Implémentation et réalisation

Chapitre III: Implémentation et réalisation

III.1 Introduction

Dans ce chapitre, nous allons présenter les expérimentations effectuées sur notre modèle afin de démontrer son efficacité à détecter covid 19 a partir de Scanner thoracique. Nous détaillerons d'abord les environnements matériels et logiciels que nous avons utilisés pour développer notre application, puis nous comparerons les résultats.

III. 2 Environnement de développement logiciel

III. 2.1 TensorFlow

TensorFlow[1] est un framework de programmation pour le calcul numérique qui a été rendu Open Source par Google en Novembre 2015. Depuis son release, TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des frameworks les plus utilisés pour le Deep Learning et donc les réseaux de neurones. Son nom est notamment inspiré du fait que les opérations courantes sur des réseaux de neurones sont principalement faites via des tables de données multi-dimensionnelles, appelées Tenseurs (Tensor). Un Tensor à deux dimensions est l'équivalent d'une matrice. Aujourd'hui, les principaux produits de Google sont basés sur TensorFlow: Gmail, Google Photos, Reconnaissance de voix.

III. 2.2 Keras

Keras[2] est une API de réseaux de neurones de haut niveau, écrite en Python et capable de fonctionner sur TensorFlow ou Theano. Il a été développé en mettant l'accent sur l'expérimentation rapide. Être capable d'aller de l'idée à un résultat avec le moins de délai possible est la clé pour faire de bonnes recherches. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), et son principal auteur et mainteneur est François Chollet, un ingénieur Google.

En 2017, l'équipe TensorFlow de Google a décidé de soutenir Keras dans la bibliothèque principale de TensorFlow. Chollet a expliqué que Keras a été conçue comme une interface plutôt

1. https://www.tensorflow.org/
2. https://keras.io/

que comme un cadre d'apprentissage end to end. Il présente un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilitent la configuration des réseaux neuronaux indépendamment de la bibliothèque informatique de backend. Microsoft travaille également à ajouter un backend CNTK à Keras aussi.

III. 2.3 Python

Python[3] est un langage de programmation de haut niveau interprété (il n'y a pas d'étape de compilation) et orienté objet avec une sémantique dynamique. Il est très sollicité par une large communauté de développeurs et de programmeurs. Python est un langage simple, facile à apprendre et permet une bonne réduction du cout de la maintenance des codes. Les bibliothèques (packages) python encouragent la modularité et la réutilisabilité des codes.

III. 2.4 Scikit-learn

Scikit-learn est une bibliothèque libre Python dédiée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria et Télécom ParisTech. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec des autres bibliothèques libre Python, notamment NumPy et SciPy.

III. 2.5 Anaconda

Anaconda[4] est une distribution libre et open source des langages de programmation Python et R appliquée au développement d'applications dédiées à la fouille de données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement. Les versions dépaquetages sont gérées par le système de gestion de paquets conda. La distribution Anaconda est utilisée par plus de 6 millions d'utilisateurs et comprend plus de 250 paquets populaires fouille de des données adaptés pour Windows, Linux et MacOs.

3. https://www.python.org/
4. https://www.anaconda.com/

III. 2.6 Pycharm IDE

PyCharm est un environnement de développement intégré utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django. Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plate forme qui fonctionne sous Windows, Mac OS X et Linux. Il est décliné enédition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous licence Apache.

III. 2.7 PyQt

PyQt[5] est l'une des liaisons Python les plus populaires pour le framework C ++ multiplateforme Qt. PyQt a été développé par Riverbank Computing Limited. Qt lui-même est développé dans le cadre du projet Qt. PyQt fournit des liaisons pour Qt 4 et Qt 5. PyQt est distribué sous un choix de licences: GPL version 3 ou une licence commerciale.

PyQt est disponible en deux éditions: PyQt4 qui compilera contre Qt 4.x et 5.x et PyQt5 qui ne compilera que contre 5.x. Les deux éditions peuvent être construites pour Python 2 et 3. PyQt contient plus de 620 classes qui couvrent les interfaces utilisateur graphiques, la gestion XML, la communication réseau, les bases de données SQL, la navigation Web et d'autres technologies disponibles dans Qt.

III. 2.8 OpenCV

OpenCV[6] (Open source Computer Vision) est une bibliothèque libre de vision par ordinateur, qui possède plus de 2500 algorithmes optimisés, La bibliothèque comprend un ensemble complet d'algorithmes de vision par ordinateur et d'algorithmes d'apprentissage. Elle existe depuis une décennie et est publiée sous la licence Berkeley Software Distribution (BSD), ce qui facilite l'utilisation et la modification du code par les utilisateurs.

Les modules intégrés d'OpenCV sont suffisamment puissants et polyvalents pour résoudre la plupart des problèmes de vision informatique pour lesquels des solutions bien établies sont

5. https://pypi.org/project/PyQt5/
6. https://opencv.org/

disponibles. Ils sont destinés à des applications en temps réel et conçus pour être exécutés très rapidement.

OpenCV a été officiellement lancé en tant que projet de recherche au sein d'Intel Research afin de faire progresser les technologies dans les applications gourmandes en ressources. Parmi les principaux contributeurs au projet figuraient des membres d'"Intel Research Russia" et de l'équipe "Performance Library d'Intel".

