Comparaison des modèles d'extraction de caractéristiques discriminantes

Mohamed BEN HAMDOUNE, Lucas ISCOVICI

Paris, France

Abstract

En machine learning, l'extraction des caractéristiques commence à partir d'un ensemble initial de données mesurées et construit des valeurs dérivées (caractéristiques) destinées à être informatives et non redondantes. Cela a pour conséquence de faciliter ainsi les étapes ultérieures d'apprentissage et de généralisation en conduisant à de meilleures interprétations humaines.
Dans ce document, nous proposerons quelques techniques de visualisation et de réduction de données sur le jeu de données Fashion-MNIST en association avec un modèle d'apprentissage.

Keywords: Auto-encodeur, ACP, t-SNE, K-means, Fashion-MNIST

Contents

1. Introduction

L'idée des auto-encodeurs a été mentionnée pour la première fois en 1986, dans un article analysant de manière approfondie la backpropagation [8]. Dans les années suivantes, l'idée a refait surface dans d'autres documents de recherche. Un article publié en 1989 a permis d'introduire plus avant les auto-encodeurs en mettant en la capacité d'extraire les caractéristiques linéaires [1]. Par la suite, il a été découvert que l'on pouvait découvrir des représentations factorielles non linéaires [4].
Dans ce contexte ci, nous avons eu comme mini-projet de réaliser la comparaison entre l'ACP et les auto-encodeurs fasse à un jeu de données complexes d'une certaine manière.
Ensuite, nous essayerons de visualiser ces résultats en utilisant une méthode de clustering afin de voir les conséquences de l'encodage.

1.1. Analyse en Composantes Principales

L'ACP consiste à transformer des variables corrélées en axes décorélés les un des autres. Ces axes sont composés uniquement de combinaisons linéaires des variables précédentes.
Cette méthode est une des première méthode d'analyse factorielle et moteur central de toute autre méthode d'analyse factorielle.
Dans notre cas, nous projeterons les points du jeux de données sur les deux premiers axes factoriels créant ainsi un nouvel espace vectoriel, les points du jeu de données initial projetés sur un espace réduit, nous esperons pouvoir grâce à ces nouveaux axes et en fonction de l'inertie expliquée, faire parler notre jeu de données et avoir une idée plus claire de la structure en classe de celui-ci.

1.2. t-distributed stochastic neighbor embedding

t-SNE tente de trouver une configuration optimale selon un critère de théorie de l'information pour respecter les proximités entre points : deux points qui sont proches (resp. éloignés) dans l'espace d'origine devront être proches (resp. éloignés) dans l'espace de faible dimension.
Une distribution de probabilité est également définie de la même manière pour l'espace de visualisation. L'algorithme t-SNE consiste à faire concorder les deux densités de probabilité, en minimisant la divergence de Kullback-Leibler entre les deux distributions par rapport à l'emplacement des points sur la carte. La divergence de Kullback-Leibler n'est pas une métrique propre,

car elle n'est pas symétrique et ne satisfait pas non plus l'inégalité du triangle. Donc, la divergence KL est préférable de ne pas être interprétée comme une "mesure de distance" entre les distributions, mais plutôt comme une mesure de l'augmentation d'entropie due à l'utilisation d'une approximation de la distribution vraie plutôt que de la distribution vraie elle-même.

1.3. Auto-encodeur

De nos jours, le débruitage des données et la réduction de la dimensionnalité pour la visualisation des données sont considérés comme deux principales applications pratiques intéressantes des autoencodeurs.
Si le seul but des auto-encodeurs était de copier l'entrée dans la sortie, ils seraient inutiles. En effet, nous espérons qu'en entraînant l'auto-codeur à copier l'entrée dans la sortie, la représentation latente h aura des propriétés utiles.
Avec des contraintes de dimensionnalité et de parcimonie appropriées, les auto-encodeurs peuvent apprendre des projections de données plus intéressantes que l'ACP ou d'autres techniques de base.
Les auto-encodeurs sont appris automatiquement à partir d'exemples de données. Cela signifie qu'il est facile de former des instances spécialisées de l'algorithme qui fonctionneront bien avec un type d'entrée spécifique.
Un auto-encodeur est un réseau de neurones comportant trois couches: une couche d'entrée, une couche cachée (codage) et une couche de décodage. Le réseau est formé pour reconstruire ses entrées, ce qui oblige la couche cachée à essayer d'obtenir de bonnes représentations des entrées.

