Conception et implémentation

3.1 Introduction :

Dans ce chapitre, nous allons procéder à la conception de notre application. Nous allons présenter aussi la mise en oeuvre de notre application en utilisant le langage Python. En commençant tout d'abord par une présentation du langage de programmation choisi. Ensuite nous décrivons les captures d'écran de l'exécution de notre application.

3.2 Configuration matérielle

3.2.1 Configuration matérielle distante (Google Colab)

Google Collaboratory ou Colab [38] : un outil Google simple et gratuit pour nous initier aux réseaux profonds ou collaborer avec nos collègues sur des projets en science des données. Colab permet d'améliorer nos compétences de codage en langage de programmation Python, de développer des applications en réseaux profonds, en utilisant des bibliothèques populaires telles que Keras, TensorFlow sans installation, ainsi que l'utilisation d'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 à votre compte Google Drive.

3.3 Environnement de développement logiciel

3.3.1 Python

Python [39]est un langage de programmation de haut niveau interprété et orienté objet.
Il est très sollicité par une large communauté de développeurs et de programmeurs. Python est un langage simple et facile à apprendre. Les bibliothèques de python sont disponibles pour la majorité des plateformes et peuvent être redistribuées gratuitement.

3.3.2 Tensorflow

TensorFlow [40] est une bibliothèque de logiciels open source publiée en 2015, par Google pour faciliter la conception, la construction et la formation de modèles d'apprentissage en profondeur pour les développeurs. TensorFlow était prévu à l'origine comme une bibliothèque interne que les développeurs de Google utiliseraient pour construire des modèles en interne. A un niveau élevé, TensorFlow est une bibliothèque Python qui permet aux utilisateurs d'exprimer un calcul arbitraire sous forme de graphique de flux de données. Les noeuds de ce graphique représentent des opérations mathématiques, tandis que les arêtes représentent les données communiquées d'un noeud à un autre. Les données de TensorFlow sont représentées sous la forme de tenseurs, qui sont des tableaux multidimensionnels. Bien que ce cadre pour la réflexion sur le calcul soit précieux dans de nombreux domaines différents, TensorFlow est principalement utilisé pour l'apprentissage profond dans la pratique et la recherche.

3.3.3 Keras

Karas [42] est un Framework de plus haut niveau, il est écrit et entretenu par Francis Chollet et un autre membre de l'équipe Google Brain pour l'utiliser avec python, il aide à créer un modèle avec paramètre prédéfini (fonction de coût, optimiseur, fonctions d'activation ...).

3.4 Dataset

Nous avons utilisé la base d'images Pascal VOC (Visual Object Classe) version 2012, elle contient 11530 images couleur de différentes dimensions pour l'apprentissage distribués sur 20 classes. Dans notre projet nous avons utilisé 2000 images qui ont été redimensionnement avant de les soumettre au réseau.
cette base Contient 20 class : avion, vélo, oiseau, bateau, bouteille, autobus, voiture, chat, chaise, vache, table à manger, chien, cheval, moto, la personne, plante en pot, mouton, canapé, train, TV monitor.

3.4.1 Découpage de la base d'images

La base des images est découpée en trois parties :

— Les images d'apprentissage/entraînement (training set), qui vont servir à apprendre le réseau. Cela représente 1800 images ;

— Les images de test (test set), qui servent à évaluer les progrès de notre modèle. Elles présentent 100 images.

— Les images de validation (validation set), qui présentent 100 images.

3.4.2 Fonction de redimensionnement

Les images dapprentissage ont des tailles différentes, par conséquent, elles doivent étre redimensionnées avant d'étre utilisées comme entrée du modèle. Tandis que notre système requiert une dimensionnalité d'entrée constante, nous avons sous-échantillonné toutes les images à une résolution fixe de 320 X 320.

3.5 Présentation de l'application

Nous allons maintenant présenter les résultats obtenus grâce à des expériences réalisées par l'application de notre approche de supprimer le filigranne on utilise le deep learning .

3.5.1 Auto-encodeur (AE)

définition : Un auto-encodeur est un réseau de neurones conçu pour l'apprentissage d'une représentation d'un ensemble de données, dans le but de réduire la dimension de cet ensemble. Il permet d'extraire des caractéristiques pertinentes à l'entrée dont le but est de les reconstruire de façon non supervisée. dans notre application on utilise L'auto-encodeur convolutif comme un de pré-entrainement. Les paramètres d'autoencodeur sont initialisés de manière aléatoire.
objectif : L'intérêt de l'utilisation des auto-encodeurs est d'apprendre une représentation d'un ensemble des données, mais dans notre projet notre but est d'améliorer les résultats de l'auto encodeur débruteur. Nous avons entrainé un auto-encodeur et l'utilisé pour initialiser les valeurs des poids d'un auto-encodeur débruteur avec la même architecture.la figure3.1 présente l'architecture de notre réseau.

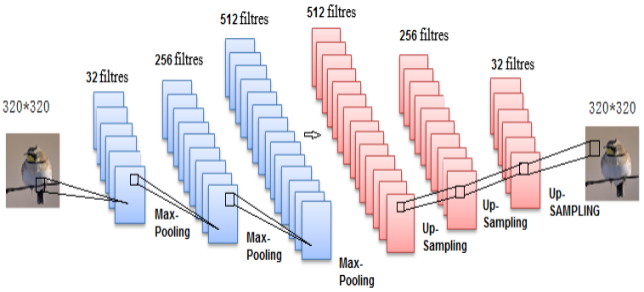

FIGURE 3.1 – L'architecture de notre Auto-encodeur (AE)

3.5.2 Combinaison auto-encodeur débruiteur

L'auto-encodeur débruiteur est une autre variante d'auto-encodeur, Le DAE (Denoising Autoencoder) est un enregistreur qui reçoit des données bruitées en entrée et qui est entraîné à prédire les données originales non bruitées en sortie.