III. 3 Configuration matérielle

III.3.1 Configuration matérielle distante (Google Colab)

Google Collaboratory[7] ou Colab1, un outil Google simple et gratuit pour vous initier au Réseaux profonds ou collaborer avec vos collègues sur des projets en science des données .Colab permet d'améliorer vos compétences de codage en langage de programmation Python, de développer des applications en Réseaux Profonds en utilisant des bibliothèques populaires telles que Keras, TensorFlow, PyTorch et OpenCV sans installation ainsi que d'utiliser un environnement de développement (Jupyter Notebook) qui ne nécessite aucune configuration. Cependant, chaque 12 heures, la machine virtuelle mise à disposition par Google est réinitialisée nécessitant un mécanisme de sauvegarde des données en cours .De plus, les documents Colab (Jupyter Notebook) sont enregistrés directement votre compte Google Drive**.**

L'infrastructure distante mise à disposition par Google Colab et utilisée pour l'entraî-nement possède cette configuration :

- Processeur Intel Core Xeon CPU @2.3Ghz, 45MB Cache.
- Processeur graphique NVIDIA Tesla K80, ayant 2496 cœurs CUDA, Compute 3,7, 12 Go (11.439 Go utilisable) GDDR5 VRAM.
- Mémoire vive de 12.6 Go.
- Disque dur de capacité 320 Go.
- Jupyter Notebook
- Système d'exploitation Linux x64 bits.

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

III.3.2 Configuration matérielle local

Pour la reconnaissance, nous avons utilisé un pc portable personnel possédant cette configuration:

- Processeur Intel Core i5-9400f CPU @3.5Ghz,5 .0 GhZ.
- Processeur graphique NVIDIA GEFORCE GTX 1060 3Go VRAM
- Mémoire vive de 8 Go
- Disque dur hybride SSD de capacité 256 Go et HDD de capacité 1 To
- Système d'exploitation Windows 10 x64 bits

III.3.3 Les base d'images

Kaggle[8] est une collection de données scientifiques. Kaggle permet aux utilisateurs de rechercher et de publier des ensembles de données et des explorateurs, de créer des modèles dans un environnement de science des données basé sur le Web, de travailler avec d'autres données scientifiques et des moteurs de correspondance automatique, et de participer à des concours pour résoudre les failles de données.

III.3.4 Architecture de notre réseau

Au cours de nos expérimentations, nous avons créé trois modèles (modèle 1, modèle 2 et modèle 3) avec différentes architectures, où on a appliqué les trois modèles sur la base d'images
Dans ce qui suit on présente l'architecture des trois modèles :

Architecture du modèle 01

Le premier modèle que nous présentons dans la figure III.1 est composé de cinq couches de convolution et deux couches de maxpooling et trois couches de fullyconnected.
L'image en entrée est de taille 244*244, l'image passe d'abord à la première couche de convolution. Cette couche est composée de 32 filtres de taille 3*3, Chacune de nos couches de convolution est suivie d'une fonction d'activation ReLU cette fonction force les neurones à retourner des valeurs positives, après cette convolution 32 featuresmaps de taille 242*242 seront créés.

8. https://www.kaggle.com/

Les 32 featuremaps qui sont obtenus auparavant sont donnés en entrée de la deuxième couche de convolution qui est composée aussi de 32 filtres, une fonction d'activation RELU est appliquée sur la couche de convolution, ensuite on applique Maxpooling pour réduire la taille de l'image, nombre des paramètres et donc le temps de calcul. À la sortie de cette couche, nous aurons 32 featuremaps de taille 120*120.

On répète la même chose avec les couches de convolutions trois, quatre et cinq, ces couches sont composées de 64 filtres, la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après la couche de convolution cinq. À la sortie de cette couche, nous aurons 64 featuremaps de taille 57*57. Le vecteur de caractéristiques issu des convolutions a une dimension de 207936.

Après ces cinq couches de convolution, nous utilisons un réseau de neurones composé de 3 couches fullyconnected. La première couches est 1 024 neurones où la fonction d'activation utilisée est le ReLU, La 2 ème couches est 512 neurones où la fonction d'activation utilisée est le ReLU, et la 3 ème couche est un sigmoid qui permet de calculer la distribution de probabilité des 2 classes (nombre de classe dans la base d'image).

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 242, 242, 32)	896
conv2d_10 (Conv2D)	(None, 240, 240, 32)	9248
max_pooling2d_6 (MaxPooling2	(None, 120, 120, 32)	0
conv2d_11 (Conv2D)	(None, 118, 118, 64)	18496
conv2d_12 (Conv2D)	(None, 116, 116, 64)	36928
conv2d_13 (Conv2D)	(None, 114, 114, 64)	36928
max_pooling2d_7 (MaxPooling2	(None, 57, 57, 64)	0
flatten_2 (Flatten)	(None, 207936)	0
dense_6 (Dense)	(None, 1024)	212927488
dropout_4 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 512)	524800
dropout_5 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 1)	513

