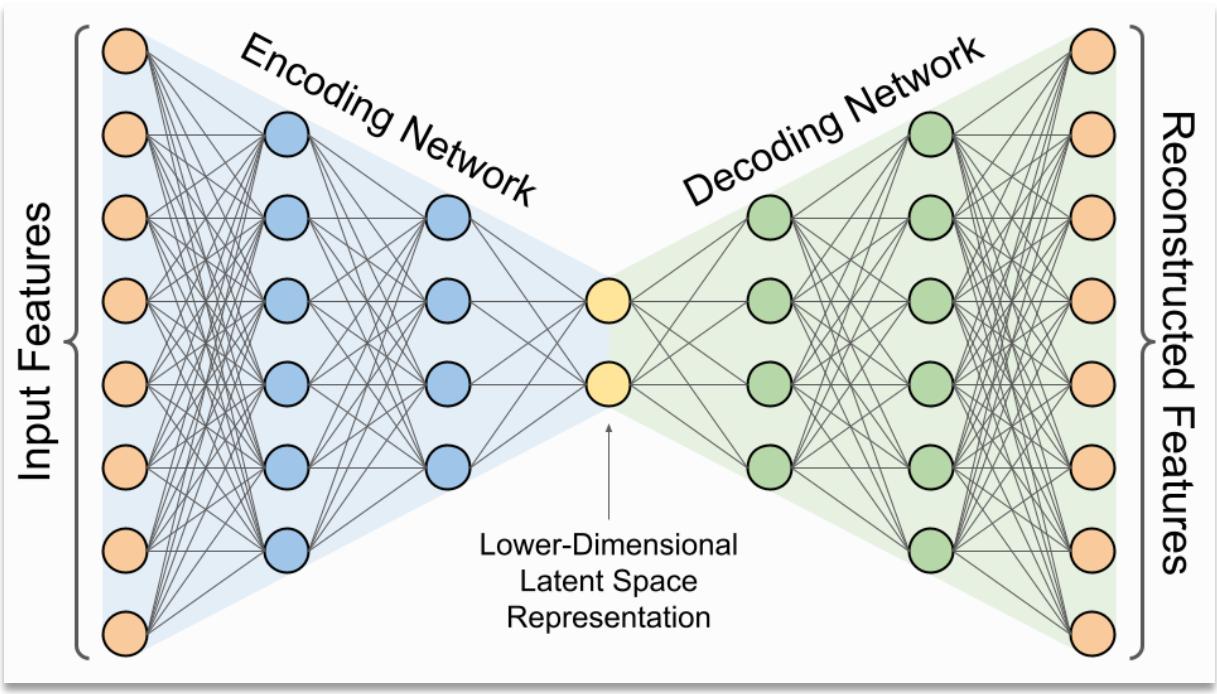
Analyse de données : TP3

L’auto-encodeur avec Keras



AL NATOUR Mazen et HERVOUET Léo

AD 2023

Table des matières

Introduction 3
1 – Chargement des données Fashion-MNIST 4
2 – Auto-encodeur avec une seule couche cachée 5
2.1 – Définition des modèles de l'encodeur et du décodeur 5
2.2 – Apprentissage et évaluation de la performance du modèle 6
2.3 – Analyse des résultats de l'auto-encodage avec le modèle appris 7
2.4 – Paramétrage 9
3 – Auto-encodeur profond 10
3.1 Définition du modèle 10
3.3 Analyse des résultats de l'auto-encodage avec le modèle appris 11
Conclusion 12

Introduction

Dans ce TP, nous allons voir comment réaliser des auto-encodeurs avec comme objectif de compresser des images issues du dataset 'Fashion-MNIST' tout en préservant leur caractère visuel. Nous commencerons avec un auto-encodeur à une seule couche cachée. Nous allons ensuite voir comment réaliser un auto-encodeur plus complexe avec plusieurs couches cachées, en anticipant une amélioration potentielle de la performance par rapport à notre modèle initial. Nous analyserons les résultats obtenus, évaluant la similarité entre les images originales et celles générées par nos auto-encodeurs. Des métriques comme l'indice de similarité structurelle (SSIM) et l'erreur quadratique moyenne (MSE) jouerons un rôle crucial dans cette évaluation. Enfin, nous allons explorer les paramètres clés, comme le nombre d'époques et de neurones, pour comprendre leur impact sur la performance des modèles. Ces ajustements permettent une utilisation optimale des auto-encodeurs.

