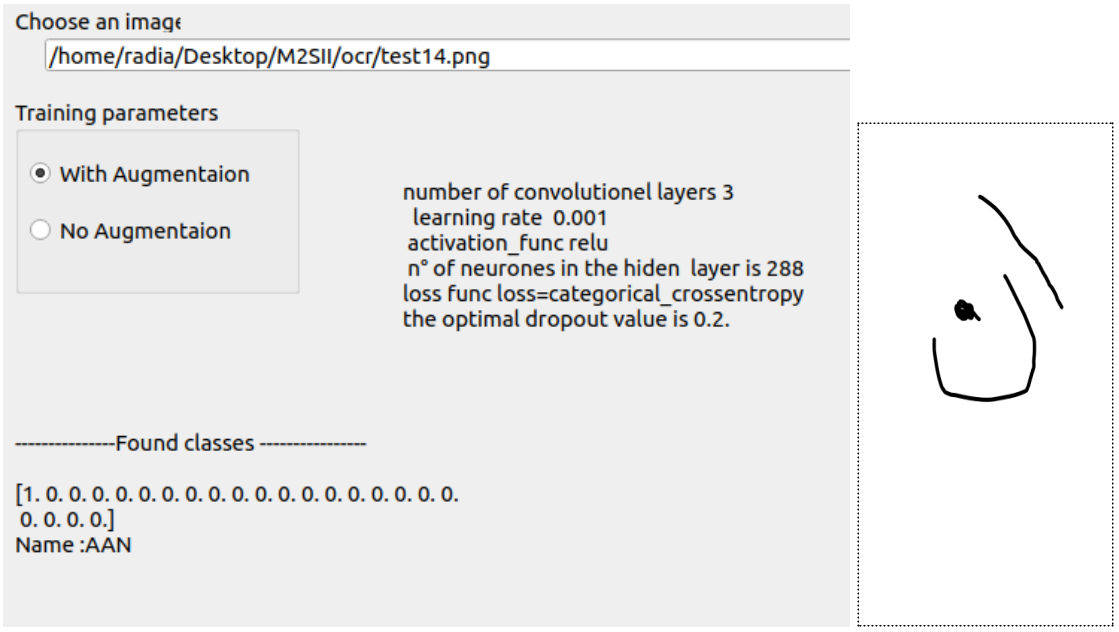


1. Le chemin de l'image à traiter.
2. Le bouton pour choisir l'image sous formats .png, .jpg ou .tif (on laisse le choix à l'utilisateur pour entrer le chemin ou utiliser le bouton).
3. Le type d'augmentation voulu.
4. Le bouton pour commencer l'entraînement.
5. La zone d'affichage du résultat.

Test the augmentation model with abd image:



Test sur une nouvelle image crée par l'utilisateur



Conclusion

Dans ce projet, nous avons créé dans la première partie un modèle de reconnaissance du manuscrit arabe en permutant entre les différents hyperparamètres afin de trouver ceux qui forment le meilleur modèle. Dans la 2 eme partie, nous avons étudié l'effet de l'ajout de quelque technique d'augmentation au modèle, les résultats ete beaucoup plus meuilleurs a cause du nombre de nouvelles image générées. A la fin, on a créé une interface visuelle afin de tester différentes images et comparer entre les deux modèles formés.
Comme perspective, nous voulant exploiter d'autres techniques d'augmentation de données et en utilisant un Dataset plus grand, afin de trouver de développer les modèles et trouvez resultats plus meilleurs

Ressources:

https://colab.research.google.com/

https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/

https://machinelearningmastery.com/image-augmentation-deep-learning-keras/