1.4. PSNR et MSE

Pour quantifier les résultats, nous utiliserons deux méthodes. Le PSNR (Peak Signal to Noise Ratio) calcule le rapport signal / bruit de pointe, en décibels, entre deux images. Ce rapport est souvent utilisé comme mesure de qualité entre l'image originale et une image compressée. Plus le PSNR est élevé, meilleure est la qualité de l'image compressée ou reconstruite. Ensuite l'erreur moyenne carrée et le PSNR sont les deux mesures d'erreur utilisées pour comparer la qualité de la compression d'image. Le MSE représente l'erreur au carré cumulative entre l'image compressée et l'image d'origine, tandis que le PSNR représente une mesure de l'erreur maximale. Plus la valeur de MSE est faible, plus l'erreur est faible.

1.5. NMI et ARI

L'information mutuelle (MI) de deux variables aléatoires est une mesure de la dépendance mutuelle entre les deux variables. Plus précisément, il quantifie la "quantité d'informations" (en bits) obtenues à propos d'une variable aléatoire en observant l'autre variable aléatoire. Nous utilisons une version normalisé, c'est à dire que les valeurs sont comprises entre 0 et 1 d'où le nom de NMI (Normalized Mutual Information).

La mesure de l'indice Rand en statistiques est une mesure de la similarité entre deux clusters de données. l'index de Rand est lié à la précision, mais est applicable même lorsque les labels ne sont pas utilisées puisque nous mesurons la consistance (le taux d'accord) entre deux clusters. Nous utilisons une version ajusté, c'est à dire que les valeurs sont comprises entre 0 et 1 d'où le nom de ARI (Adjusted Rand Index).

2. Fashion MNIST

Fashion-MNIST est un ensemble de données d'images d'article de Zalando, consistant en un ensemble de formation de 60 000 exemples et un ensemble de test de 10 000 exemples. Chaque exemple est une image en niveaux de gris de 28x28, associée à une étiquette de 10 classes.
Chaque image a une hauteur de 28 pixels et une largeur de 28 pixels, pour un total de 784 pixels. Chaque pixel est associé à une valeur de pixel unique, indiquant la clarté ou l'obscurité de ce pixel, les chiffres les plus élevés signifiant plus sombre. Cette valeur de pixel est un entier compris entre 0 et 255.
Selon les auteurs, les données Fashion-MNIST sont destinées à remplacer directement les anciennes données de chiffres manuscrits du MNIST, car il y avait plusieurs problèmes avec les chiffres manuscrits. Par exemple, il était possible de distinguer correctement plusieurs chiffres en regardant simplement quelques pixels et cela même avec des classificateurs linéaires, il était possible d'obtenir une précision de classification élevée. Les données Fashion-MNIST s'annoncent plus diverses, de sorte que les algorithmes d'apprentissage automatique doivent apprendre des fonctionnalités plus avancées pour pouvoir séparer les classes de manière fiable.