1 – Chargement des données Fashion-MNIST

Nous préparons les données de Fashion-MNIST en les chargent avec ‘keras.datasets.fashion_mnist.load_data()’.

Ensuite nous vérifions si les données reçues sont bien conformes à ce que nous attendons grâce à des assertions.

Dans ‘x_train’ nous devons avoir 60 000 images de 28 sur 28 pixels et dans ‘x_test’ 10 000 images aussi de 28 sur 28 pixels.

‘x_train’ va nous permettre de faire l’apprentissage et ‘x_test’ va nous permettre de faire la correction de cette apprentissage.

Pour finir, nous normalisons les données des images pour passer à dans images en une dimension de longueur 784 au lieu d’images 2D de 28 sur 28 pixels.

Figure 1: Les 5 premières images de l'ensemble de données Fashion-MNIST

Les cinq premières images de l'ensemble de données Fashion-MNIST sont présentées dans le cadre. Chaque image, offre une représentation pixelisée à basse résolution d'articles vestimentaires distincts. Dans chaque image les détails les plus complexes sont principalement représenté en tons verts. Le bleu pour possiblement décrire des détails de conception ou des ombres.

2 – Auto-encodeur avec une seule couche cachée

2.1 – Définition des modèles de l'encodeur et du décodeur

Résultat de la fonction **.summary()** sur l'autoencodeur :

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 784)]	0
dense (Dense)	(None, 32)	25120
dense_1 (Dense)	(None, 784)	25872

Total params: 50992 (199.19 KB)

Trainable params: 50992 (199.19 KB)

Non-trainable params: 0 (0.00 Byte)

Dans le résultat nous avons bien notre couche d'entrée nommée 'input_1', qui prend en entrée un vecteur de 784 données (correspondant à nos images de 28x28 pixels) et qui représente notre encodeur. Ensuite nous avons la couche du milieu 'dense', qui est une couche cachée et prend en entrée un vecteur de 32 données et qui a 25120 paramètres modifiables. Et pour finir la couche 'dense_1' qui est notre couche de sortie qui est le décodeur avec en sortie un vecteur de 784 données et 25872 paramètres modifiables. Le calcul des paramètres modifiables est fait simplement : taille du vecteur * nombre de neurones + nombre de neurones.

Finalement ces trois couches représentent notre auto-encodeur.

2.2 – Apprentissage et évaluation de la performance du modèle

Voici notre graphique de pertes pour déterminer la performance de notre modèle :

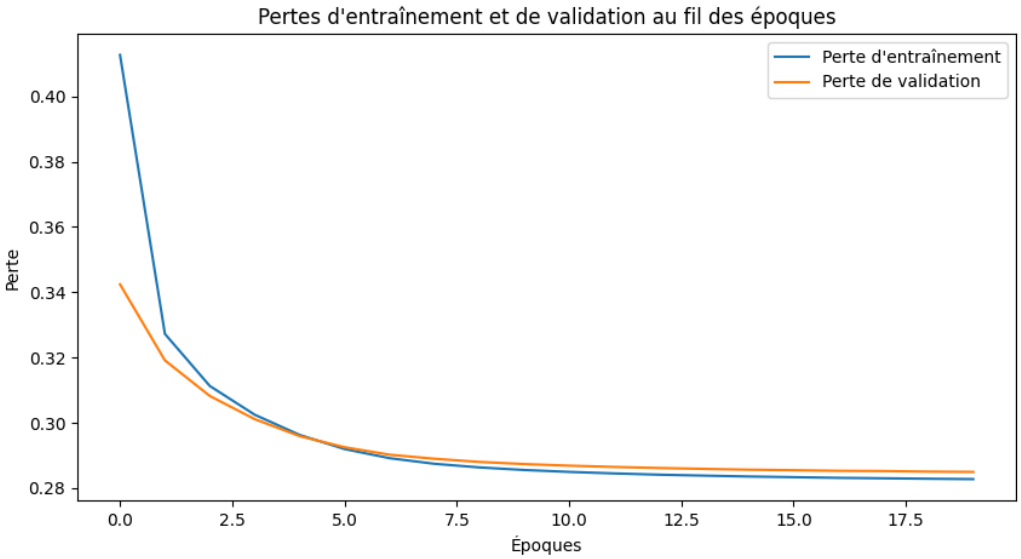


Figure 2 : Graphique des pertes

D'après le graphique, on peut tirer plusieurs observations :

On remarque une diminution des pertes, à la fois pour l'entraînement et la validation, indiquant ainsi une efficacité dans l'apprentissage du modèle à partir des données.