figure III.1 Configuration du modèle 1

Architecture du modèle 02

Le deuxième modèle que nous présentons dans la figure III.2 est composé de Trois couches de convolution et trois couches de maxpooling et deux couches de fullyconnected.
L'image en entrée est de taille 244*244, l'image passe d'abord à la première couche de convolution. Cette couche est composée de 32 filtres de taille 5*5, Chacune de nos couches de convolution est suivie d'une fonction d'activation ReLU cette fonction force les neurones à retourner des valeurs positives, après cette convolution 32 featuresmaps de taille 240*240 seront créés.
Ensuite on applique Maxpooling pour réduire la taille de l'image ainsi la quantité de paramètres et de calcul. À la sortie de cette couche, nous aurons 32 featuremaps de taille 120*120
Les 32 featuremaps qui sont obtenus auparavant ils sont donnés en entrée de la deuxième couche de convolution qui est composée aussi de 64 filtres, une fonction d'activation RELU est appliquée sur la couche de convolution, ensuite on applique Maxpooling pour réduire la taille de l'image ainsi que le nombre de paramètres et le temps de calcul. À la sortie de cette couche, nous aurons 60featuremaps de taille 58*58.
On répète la même chose avec la couche de convolution trois, Cette couche est composéede 128 filtres, la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après la couche de convolution trois. À la sortie de cette couche, nous aurons 128 featuremaps de taille 27*27. Le vecteur de caractéristiques issu des convolutions a une dimension de 93312.
Après ces trois couches de convolution, nous utilisons un réseau de neurones composé de deux couches fullyconnected. la première couche est composé 1 024 neurones où la fonction d'activation utilisée est le ReLU, et la Deuxièm couche est un sigmoid qui permet de calculer la distribution de probabilité des 2 classes (nombre de classe dans la base d'image).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 240, 240, 32)      2432
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 120, 120, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 116, 116, 64)      51264
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 58, 58, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 54, 54, 128)       204928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 27, 27, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 93312)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              95552512
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 2050
=================================================================
Total params: 95,813,186
Trainable params: 95,813,186
Non-trainable params: 0
_________________________________________________________________
```

figure III.2 Configuration du modèle 2

Architecture du modèle 03

Le troisième modèle que nous présentons dans la figure III.3 est composé de six couches de convolution et trois couches de maxpooling et de trois couches de fullyconnected.

L'image en entrée est de taille 244*244, l'image passe d'abord à la première couche de convolution. Cette couche est composée de 32 filtres de taille 3*3, la fonction d'activation ReLU est utilisé, cette fonction d'activation force les neurones à retourner des valeurs positives, après cette convolution 32 featuresmaps de taille 242*242 seront créés.

Ensuite, les 32 featuremaps qui sont obtenus ils sont données en entrée de la deuxième couche de convolution qui est composée aussi de 32 filtres. La fonction d'activation ReLU est appliquée sur les couches de convolutions. Le Maxpooling est appliqué après pour réduire la taille de l'image et des paramètres. À la sortie de cette couche, nous aurons 32 featuremaps de taille 120*120.

On répète la même chose avec les couches de convolutions trois, quatre, cinq et six (les couches trois et quatre sont composées de 64 filtres et les couches cinq et six sont composées de 128 filtres), la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après les couches de convolutions quatre et six. À la sortie de la dernière couche Maxpooling, nous aurons 128 featuremaps de taille 27*27. Le vecteur de caractéristiques issu des convolutions a une dimension de 93312.

Après ces six couches de convolution, nous utilisons un réseau de neurones composé de 4 couches fullyconnected La première couches est 1 024 neurones où la fonction d'activation utilisée est le ReLU, La 2 ème couches est 512 neurones où la fonction d'activation utilisée est le ReLU, La 3 ème couches est 128 neurones où la fonction d'activation utilisée est le ReLU, et la 4 ème couche est un sgmoid qui permet de calculer la distribution de probabilité des 2 classes (nombre de classe dans la base d'image).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 242, 242, 32)      896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 240, 240, 32)      9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 120, 120, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 118, 118, 64)      18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 116, 116, 64)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 58, 58, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 56, 56, 128)       73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 54, 54, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 27, 27, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 93312)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              95552512
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
```