Figure 1: Visualisation des données

2.1. Labels

Chaque exemple de *train* et *test* est attribué à l'un des labels suivants:

1. T-shirt / Haut
2. Pantalon
3. Pull
4. Robe
5. Manteau
6. Sandales
7. Chemise
8. Baskets
9. Sacs
10. Bottines

3. Réduction de dimension pour Fashion-MNIST avec une ACP

Étant donné que nous, en tant qu'êtres humains, aimons nos diagrammes à deux et trois dimensions, commençons par cela et générons, à partir des 784 dimensions d'origine, les deux premières composantes principales. Et nous verrons également quelle part de la variation dans le jeu de données total est réellement prise en compte.
L'avantage de l'ACP (et de la réduction de la dimensionnalité en général) est qu'elle compresse nos données pour en faire une modélisation plus efficace. Cela signifie, par exemple, que les variables hautement corrélées et colinéaires seront compressées. En tant que technique de prétraitement, la réduction de la dimensionnalité est particulièrement utile lorsqu'elle est utilisée pour redéfinir un jeu de données avec de nombreuses variables.

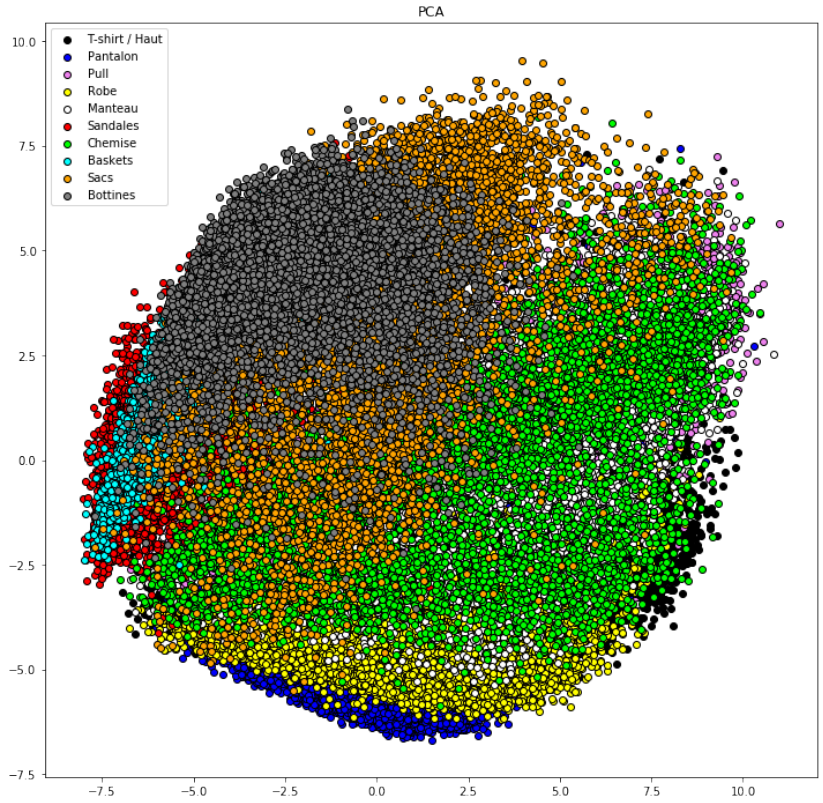

Figure 2: Visualisation de l'ACP

L'ACP résume le jeu de données dans sa globalité, les vêtements du bas sont bien représentées et bien séparées des *Pantalons*. Le problème de ce nuage de points finalement est qu'il n'y a pas de séparation dans l'espace discriminante mais plutôt un rassemblement de toutes les classes indiquant finalement qu'il s'agit de vêtements dans leurs formes.

En ce qui concerne le nombre de composantes principales, nous pouvons constater que 75% de l'information est disponible avec seulement 14 facteurs.

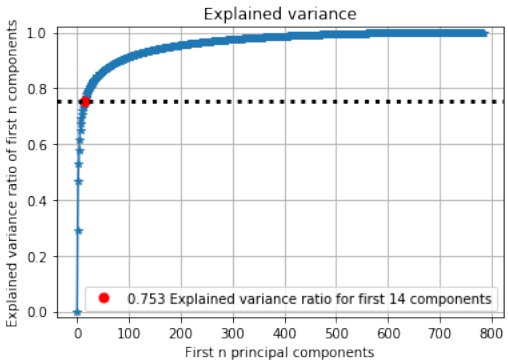

Figure 3: Taux d'informations fixé à 0.75

Une représentation de chaque vêtement est disponible dans le notebook afin d'éviter de surcharger le rapport. Une reconstruction à 14 vecteurs semble bien se dérouler. Il est particulièrement difficile à appliquer aux images comportant des trous. L'ACP a du mal à trouver les limites.
Fashion MNIST est un jeu de données plus complexe que MNIST sur uniquement des chiffres allant de 0 à 9.