La stabilisation des pertes devient apparente après environ 7,5 époques, avec de légères fluctuations. Cela suggère que le modèle a peut-être atteint un point où des améliorations significatives de sa performance avec les données d'entraînement actuelles deviennent moins probables.

La perte d'entraînement demeure constamment inférieure à la perte de validation, conforme aux attentes en apprentissage supervisé. Bien que l'écart entre les deux ne soit pas très grand, cela reste un signe positif.

Le modèle semble bien performant et s'ajuster de manière appropriée aux données d'entraînement et de validation. Il pourrait interrompre l'entraînement après environ 7,5 époques, car les pertes semblent se stabiliser à ce stade.

2.3 – Analyse des résultats de l'auto-encodage avec le modèle appris

Voici nos résultats après le décodage des images :

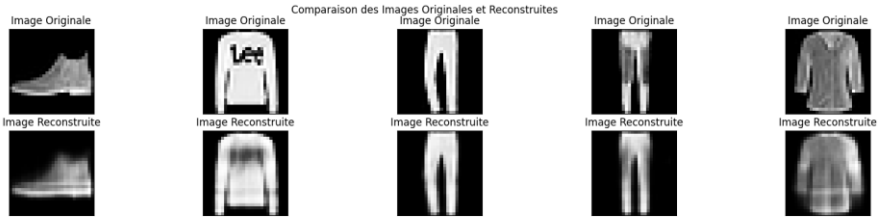


Figure 3 : Images de tests (en haut) et images après le décodage (en bas)

Nous pouvons voir que nos images, après le décodage, restent au même endroit que les images d'origines. Seulement nous pouvons voir que les petits détailles ne reste pas, comme la semelle de la chaussure ou le texte sur le pull-over.

Voici les digrammes en boîte pour l'SSIM et l'MSE :

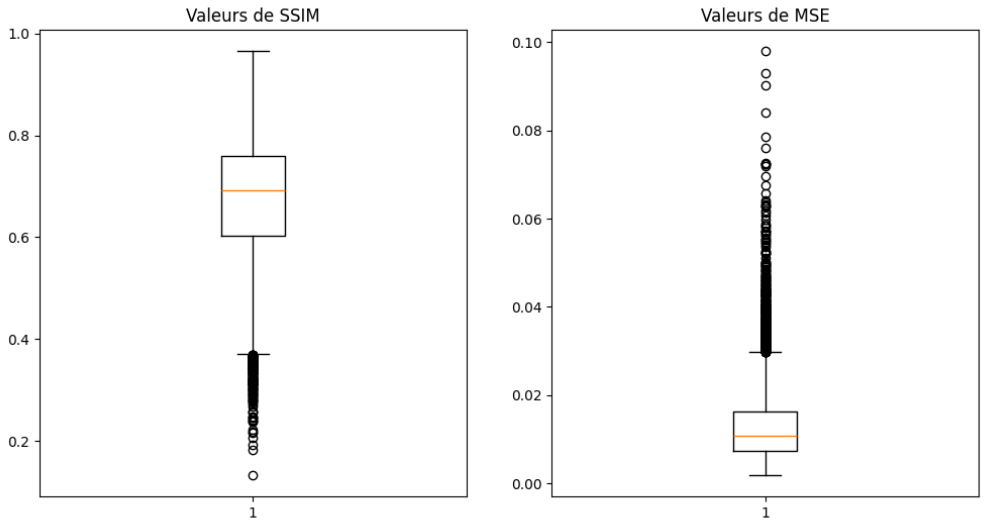


Figure 4 : Diagramme en boîte de l'SSIM et de l'MSE

Indice de Similarité Structurelle (SSIM) :