figure III.3 Configuration du modèle 3

III.3.5Résultats obtenus et discussion

Résultats obtenus pour le modèle 1

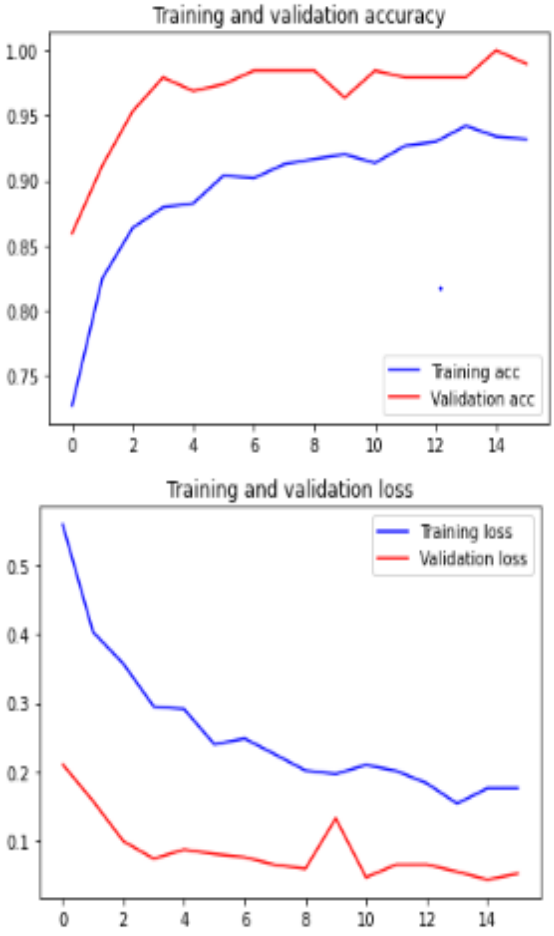


Figure III.4 Précision et Erreur pour le Modèle 01

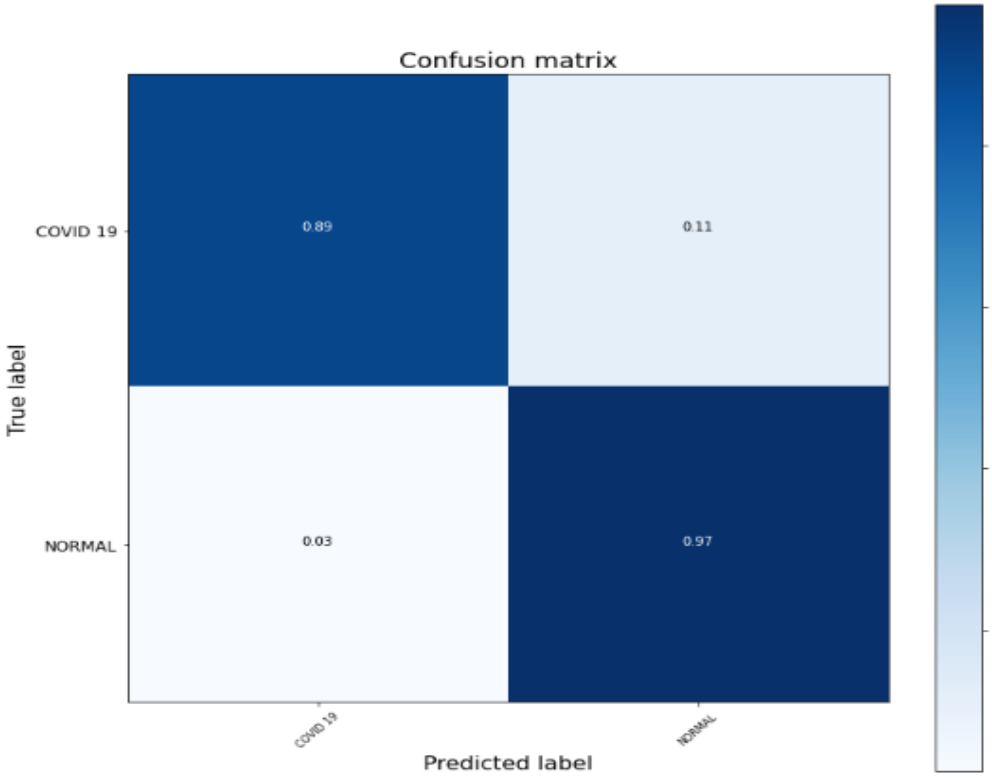


Figure III.5 Matrice de Confusion pour le Modèle 01

Résultats obtenus pour le modèle 2

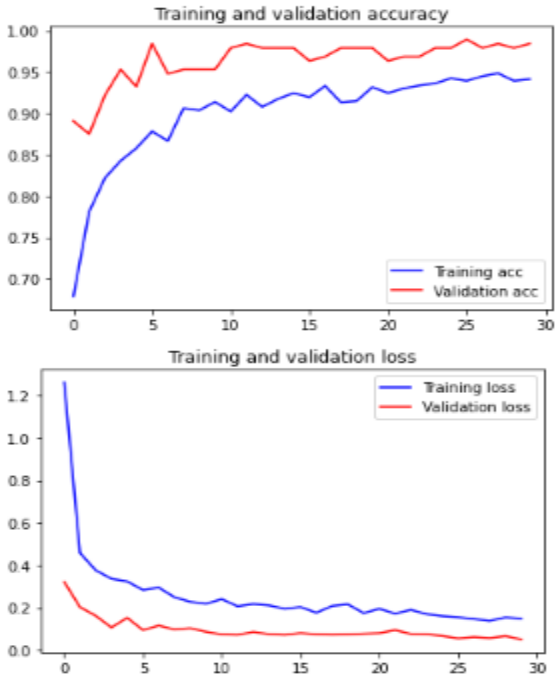


Figure III.6 Précision et Erreur pour le Modèle 02

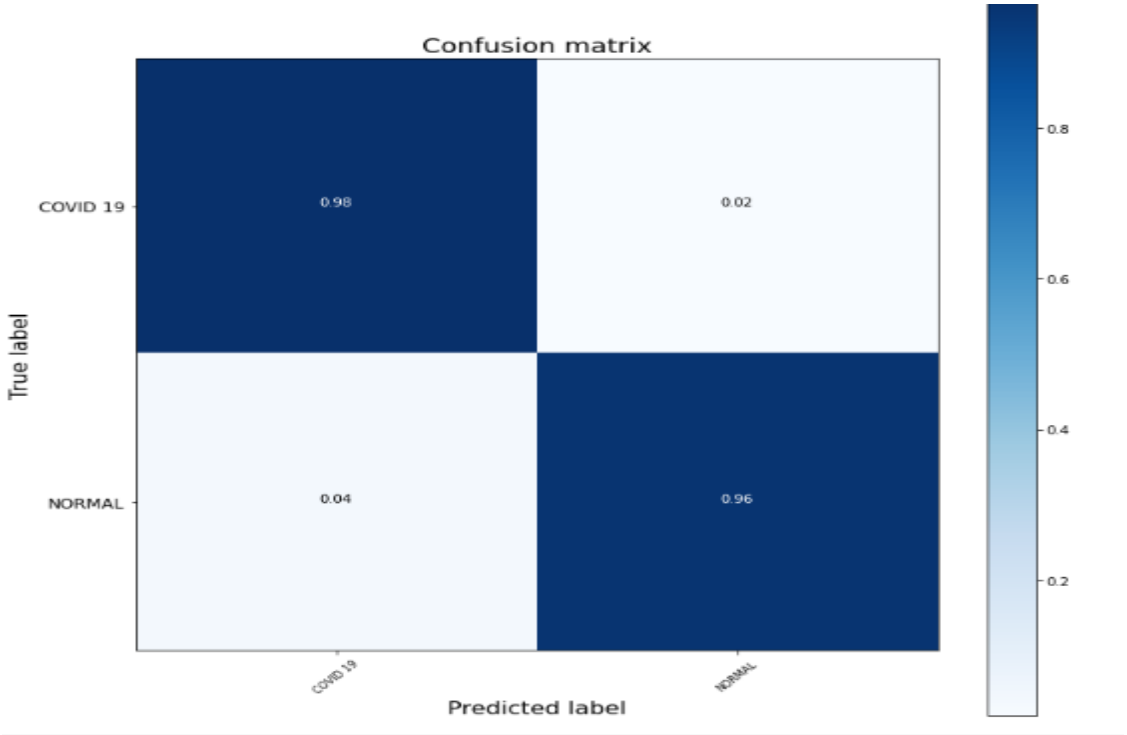


Figure III.7 Matrice de Confusion pour le Modèle 02

Résultats obtenus pour le modèle 3