En appliquant cet exemple de technique de reconstruction à un plus grand nombre de données, nous pouvons commencer à comprendre quelles caractéristiques sont et ne sont pas facilement modélisées.

Dans ce cas, nous soupçonnons que trouver les bords des vêtements qui ne sont pas contigus (par exemple le corset d'une paire de talons ou les sangles d'un sac à main) est la partie la plus difficile à modéliser de cet ensemble de données. Cela indique que nous devrions nous préoccuper un peu de cet aspect des données même si l'ACP comme technique de compression pure et dure reste acceptable.

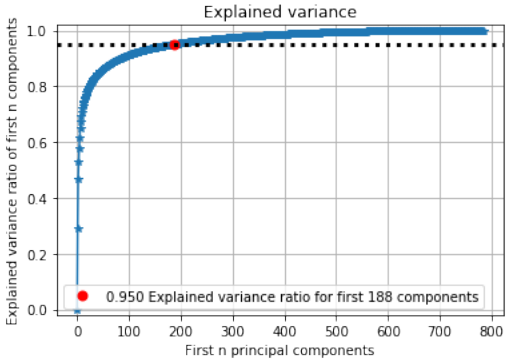

Figure 4: Taux d'informations fixé à 0.95

95% de l'information est représentées par 188 composants, cela signifie seulement 23% des facteurs (784 colonnes à la base). Á titre d'information, nous avons 99% des informations contenus dans les 459 premières composantes. Voici un tableau résumant les résultats sur l'ACP lorsque l'on applique la reconstruction de la matrice suivants les 3 pourcentages de variance choisi :

Nombre de composants	PSNR Moyen	MSE Moyen	Pourcentage de variance
14	17.15	0.021	75%
188	24.62	0.004	95%
459	32.17	0.0008	99%

4. Réduction de dimension pour Fashion-MNIST avec t-SNE

Cette technique est extrêmement populaire dans la communauté du *deep learning*. Malheureusement, la fonction de coût de la t-SNE implique des mécanismes mathématiques non triviaux et nécessite des efforts de compréhension importantes.
Mais, grosso modo, ce que t-SNE essaie d'optimiser, c'est de préserver la topologie des données. Pour chaque point, il construit une notion des autres points qui sont ses "voisins", essayant de faire en sorte que tous les points aient le même nombre de voisins. Ensuite, il essaie de les intégrer afin que tous les points aient le même nombre de voisins.
La similarité entre les distributions est définie à l'aide de la mesure de la divergence de Kullback-Leibler qu'on essayera de minimiser à travers une recherche parmi une combinaison de valeurs.

```
params = {'perplexity': [x for x in range(5,200,20)],
          'learning_rate' : [x for x in range(100,1000,100)],
          'n_iter' : [x for x in range(100,1000,100)],
          'angle' : [0.2,0.3,0.4,0.5,0.6,0.7,0.8]}
```