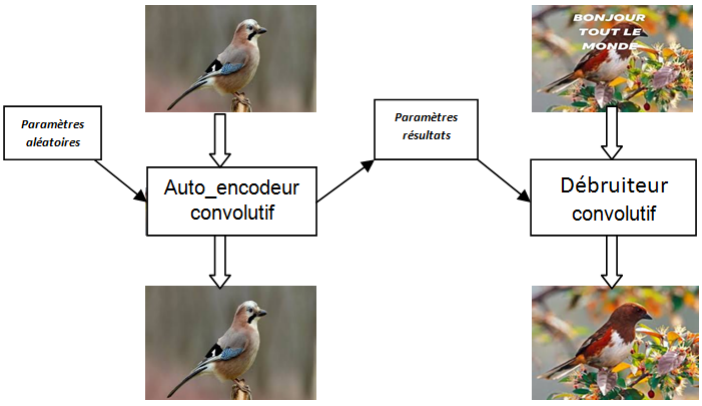

FIGURE 3.2 – Combinaison auto-encodeur débruiteur

3.5.3 Configuration du CNN

Nous allons entrainer un réseau de neurones convolutif. Le réseau possède 13523203 paramètres entrainables et il consiste en 7 couches convolutives. Une couche de Max Pooling est placée après chaque couche

convolutive d'encodage et une couche de Up- Sampling entre chaque deux couches convolutives de décodage. Pour un apprentissage plus rapide, ReLu est la fonction d'activation. La figure3.3 montre la structure de notre CNN.

```
input_img = Input(shape=(320, 320, 3))
x = Conv2D(32, (5,5), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(256, (5, 5), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(512, (5, 5), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(512, (5, 5), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(256, (5, 5), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)

x = Conv2D(32, (5, 5), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(3, (5, 5), activation='sigmoid', padding='same')(x)
```

FIGURE 3.3 – Composants de notre réseau de neurones

D'après la figure 3.3, notre modèle CNN comprend les composants suivants : **resume=*,,*

— **La couche d'entrée :** sa dimension 320*320*3, le nombre 3 représente les couleurs Rouge, Vert et Bleu.

— **La couche de convolution :** dans cette couche la taille du filtre est fixée à 5*5.

  — Le nombre de filtre où la profondeur a été fixée à 32, 256, 512 successivement pour les couches d'encodage, est l'inverse pour les couches de décodage.

  — La fonction d'activation ReLU : force le réseau à retourner des valeurs positives. Tout nombre inférieur à 0 est converti en 0, cette technique est appliquée pour toutes les couches, sauf la dernière couche qui utilise la fonction sigmoid.

— **La couche de Pooling :**nous avons utilisé un max pooling de taille 2x2 pour réduire la taille des paramètres de l'image.

— **La couche de Up-Sampling :** est la couche inverse de pooling qui permet d'inverser le pooling ou de reconstruire l'image. La taille utilisée est 2x2.

— Le pas :à été fixé à 1 dans les couches de Pooling et à 2 dans la couche de Up- Sampling.

— Dans notre modèle nous n'avons pas utilisé une couche entièrement connectée.
Les hyperparamètres sont fixés pour toutes les expériences comme suit :

— La taille de mini-batch : 16.

— La fonction de coût utilisé : MSE.

— Toutes les couches convolutives utilisent la fonction ReLu sauf la couche de sortie utilise la fonction Sigmoïde.

— L'optimiseur utilisé est adam.

— La taille des filtres (5x5), facteur de sous-échantillonnage (2x2).

3.5.4 les méthodes d'évaluation :

Afin d'évaluer la qualité des image construire par notre auto-encodeur débruiteur et pour évaluer l'efficacité de notre débruiteur(Efficacité de la suppression du filigrane) on utilise le SSIM (mesurer la similarité de structure entre deux image)et le CW-SSIM.

Couche	Paramétre
Entrée	image RVB 32X32 pixl
Conv2D(ReLu)	Profondeur : 32 Fenêtre : 5X5 Pas : 1
Couche de Pooling	Fenêtre : 2X2 : Pas : 1
Conv2D(ReLu)	Profondeur : 256 Fenêtre : 5X5 Pas : 1
Couche de Pooling	Fenêtre : 2X2 : Pas : 1
Conv2D(ReLu)	Profondeur : 512 Fenêtre : 5X5 Pas : 1
Couche de Pooling	Fenêtre : 2X2 : Pas : 1
Conv2D(ReLu)	Profondeur : 512 Fenêtre : 5X5 Pas : 1
Couche de UpSumpling	Fenêtre : 2X2 : Pas : 2
Conv2D(ReLu)	Profondeur : 256 Fenêtre : 5X5 Pas : 1
Couche de UpSumpling	Fenêtre : 2X2 : Pas : 2
Conv2D(ReLu)	Profondeur : 32 Fenêtre : 5X5 Pas : 1
Couche de UpSumpling	Fenêtre : 2X2 : Pas : 2
Conv2D(sigmoid)	Profondeur : 32 Fenêtre : 5X5 Pas : 1

TABLE 3.1 – Configuration de notre CNN

– SSIM1 :le SSIM entre les images bruitées et les image reconstruites par le réseau.

– SSIM2 :le SSIM entre les images originales et les image reconstruites par le réseau.

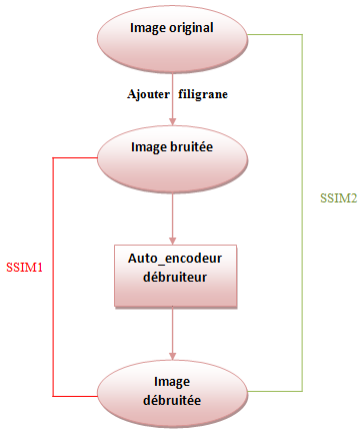

FIGURE 3.4 – Plan d'évaluation

3.6 Résultats d'application

3.6.1 Les résultats obtenus par l'auto-encodeur convolutif

Les résultats obtenus par l'auto-encodeur convolutif sont très proches aux images originales,les résultats sont visualisables dans la figure 3.5.