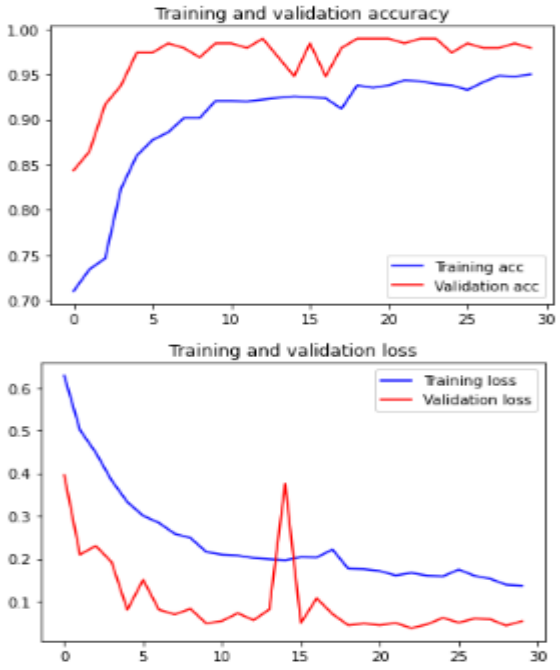


Figure III.8 Précision et Erreur pour le Modèle 03

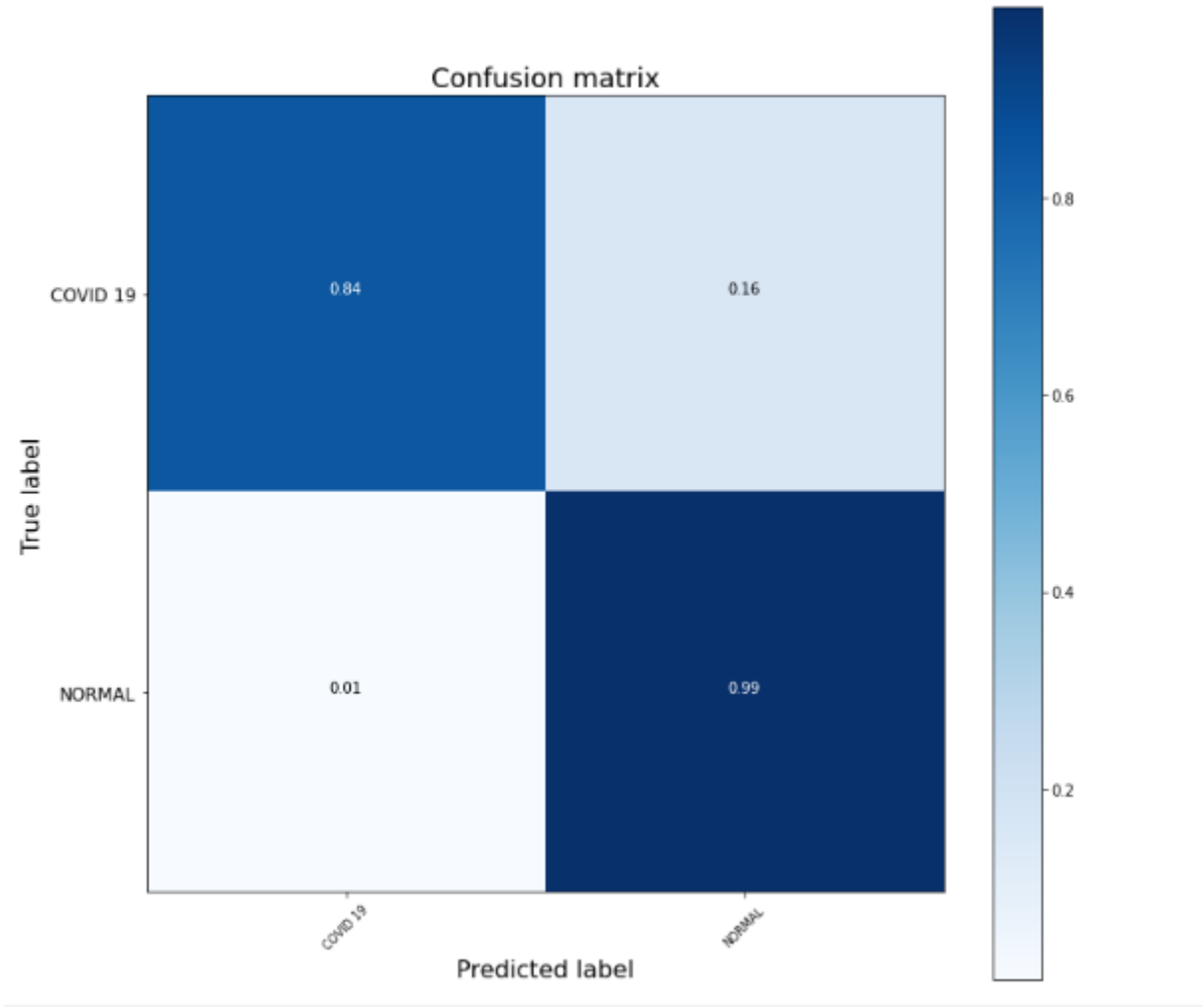


Figure III.9 Matrice de Confusion pour le Modèle 03

Après l'analyse des résultats obtenus, On constate les remarques suivantes :

La précision de l'apprentissage et de test augmente avec le nombre d'époques, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. Si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'époques et vice versa De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époques.