Le SSIM est une mesure de la similarité entre deux images, variant de -1 à 1, où 1 indique une identité parfaite. La boîte illustre l'écart interquartile (IQ), contenant la moitié médiane des valeurs de SSIM. Le bord supérieur représente le troisième quartile (Q3), le bord inférieur indique le premier quartile (Q1), et la ligne orange à l'intérieur de la boîte représente la médiane. Les "moustaches" reflètent la variabilité des données au-delà des quartiles, décrivant la plage des valeurs de SSIM englobant la majorité des données. Les points en dessous de la boîte sont des valeurs aberrantes, indiquant des différences marquées par rapport à la majorité. Une médiane et des valeurs proches de 1 suggèrent une reconstruction efficace des images, tandis que des valeurs nettement inférieures à 1 pourraient indiquer des difficultés.

Erreur Quadratique Moyenne (MSE) :

Le MSE évalue la différence entre deux images, avec des valeurs toujours positives. La boîte représente l'écart interquartile (IQ), englobant la moitié médiane des valeurs de MSE. Le bord supérieur représente le troisième quartile (Q3), le bord inférieur indique le premier quartile (Q1), et la ligne orange à l'intérieur symbolise la médiane. Les "moustaches" reflètent la variabilité des données au-delà des quartiles, délimitant la plage des valeurs de MSE contenant la majorité des données. Les points au-dessus de la moustache supérieure sont des valeurs aberrantes, signalant des différences marquées par rapport à la majorité. Une médiane et des valeurs proches de 0 suggèrent une reconstruction efficace des images, tandis que des valeurs nettement supérieures à 0 pourraient indiquer des difficultés.

En conclusion, ces graphiques offrent un aperçu de la distribution des valeurs de SSIM et MSE pour les images de test, permettant d'évaluer l'efficacité de l'auto-encodeur dans la reconstruction des images. Ici l'auto-encodeur est relativement efficace, nous avons une médiane de similarité au-dessus de 0.7 sur 1 et la médiane de dissimilarité est légèrement au-dessus de 0.01. Ces deux mesures montrent que notre reconstruction d’image est plutôt bonne.

2.4 – Paramétrage

Dans cette partie, nous allons voir l'influence des différents paramètres déclarés jusqu'à maintenant sur les résultats du décodage.

Résultat du code : (L. 125-148)

[[0.7425921836861644, 0.7309761612279853, 0.9455727924698135, 0.9138925711320268, 0.6066343041707266], [0.7509531459338954, 0.7245354590477221, 0.9467507324481962, 0.9146210806812985, 0.609906404645776], [0.7679973268988578, 0.7291692950179834, 0.9495912061065703, 0.9123532720302785, 0.6010486880207125], [0.7770592143303047, 0.7296373258583645, 0.9457723575100724, 0.9156832520902075, 0.5945080856533462], [0.7778976531747183, 0.7388589863610336, 0.9472711875820461, 0.9134284193725696, 0.5987875530444887]]

On peut voir qu'augmenter le nombre d'époques augmente légèrement la similitude :

- Pour la première image : 74.3% → 77.8%.
- Pour la deuxième image : 73.1% → 73.9%.
- Pour la troisième image : 94.6% → 94.7%.
- Pour la quatrième image : 91.4% → 91.6%.
- Pour la cinquième image : 60.7% → 59.9%.

Les résultats ne sont évidemment pas concluants, il faudrait avoir des centaines d'époques d'écart pour vraiment voir des différences et avoir une conclusion mais cela demande des capacités et temps de traitement plus forts.

Il faut toutefois faire attention au sur ajustement des prédictions avec un nombre élevé d'époques.