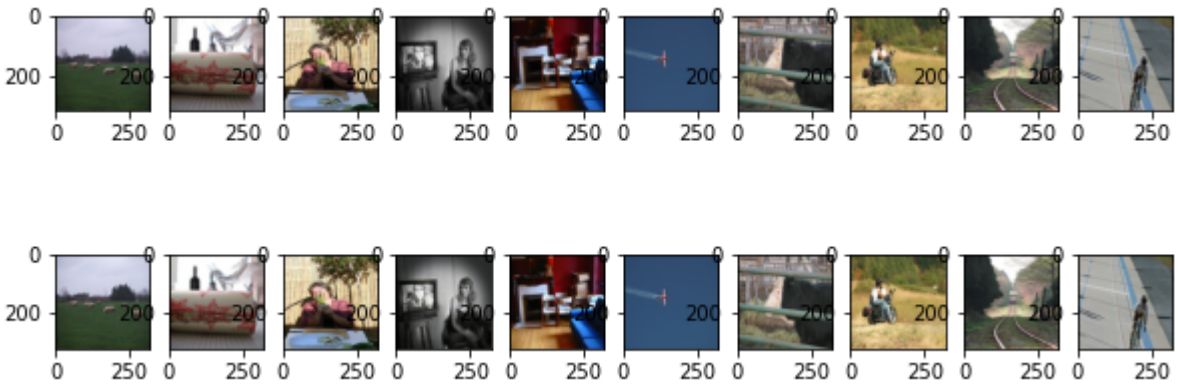

FIGURE 3.5 – Image résultat d'auto-encodeur convolutif.

les graphes de la Figure 3.6 montrent que la précision d'apprentissage et de validation augmente avec le nombre des époques. En même temps, l'erreur d'apprentissage et de validation diminue avec le nombre des époques. Donc à chaque époque notre modèle apprend à reconnaitre plus d'informations et de manière plus rapide.

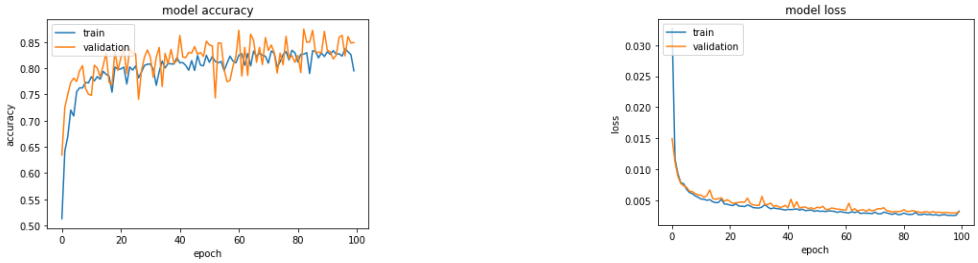

FIGURE 3.6 – Graphes de Précision et d'Erreur de l'auto-encodeur convolutif.

La figure 3.7 montre les valeurs du SSIM entre les images originales et les images reconstruites par notre réseau de neurones .Notons que certaines images ayant un SSIM proche de 1. Elles sont très semblables aux images originales. Ce sont des images simples avec un fond uni et qui contiennent moins de détails.

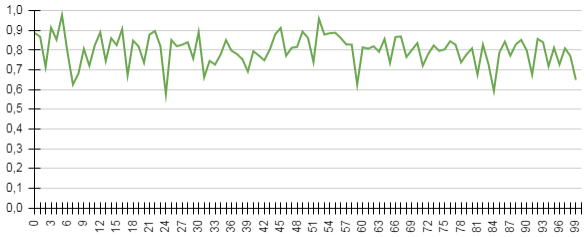

FIGURE 3.7 – Graphe de SSIM entre les images originales et les images obtenues par l'autoencodeur convolutif.

3.6.2 les résultats obtenus par l'autoencodeur débruiteur convolutif

Le processus d'apprentissage dans l'auto-encodeur débruiteur passe par deux étapes .La première consiste à entraîner le CNN pour qu'il puisse effectuer le rôle d'un auto-encodeur convolutif. Le but de cette partie est d'entraîner le réseau à reproduire l'image à partir de sa version originale. Les paramètres du réseau résultat seront utilisés comme paramètres initiaux de la deuxième partie d'apprentissage. Cette partie permet d'entraîner le réseau à reproduire les images originales à partir de leurs versions filigranées. Les images de la base d'apprentissage de cette étape sont crées en ajoutant des filigranes aux images originales.

Le filigrane "test figrane"

Notre CNN construit une image sans filigrane. Les résultats sont présentés dans la figure 3.8.

FIGURE 3.8 – Images résultats d'auto-encodeur convolutif débruiteur

D'après l'image 3.8 ,nous notons que les images résultantes sont des images sans filigrane et proches des images originales.Pour prouver cette observation, les images doit êtres évaluées par SSIM(évaluation objectif).