Nous remarquons aussi que la totalité des images mal classées est de 53 images a partir de 337 pour le model 1 , 21 images a partir de 337 pour le model 2 et 58 images a partir de 337 pour le model 3

Les matrices de confusion permettent d'évaluer la performance de nos modèles, puisqu'elle reflète les métriques du Vrai Positif, Vrai Négatif, Faux Positif et Faux Négatif

	Architecture utilisée			Nombre époques	Précision obtenu Sur la base d'apprentissage	Précision obtenu Sur la base de validation	Temps d'exécution
	Couche de conv	Couche de Pooling	Fully Connected				
Model 01	**5**	**2**	**3**	**10**	**91%**	**71%**	**900 s 15 min**
Model 02	**3**	**3**	**2**	**30**	**95%**	**83%**	**2040 s 30 min**
Model 0 3	**6**	**3**	**4**	**20**	**93%**	**74%**	**1320 s 22min**

Le tableau montre l'architecture utilisée dans chaque modèle ainsi que le nombre d'époques (passages sur l'ensemble des exemples de la base d'apprentissage). Les résultats obtenus sont exprimés en termes de précision d'apprentissage, de validation, et enfin la dernière colonne représente le temps d'exécution. Nous remarquons que le modèle 2 présente les meilleurs résultats trouvés. Le nombre d'époques et de couches de convolution reflètent ces bons résultats, cependant le temps d'exécution était très couteux (à cause du nombre d'époques)

Les résultats obtenus se sont améliorés au fur et à mesure que nous avons approfondie notre réseau et augmenté le nombre d'époques. La base d'apprentissage est également un élément déterminant dans les réseaux de neurones convolutionnels, il faut avoir une base d'apprentissage de grande taille pour aboutir à des meilleurs résultats**.**

III.4 Conclusion

Dans ce chapitre, nous avons expliqué la mise en œuvre de notre application ainsi que la configuration matérielle et logicielle utilisée. Nous avons également présenté nos résultats expérimentaux obtenus en faisant une étude comparative sur les trois modèles utilisés.

Conclusion générale

Conclusion générale

La classification d'images est une tâche importante dans le domaine de la vision par ordinateur, la reconnaissance d'objets et l'apprentissage automatique. Grâce à l'apprentissage en profondeur (Deep Learning), l'avenir de l'IA se développe très rapidement.

Dans ce projet nous avons entamé les notions fondamentales sur les réseaux de neurones en générale et les réseaux de neurones convolutionnels en particulier. Nous avons introduit ces réseaux de neurones convolutionnels en présentant les différents types de couches utilisées dans la classification: la couche convolutionnelle, la couche de correction, la couche de pooling et la couche de dropout. Nous avons parlé aussi sur les méthodes de régularisation (dropout et data augmentation) utilisées pour éviter le problème de sur apprentissage.

Pour réaliser notre travail de classification on a utilisé le deep learning, la méthode d`apprentissage qui a montré ses performances ces dernières années et nous avons choisi les CNNs comme réseau de neurones artificiels acycliques (feed-forward). Parmi les avantages des réseaux convolutifs est l'utilisation d'un poids unique associé aux signaux entrant dans tous les neurones d'un même noyau de convolution. Cette particularité nous a permis de réduire l'empreinte mémoire, améliorer les performances et garder une invariance du traitement par translation. Donc notre choix est justifié par la simplicité et l'efficacité de la méthode. Les résultats obtenus lors de la phase de test confirment l`efficacité de notre approche.

Notre travail n`est que dans sa version initiale, on peut dire que ce travail reste ouvert pour des travaux de comparaison et/ou d'hybridation avec d'autres méthodes de classification.

Bibliographie

Bibliographie

[01] Naciri H., Chaoui N, Conception et Réalisation d'un système automatique d'identification des empreintes digitales, Mémoire de PFE, Université de Tlemcen, 2003.

[02] Hadjila F. &Bouabdellah R, Reconnaissance des visages par les réseaux de neurones, Mémoire de PFE, Université de Tlemcen, 2003.

[03] Rafael C.Gonzalez& Richard E.Woods , Digital Image Processing, Pearson Education Inc, 2008

[04] Nasrin, Abdllaoui Nezha. la classification on hiérarchique axendante. 2013/2014.

[05] Hirotk Suzuki, Pascal Matsakis. Exploitation de connaissances structurelles en classification on d'image:Utilisation de méthodes heuristiques d'optimisation combinatoire.\

[06] BORGI A., AKDAG H. Supervised Learning and Approximate. 2001.
[07] R. S. Michalski, R.L. Chilausky. Knowledge acquisition by encoding expert rules versus computer induction from examples : a case study involving. 1981.

[08] Parrochia. Classifications, histoire et problèmes formels. Cinquièmes Rencontres de la société
Francophone de classification SFC'97. Lyon : s.n., Septembre 1997.