Résultat du code : (L. 150-174)

[[0.7791363880475791, 0.7316278440424104, 0.9488514212778935, 0.9137192485198034, 0.5982589893130428], [0.7757232303774239, 0.7321932006604577, 0.9499324250781961, 0.9096220902832833, 0.5922645964092039], [0.7841359917517359, 0.7387586121356406, 0.9501786822642279, 0.9135342239399841, 0.5959953659807827], [0.7783718777591229, 0.7415749942378086, 0.9505766059803491, 0.9095220196451081, 0.5928497568434548], [0.7780493433015052, 0.7477117563645143, 0.9540143694230631, 0.9118360117901414, 0.599118076408507]]

On peut voir qu'augmenter le nombre de neurones n'augmente pas vraiment la similitude :

- Pour la première image : 77.9% → 77.8%
- Pour la deuxième image 73.2% → 74.8%
- Pour la troisième image : 94.9% → 95.4%
- Pour la quatrième image : 91.4% → 91.2%
- Pour la cinquième image : 59.8% → 59.9%

3 – Auto-encodeur profond

3.1 Définition du modèle

Résultat de la fonction summary**()** sur l'auto encodeur :

Model: "model_3"

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 784)]	0
dense_2 (Dense)	(None, 128)	100480
dense_3 (Dense)	(None, 64)	8256
dense_4 (Dense)	(None, 32)	2080
dense_5 (Dense)	(None, 64)	2112
dense_6 (Dense)	(None, 128)	8320
dense_7 (Dense)	(None, 784)	101136

Total params: 222384 (868.69 KB)

Trainable params: 222384 (868.69 KB)

Non-trainable params: 0 (0.00 Byte)

Comme pour le premier auto-encodeur que nous avons fait, nous avons une couche d'entrée pour nos données (nos images d'habits) mais cette fois-ci nous avons plus de couches cachées. Les couches 'input_3', 'dense_2' et 'dense_3' représentent notre encodeur mais ici nous réduisons les neurones petit à petit, de 784, à 128, puis à 64 et enfin à 32 pour finir avec la même réduction que notre premier encodeur.

Le décodeur est représenté par les couches 'dense_5', 'dense_6' et 'dense_7'. Comme pour l'encodeur nous avons deux couches de plus que notre premier décodeur, nous passons de 32 neurones à 64 puis à 128 pour finir à 784 comme nos données d'entrée.

Comme nous avons plus de couches, forcément nous avons plus de paramètres modifiables, pour cet auto-encodeur nous avons plus de quatre fois plus que notre premier auto-encodeur.

3.3 Analyse des résultats de l'auto-encodage avec le modèle appris

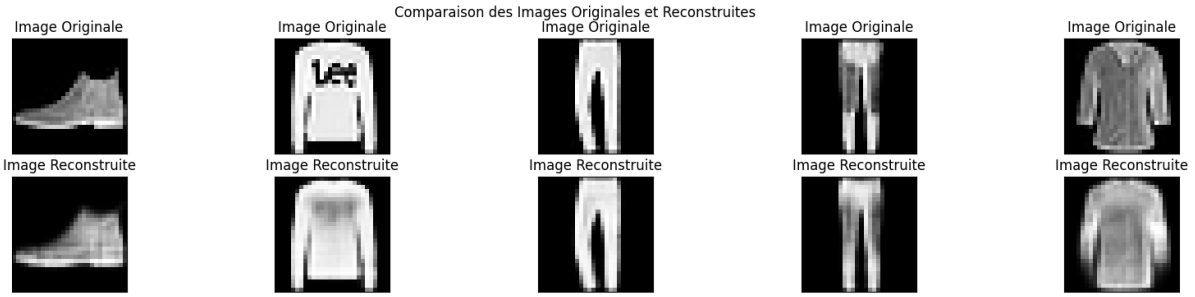


Figure 5 : Images obtenu de l'auto-encodeur profond (input / output)

Comparé aux images de la première partie, nous remarquons qu'avec l'auto-encodeur profond, nous conservons plus de détails et les images sont bien plus nettes. : Images de tests (en haut) et images après le décodage (en bas)