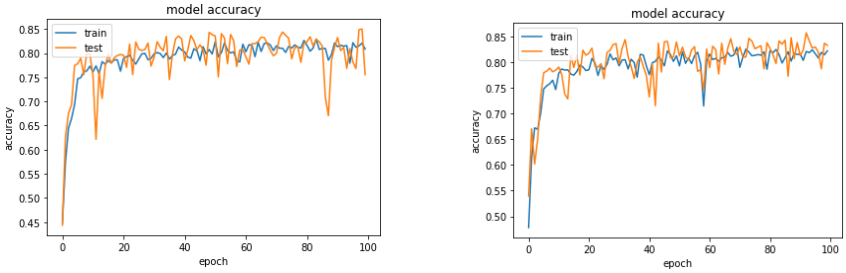

FIGURE 3.9 – Graphe de la Précision pour l'auto-encodeur convolutif débruiteur

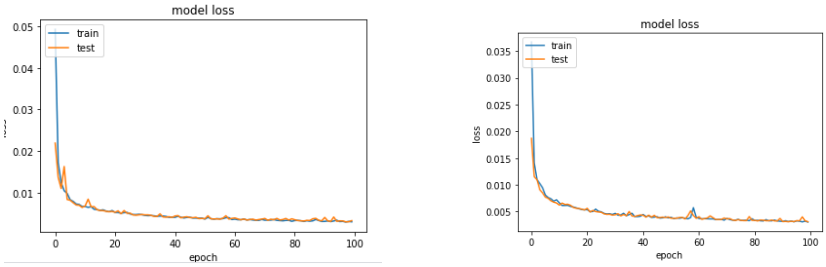

FIGURE 3.10 – Graphe de l'erreur pour l'auto-encodeur convolutif débruiteur

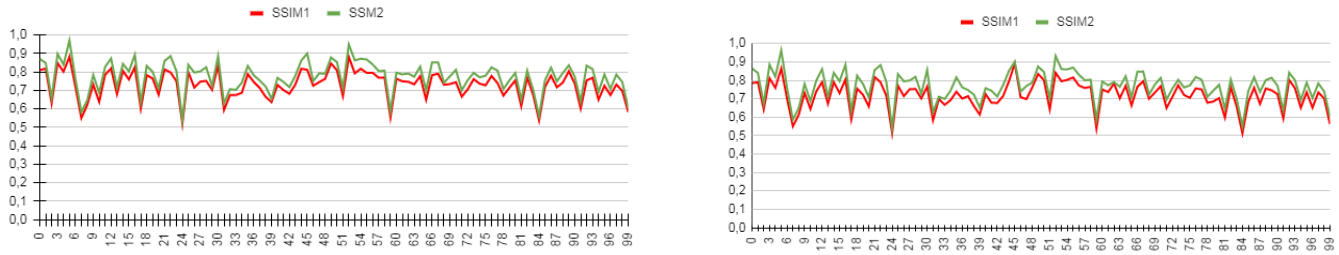

FIGURE 3.11 – SSIM

filigranne : text court

Les résultats sont présentés dans la figure 3.12.

FIGURE 3.12 – Images résultats d'auto-encodeur débruiteur (text court)

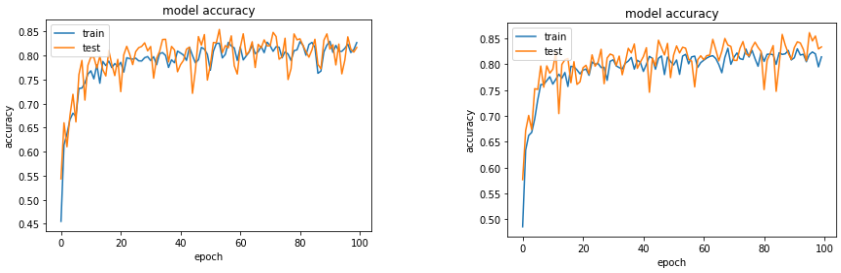

FIGURE 3.13 – Graphe de la Précision pour l'auto-encodeur débruiteur

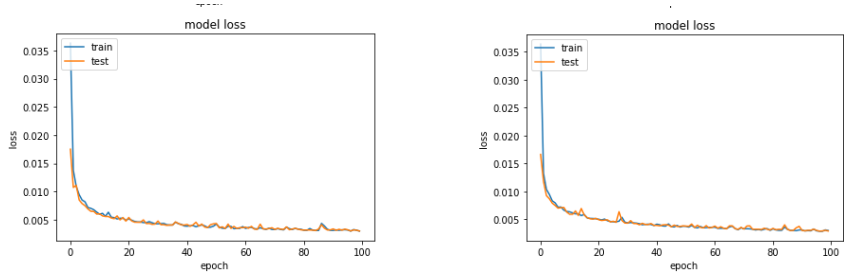

FIGURE 3.14 – Graphe de l'erreur pour l'auto-encodeur débruiteur

D'après les graphes de SSIM résultats présenter dans la figure 3.15 ,On remarque qu'ils sont pareils, donc les images résultats de ces deux traitements sont de la même qualité.
On remarque Aussi ,que notre application capable de reconstruire une image de la même qualité que limage originale (SSIM=1 dans le graphe)Sans apprendre sur l'image d'origine (image de test).

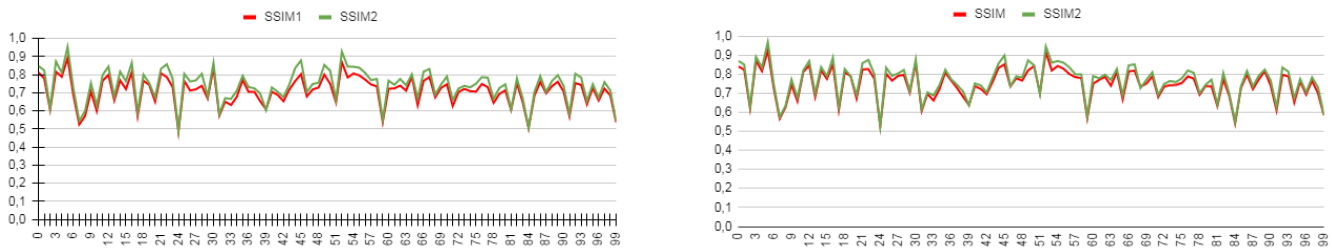

FIGURE 3.15 – Graphe de SSIM

Le filigrane : Texte aléatoire

Notre CNN reconstruit une image sans filigrane. Les résultats sont présentés dans la figure3.16.