[09] R.E. Stepp, R.S. Michalski. Learning from Observation: Conceptual Clustering, Machine Learning, An Artificial Intelligence Approach. s.l. : Morgan Kaufman Publishers, 1983.

[10].Cours d'analyse numérique : réseaux de neurones formels(http://informatique.coursgratuits.net/methodes-numeriques/reseaux-deneurones-formels.php)
[11]. LNIA - Laboratoire de Neurosciences intégratives et adaptatives Claude Touze

[12]. Jean-Paul Haton, Modèles connexionnistes pour l'intelligence artificielle, 1989.

[13]. Léon Personnaz et Isabelle Rivals, Réseaux de neurones formels pour la modélisation, la commande et la classification, CNRS Éditions, 2003.

[14] Pierre Buyssens, Fusion de différents modes de capture pour la reconnaissance du visage appliquée aux e_ transactions DOCTORAT de l'UNIVERSIT´E de CAEN Le 4 Janvier 2011

[15] Yann LeCun. Les enjeux de la recherche en intelligence artificielle. Interstices, 2016.

[16] Grégory Gelly. Réseaux de neurones récurrents pour le traitement automatique de la parole. PhD thesis, Université Paris-Saclay, 2017.

[17] SébatienFrizzi, RabebKaabi, MoezBouchouicha, Jean-Marc Ginoux, Farhat Fnaiech, and Eric Moreau. Détection de la fumée et du feu par réseau de neurones convolutifs. In Conférence Nationale sur les Applications Pratiques de l ?Intelligence Artificielle, 2017.

Bibliographie

[18] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. arXiv preprint arXiv :1202.2745, 2012.
[19] Sparsh Mittal. A survey of fpga-based accelerators for convolutional neural networks. Neural computing and applications, pages 1–31, 2018.

[20]Francois Chollet. Deep Learning mit Python und Keras : Das Praxis-HandbuchvomEntwickler der Keras-Bibliothek. MITP-VerlagsGmbH& Co. KG, 2018

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1) :1929–1958, 2014.

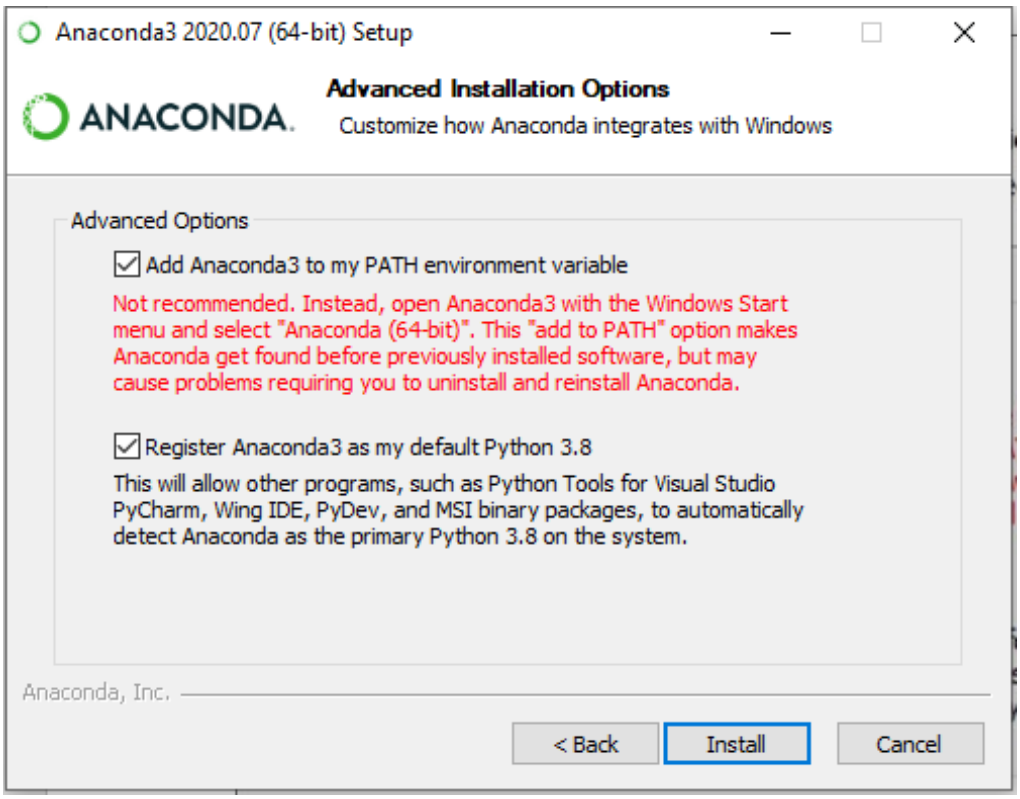
[22] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, Neural Information Processing Systems, volume 2. Morgan Kaufman, 1990.

Installation de l’Environnement de développement logiciel sous Windows 10 (64 bits)

Installation de l'Environnement

A.1 Python Environment Setup with Anaconda Python

1. Rendez-vous sur la page de téléchargements Anaconda https://www.anaconda.com/ downloadset obtenez la version Python 3.8
2. Exécuter l'installateur "Anaconda3-2020.07-Windows-x86_64.exe"



3. Vérifiez votre installation

(a) Sous windows, dans un cmd, tapper la commande suivante :

```
c :\User\hani> python --version Python
3.7
```

4. Mettre à jour Anaconda de base

(a) Sous windows, ouvrir Anaconda Prompt

(b) Dans un Anaconda Prompt, taper les commandes suivante :

```
(base) C :\Users\hani>conda update conda
(base) C :\Users\hani>conda update anaconda
(base) C :\Users\hani>conda update python
(base) C :\Users\hani>conda update --all
```