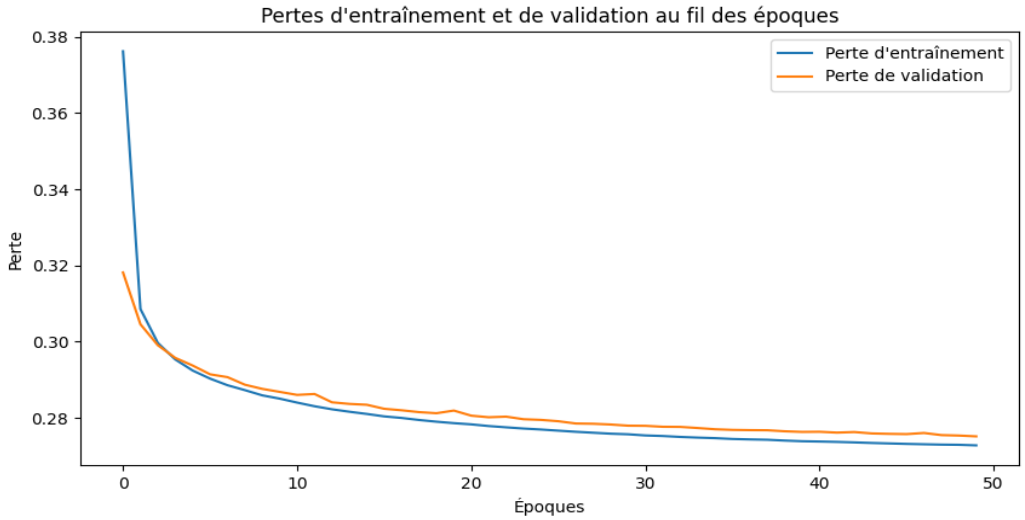


Figure 6 : Affichage des courbes de perte d'entraînement et de validation avec l'auto-encodeur profond

D'après le graphique, on peut faire plusieurs observations :

On observe une diminution des pertes, tant pour l'entraînement que pour la validation, ce qui suggère que le modèle apprend de manière efficace à partir des données.

Il est à noter que les courbes de perte convergent vers une valeur minimale, indiquant ainsi un ajustement adéquat du modèle sans signe apparent de surapprentissage.

Après environ 10 époques, on observe une stabilisation des pertes avec de légères fluctuations. Cela pourrait signifier que le modèle a atteint un plateau où il ne peut plus améliorer significativement sa performance avec les données d'entraînement actuelles.

La perte d'entraînement reste constamment inférieure à la perte de validation, conformément aux attentes en apprentissage supervisé. Toutefois, l'écart entre les deux n'est pas très grand, ce qui est un signe positif.

Le modèle semble bien performer et s'ajuster correctement aux données d'entraînement et de validation. Il serait peut-être judicieux de considérer l'arrêt de l'entraînement après environ 10 époques, étant donné que les pertes semblent se stabiliser à ce stade.

Conclusion

Dans ce TP, nous avons exploré comment utiliser des auto-encodeurs pour réduire la dimension des images de basse résolution de 'Fashion-MNIST'. Nous avons suivi deux approches principales : un auto-encodeur simple avec une seule couche cachée, et un auto-encodeur profond avec plusieurs couches cachées.

Auto-encodeur simple :

Nous avons créé un modèle avec une couche d'entrée, une couche cachée de 32 neurones, et une couche de sortie. L'apprentissage a montré une réduction des pertes sur les ensembles d'entraînement et de validation. Les résultats de décodage ont bien reconstruit les images, même si quelques détails étaient perdus. Les métriques comme l'Indice de Similarité Structurelle (SSIM) et l'Erreur Quadratique Moyenne (MSE) ont confirmé la performance du modèle.

Paramétrage :

Nous avons examiné l'effet des paramètres comme le nombre d'époques et de neurones sur les résultats. Bien que les changements n'aient pas été très marqués ici, des ajustements plus importants pourraient nécessiter une analyse approfondie.

Auto-encodeur profond :

Nous avons étendu l'approche en créant un auto-encodeur profond avec plusieurs couches cachées. Ce modèle a montré une performance similaire avec une diminution des pertes pendant l'apprentissage. La reconstruction des images a bien fonctionné, montrant la capacité de l'auto-encodeur profond à comprendre des informations plus complexes.

Les deux approches ont donné des résultats satisfaisants, mais l'auto-encodeur profond avec plusieurs couches cachées a montré une capacité supérieure à capturer des informations complexes. Cela souligne l'importance de la profondeur du modèle dans la représentation des données.

En résumé, ce TP a mis en avant l'efficacité des auto-encodeurs dans la compression et la reconstruction d'images, en mettant en lumière l'impact positif de la profondeur du modèle. Ces résultats peuvent être utilisés comme base pour des applications plus avancées de traitement d'images et de compression de données.