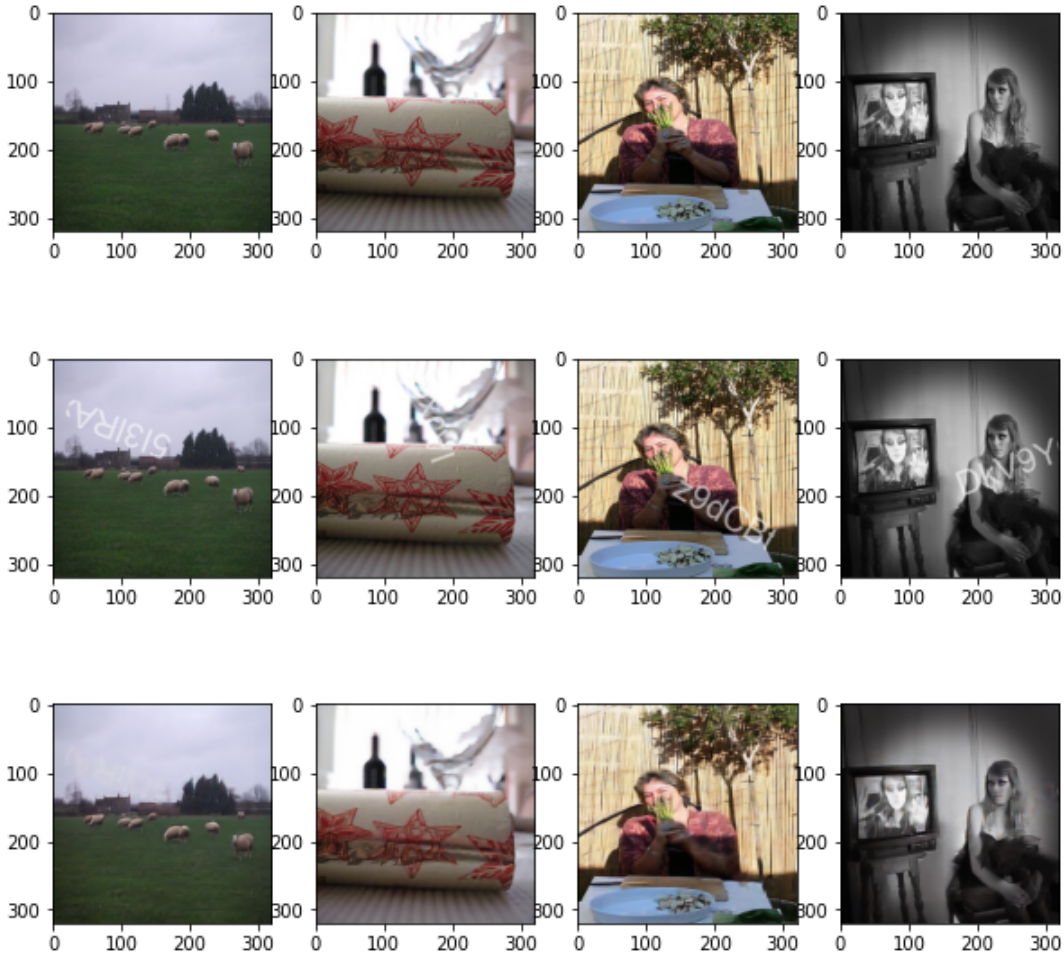

FIGURE 3.16 – Images résultats d'auto-encodeur débruiteur (texte aléatoire).

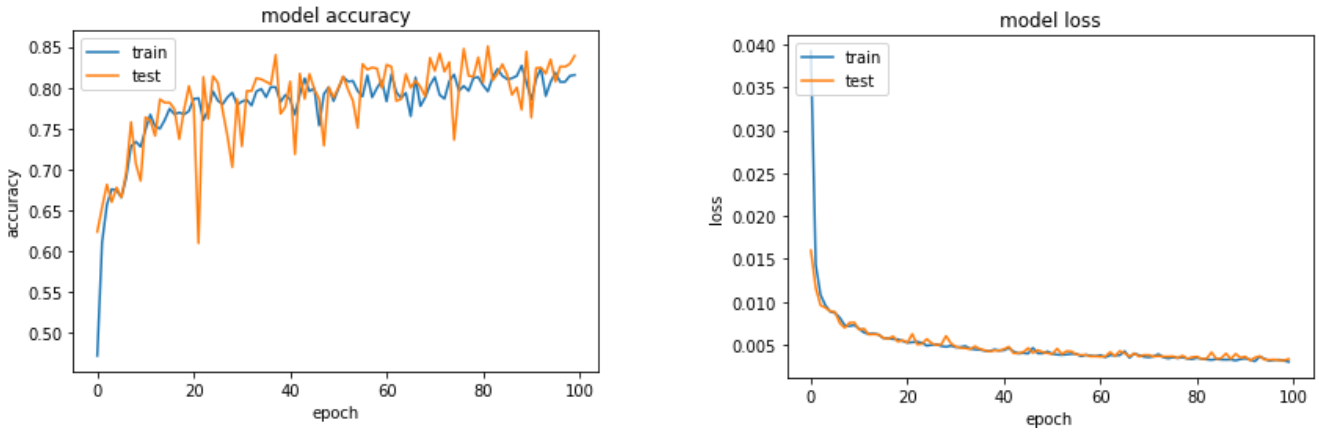

FIGURE 3.17 – Graphe de la Précision et de l'erreur pour l'auto-encodeur débruiteur

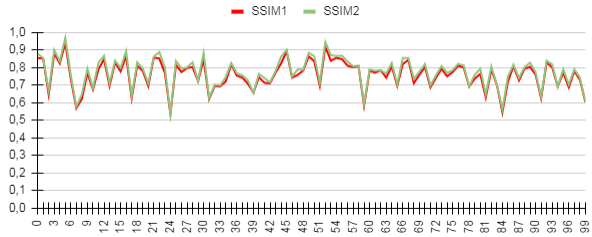

FIGURE 3.18 – SSIM des images résultats (texte aléatoire).