Cela devrait mettre à jour toute votre base Anaconda jusqu'aux derniers paquets. 5. Créer un Python "environnement virtuel0"

A.2 Install Opencv, Numpy et Matplotlib

Dans un Anaconda Prompt, taper les commandes suivantes :

```
(app_final) C :\Users\hani>conda install -c michael_wildopencv-
contrib
(app_final) C :\Users\hani>conda install -c conda-forge matplotlib
(app_final) C :\Users\hani>conda install -c anaconda numpy
```

A.3 Install Keras et Tensorflow

1. Ouvrir Anaconda Prompt avec les droits d'administration.
2. Dans un Anaconda Prompt, taper les commandes suivante :

```
(base) C :\Users\hani>conda activate app_final
(app_final) C :\Users\hani>conda install keras-gpu
```

(a) Install Tensorflow official :

```
(app_final) C :\Users\hani>conda install tensorflow-gpu
```

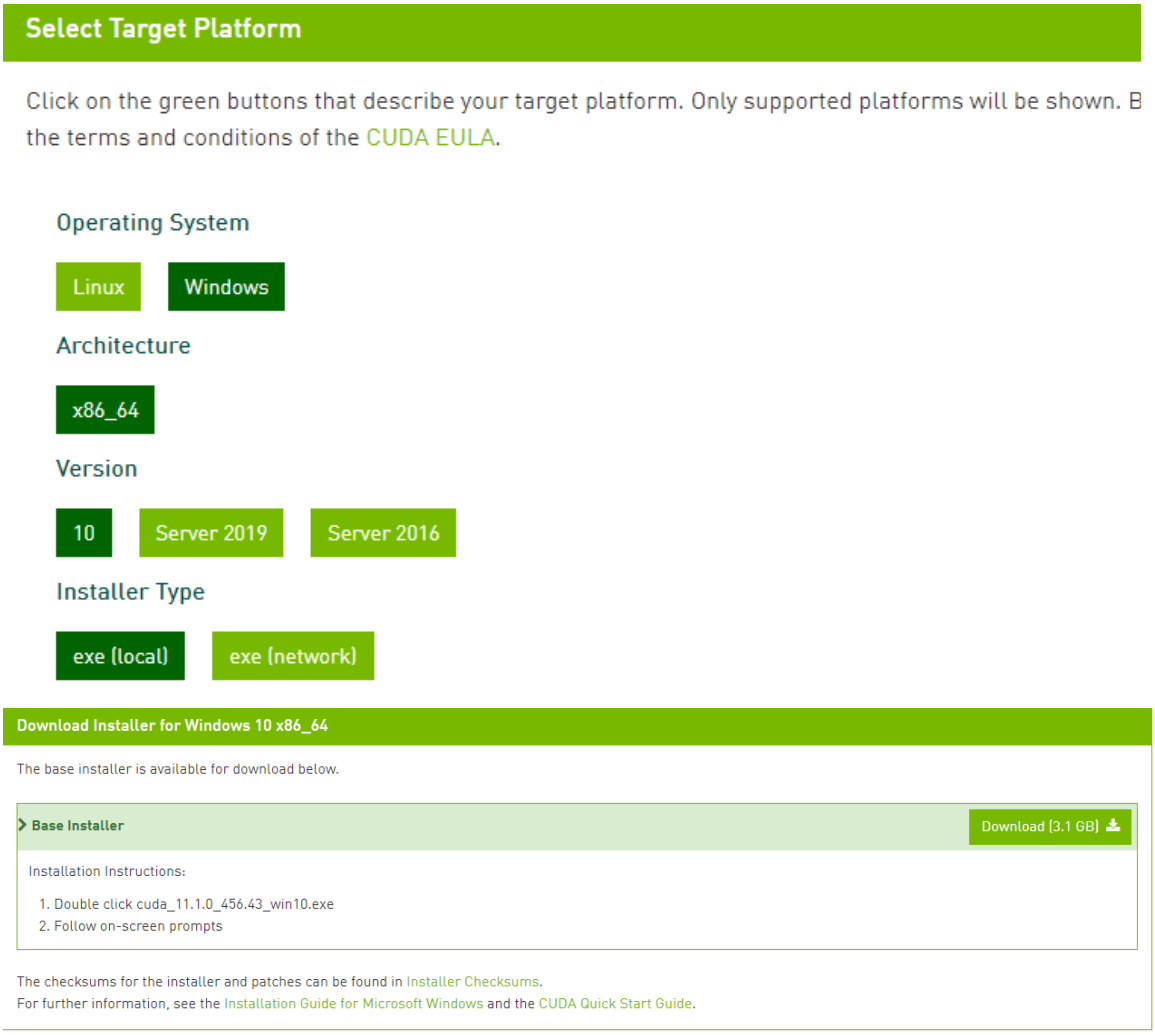
(b) Install Tensorflow pour Windows 10:

```
(app_final) C :\Users\hani>conda install -c aaronzstensorflow-gpu
(app_finql) C :\Users\hani>conda install -c anaconda cudatoolkit
(app_final) C :\Users\hani>conda install -c anaconda cudnn
```

A.4 Install CUDA et cuDNN

Ces étapes d'installation CUDA et cuDNN sont vaguement basées sur le guide d'installation Nvidia CUDA.

1. Telecharger CUDA sur le site Web de Nvidiahttps://developer.nvidia.com/ cuda-downloads.



2. Installer CUDA.
3. TelechargercuDNN sur le site Web de Nvidia Programme des développeurs https: //developer.nvidia.com/cudnn.