3.6.3 Évaluation par CW-SSIM :

Le filigrane :texte aléatoire :

FIGURE 3.19 – Images résultats .

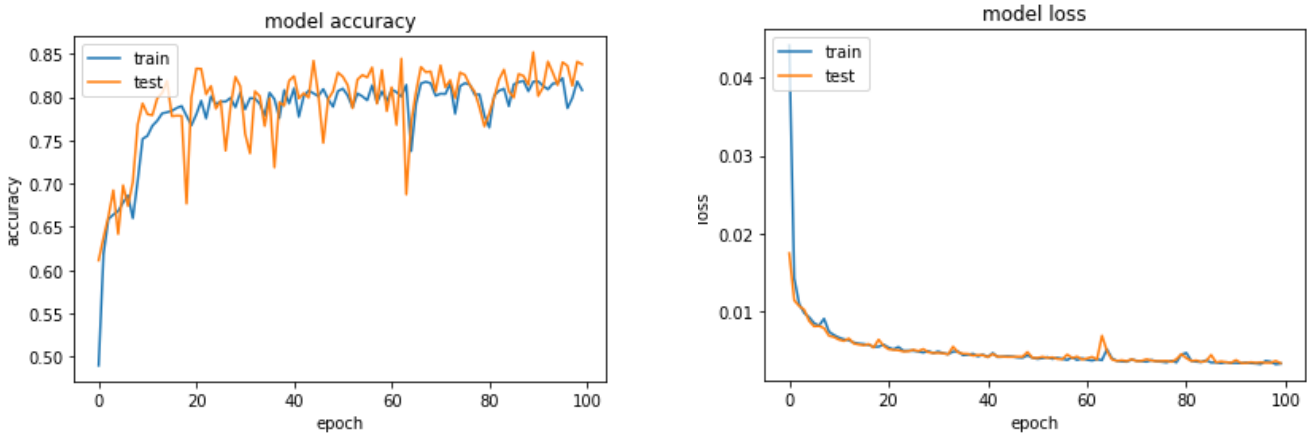

FIGURE 3.20 – Graphe de la Précision et de l'erreur pour le débruiteur(cwssim)

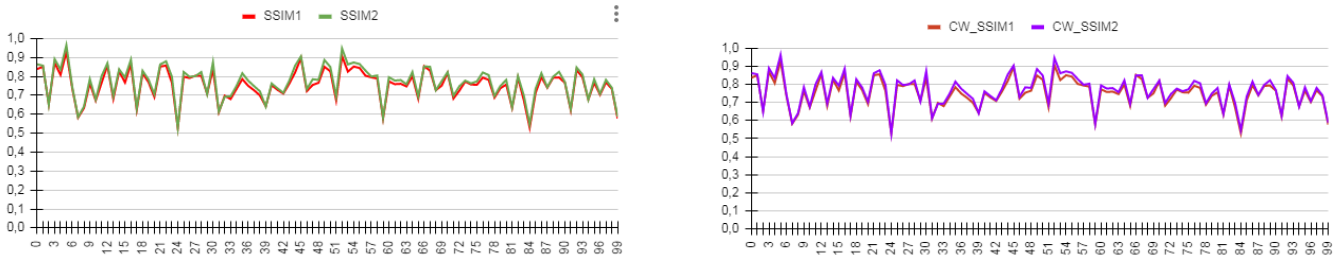

FIGURE 3.21 – Graphe de SSIM à gauche et de CW-SSIM à droit

D'après la figure 3.21 les résultats de SSIM et de CW-SSIM sont les même.

3.7 Les résultat obtenus par un débruiteur convolutif

Le filigrane :texte aléatoire :

FIGURE 3.22 – Images résultats débruiteur (texte aléatoire).

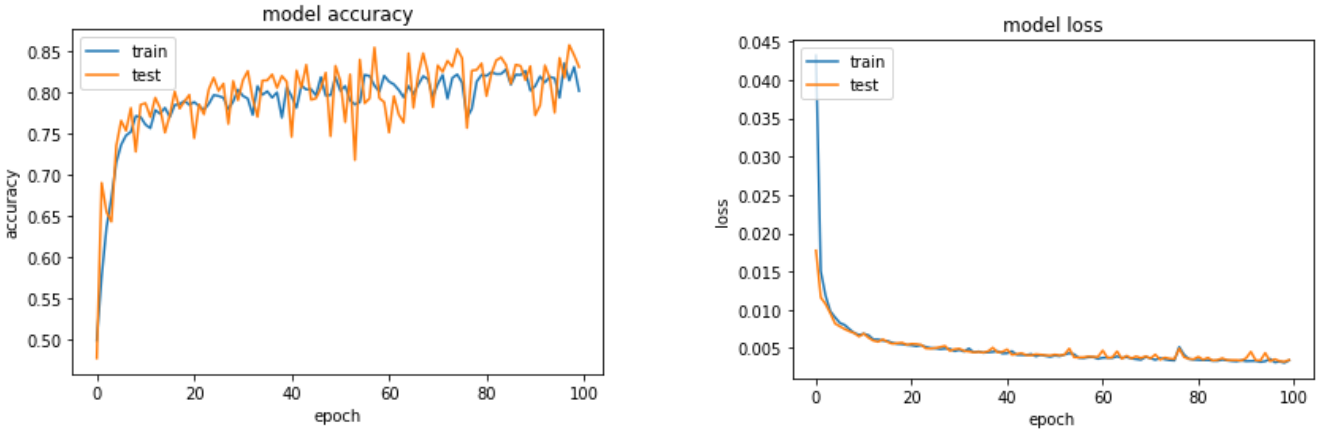

FIGURE 3.23 – Graphe de la Précision et de l'erreur pour débruiteur

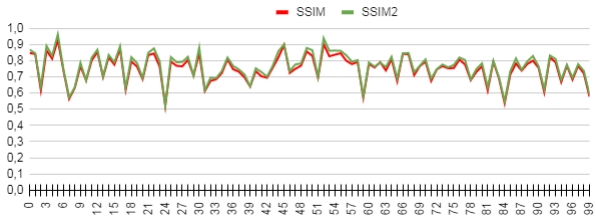

FIGURE 3.24 – SSIM des images résultats d'un débruiteur(texte aléatoire).

3.7.1 Pearson coefficient de corrélation(PCC)

Le PCC est la covariance entre le score SSIM1 et SSIM2, rapport2e au produit de leurs écart-types :

$$\frac{Covariance(SSIM1, SSIM2)}{\sigma_{SSIM1} * \sigma_{SSIM2}} \tag{3.1}$$

Il détermine la corrélation des SSIMs, une grande valeur de PCC (proche à 1) indique une bonne précision de prédiction. Le tableau ci-dessous présente les valeurs de PCC calculées :

filigrane	théme et taille	Auto-encodeur débruiteur
"ILM"	Arial /20	0.9930375173
"ILM"	Arial /50	0.9895909558
"testfigrane"	Arial /70(noir)	0,9779749979
"testfigrane"	Arial /70	0,9866073836
text aléatoir	Arial	0,9942491089
text aléatoir	Times New Roman	0,8198216837

TABLE 3.2 – Tableau des PCC calculés

On remarque qu'on a des bonnes résultats, les valeurs de PCC sont élevées (proches à 1 et elles se diminuent avec l'augmentation de bruit), ce qui indique qu'il existe une corréélation positive entre les SSIM1 et les SSIM2.

3.8 Conclusion

Dans ce chapitre nous avons présenté une méthode pour supprimer le filigrane en utilisant les réseaux de neurones profonds (Deep Learning). Nous avons utilisé un autoencodeur convolutif combiné avec un débruiteur. Nous avons présenté dans ce chapitre les différents paramètres de codage des deux réseaux de neurones précédemment cités.
Dans la deuxième partie nous avons présenté les différents résultats obtenus en fonction du SSIM et du PCC , ces résultats ont prouvé l'efficacité de cette combinaison.

Conclusion

Le filigrane n'est pas une technologie nouvelle, mais il est très ancien, tel qu'il est apparu pour la première fois sur papier en 1282, mais avec le développement de la numérisation, il est devenu largement présent sur la plupart des images d'Internet et des appareilles photos. Ce filigrane peut-être un handicap pour qui, veut utiliser ou exploiter ces images ,en particulier avec la difficulté d'obtenir l'image originale. Cela a créé un champ de recherche sur les techniques de suppression de Filigrane parce que c'est un obstacle pour les utilisateurs de ces images, et d'autre part il affecte négativement la qualité de l'image car il peut couvrir certains détails importants dans l'image. Et comme nous nous soucions de l'image et sa qualité, nous avons traité le sujet dont nous avons discuté afin de supprimer le filigrane d'une nouvelle manière, en nous appuyant sur le dernier domaine des médias automatisés, qui est l'apprentissage en profondeur. Nous avons suivi les résultats obtenus en évaluant leur qualité par évaluation visuelle et avec l'évaluation objective (SSIM).

Dans la première partie de notre travail, nous avons commencer par un autoencodeur convulutional dans le but de récupérer une image à partir de sa version originale. Il permet d'extraire des caractéristiques pertinentes à l'entrée dont le but est de les reconstruire de façon non supervisée. Les paramètres d'autoencodeur sont initialisés de manière aléatoire.

Les résultats de notre autoencodeur convolutif implémenté a donné des résultats très satisfaisants vis à vis le SSIM et la qualité visuelle. L'objectif de ce réseau dans notre travail n'étais pas de reproduire une image à partir de sa version originale mais de préparer le réseau à reproduire une image à partir de sa version filigranée. On peut considéré cette étape comme étant un prétraitement.

Dans la deuxième partie de notre travail, nous avons combiné l'auto-encodeur par un autre réseau de neurones convolutinel (débruiteur) et il est utilisé pour initialiser les valeurs des poids d'un auto-encodeur débruteur avec la même architecture. Nous avons utilisé les paramètres résultats de l'autoencodeur comme paramètres initiaux au débruiteur. Cette combinaison a donné des bonnes qualités subjective(visuelle) et objectives (SSIM, CW-SSIM).

Perspective Selon les résultats obtenus dans les expériences effectuées dans ce travail, nous proposons comme perspectives de :

Tester l'autoencodeur débruiteur sur les images filigranées par image(filigrane image).

Bibliographie

[1] Yann LeCun. Recherche sur l'intelligence artificielle, 2016.

[2] Richard M.and David Grubin. *The Secret Life of the Brain*. Joseph Henry Press, 2001.

[3] Nikhil Buduma and Nicholas Lacascio. *Fundamentalsof Deep Learning*. June 2017.

[4] Martial Mermillod. *Réseaux de neurones biologiques et artificiels ;Vers L'émergence de systémes artificiels conscients ?*, volume 129. 2016.

[5] Djeriri Youcef. Les réseaux de neurones artificiels. September 2017.

[6] Gilbert Saporta. Une bréve histoire de l'apprentissage. 2018.

[7] Barry SMITH. L'esprit connexionniste : une étude de la psychologie de hayek. 1/1999.

[8] Kary Framling. *Modélisation et apprentissage des préférences par réseaux de neurones pour l'aide à la décision multicritére*. PhD thesis, INSA de Lyon, 1996.

[9] Marc Sauget. *Parallélisation de problémes d'apprentissage par réseaux neuronaux artificiels.Application en radiothérapie externe*. PhD thesis, Université de FrancheComté, 2007.

[10] Antoine Bordes Xavier Glorot and Yoshua Bengio. Deep sparse rectifier neural networks.in proceedings of the fourteenth international conference on artificial intelligence and statistics. page 315 â“ 323. 2011.

[11] Ilya Sutskever Alex Krizhevsky and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. page 84 â“ 90. ACM, May 2017.

[12] K.Gurney. An introduction to neural networks. 1997.

[13] Aurélie Fischer. Deux méthodes d'apprentissage non supervisé :synthése sur la méthode des centres mobiles et présentation des courbes principales. *la Société Française de Statistique*, 17 mars 2014.

[14] Alphago. `https://fr.wikipedia.org/wiki/AlphaGo`.

[15] Les réseaux de neurones convolutifs. `https://datasciencetoday.net//deep-learning/lesr{é}seaux-deneurones-convolutifs`.

[16] N. Golea. Tatouage numérique des images couleurs rgb, 2010.

[17] *Les Filigranes. Dictionnaire historique des marques du papier dès leur apparition vers 1282 jusqu'en 1600*, volume 4. G CORG-OLMS, 2 edition, 1991.

[18] *Electronic Watermark ,Digital Image Computing, Technology and Applications*. 1993.

[19] Différence entre la stéganographie et la cryptographie. `https://fr.fondoperlaterra.org/comdifference-between-steganography-and-cryptography`, 2020.

[20] J. Kilian F. T. Leighton Cox, I. J. and T. Shamoon. Secure spread spectrum watermarking for multimedia, 1997.

[21] M. Banagar Zarmehi, N. and M. A. Akhaee. Optimum decoder for an additive video watermarking with laplacian noise in h.264.

[22] M El-Hajji. La sécurité d'images par le tatouage numérique dans le domaine d'ondelettes, 2012.

[23] N. V. Dharwadkar and B. Amberker. Secure watermarking scheme for color image using intensity of pixel and lsb substitution. *Journal of Computing*, 2009.

[24] B. Chen and G. W. Wornell. Quantization index modulation : a class of provably good methods for digital watermarking and information embedding, may 2001.

[25] T. Furon and P. Bas. A new measure of watermarking security applied on qim, 2013.

[26] M. Khan and T. Shah. A copyright protection using watermarking scheme based on nonlinear permutation and its quality metrics.

[27] ITU-T Recommendation P.910. Subjective video quality assessment methods for multimedia applications. recommendations of the itu, 2008.

[28] C.CHARRIER et al. Evaluation de la qualité des images, 2012.

[29] O. Le Meur P. Le Callet A. Ninassi and D. Barba. Considering temporal variations of spatial visual distortions in video quality assessment. *IEEE Journal of Selected Topics in Signal Processing(JSTSP).*, 2009.

[30] ITU-R Recommendation BT.1683. Itu-r recommendation bt.1683. objective perceptual video quality measurement techniques for standard definition digital broadcast television in the presence of a full reference., 2011.

[31] E. R. Kandel. *Principles of Neural Science.* McGraw-Hill., 2013.

[32] Hamid R. Sheikh. Zhou Wang, Alan C. Bovik and Eero P. Simoncelli. Image quality assessment : From error visibility to structural similarity., 2004.

[33] Yang C.L Xie S.L Chen, G.H. Gradientbased structural similarity for image quality assessment.

[34] A. K. Moorthy and A. C. Bovik. A two-step framework for constructing blind image quality indices,, 2010.

[35] *Ten lectures on wavelets,society for industrial and applies 100 mathematics.* 1992.

[36] K. Ghazinour O. Alaql and C. C. Lu. Classification of image distortions based on features evaluation, 2015.

[37] Rajiv Soundararajan and Alan C. Bovik. Rred indices. Reduced reference entropic differencing for image quality assessmen, 2012.

[38] Zainab Awan Noor Al Madeed and Somaya Al Madeed. Image quality assessment a survey of recent approaches. 2011.

[39] Shibao Zheng Xiaolin Chen and Rui Zhang. Reduced reference image quality assessment based on image statistics in pixel domain. *Advances on Digital Television and Wirless Multimedia Communications*, 2012.

[40] Henri Michel. Google colab : Le guide ultime, 4/11/2019.

[41] tensorflow.

[42] keras.

Résumé

Dans cette étude nous avons utilisé la méthode d'apprentissage profond et les réseaux de neurones convolutifs. Nous avons expérimenté ces réseaux pour supprimer le filigrane de façon automatique, sans utiliser l'image originale(apprentissage non supervisé). Pour se faire, nous avons appliqué la combinaison autoencodeur-débruiteur sur un ensemble d'images filigranées. La base des images utilisée est PASCAL VOC dont nous avons testées 2000 images (1800 images pour l'entrainement, 100 images pour la validation et 100 pour le teste),nous avons déformé ces image en ajoutant un text (filigrane). Nous avons procédé à une évaluation des images produites en les comparant aux images originales et les images portant un filigrane, en utilisant la métrique d'évaluation objective, basée sur l'approche structurelle (SSIM).Les résultats étaient, en grande partie, assez satisfaisants. Les réseaux de neurones convolutifs ont pu rétablir et restituer l ?authenticité et la qualité des images à 82% de celle des images originales.

Mots-clés : Apprentissage profond ,Réseaux de neurones convolutifs , Filigrane , Auto-encodeur débruiteur , Evaluation objective , Qualité de l'image .

Abstract

In this study we used the deep learning method and convoluted neural networks. We experimented these networks to remove the watermark automatically, without using the original image(unsupervised learning).To do this, we applied the autoencoder-denoiser combination on a set of watermarked images. The basis of the images used is PASCAL VOC of which we tested 2000 images (1800 images for training, 100 images for validation and 100 for testing). We evaluated the images produced by comparing them to the original images and the watermark images, using the objective assessment metric, based on the structural approach (SSIM). The results were, to a large extent, quite satisfactory. Convoluted neural networks were able to restore the authenticity and quality of the images to 82% of that of the original images.

Keywords : Deep learning ,convoluted neural networks ,watermark,autoencoder-denoiser, Objective evaluation, Image quality.