La suite de cette combinaison de paramètres fait que sans surprise, nous avons pour la perplexité, le pas d'apprentissage, et le nombre d'apprentissage. Les valeurs maximum qui sont respectivement 185,900,900. Pour l'angle celui ci est de 0.3, on aurait pu s'attendre à 0.2 mais l'angle n'as pas de grande incidence.
Nous utilisons l'algortihme de Barnes-Hut [9] et pour l'angle, des valeurs plus élevées donnent une optimisation plus rapide mais moins précise. La valeur de la divergence KL de différentes perplexités ne peut être comparée pour évaluer la qualité des clusters, car la divergence finale de KL diminue généralement à mesure que la perplexité augmente [2].
Le nuage des points a du sens puisque pour les vêtements concernant les *Sandales*, *Bottines* et *Baskets* partageant une certaines formes communes sont bien séparés des autres vêtements et regrouper ensemble. Les *Sacs* forment un regroupement non similaires et sont regroupés séparément des autres, ceci est la même conséquence pour *Pantalons*.
En revanche pour les autres vêtements (*Pull*,*Robe*,*Manteau*,*Chemise*, *T-shirt*), ils n'existent pas de distinction flagrante. On peut noter qu'une certaine majoritaire des *T-shirt* forment un regroupement et de même pour les *Robes* en jaunes. Ce qui est intéressant dans notre situation, c'est de constater que les *Robes* sont situés entre les *Pantalons* et les *Hauts* : l'algorithme ici nous montre finalement qu'il a réussi à comprendre que c'était un mixte.

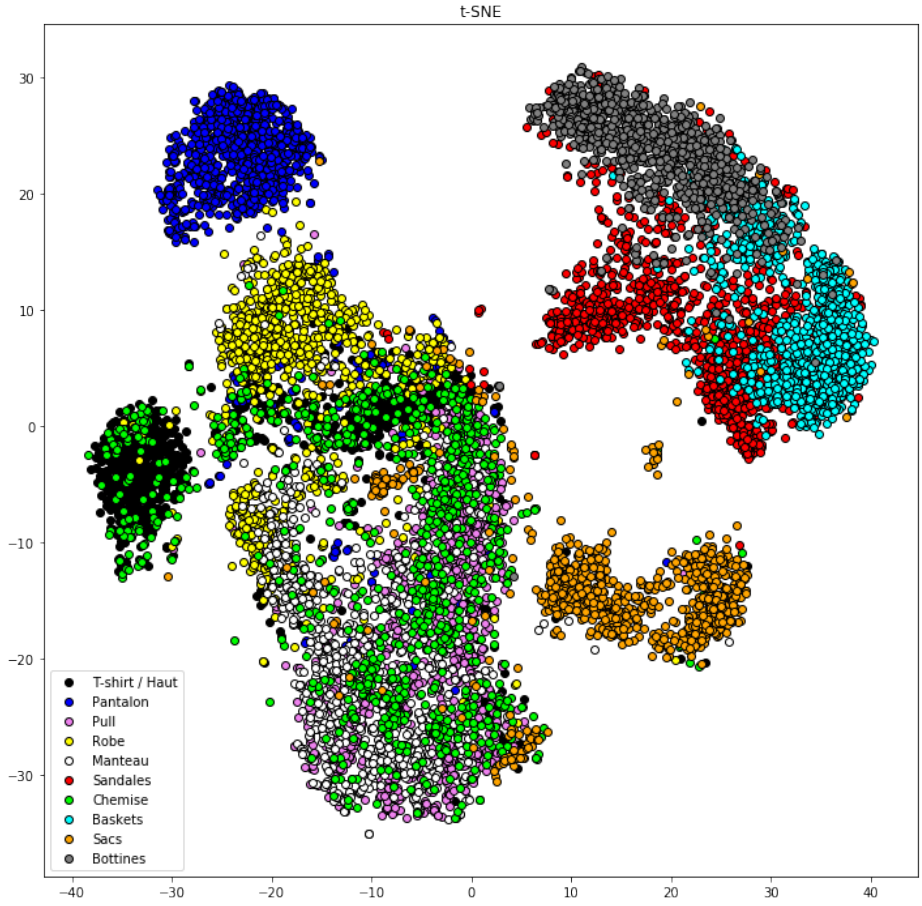

Figure 5: Visualisation de la t-SNE

Si la perplexité est trop faible, nous ne capturerons pas assez de voisins. au contraire, un nombre excessif approchera des valeurs trop loin. Avec les grands ensembles de données, une amélioration de ce paramètre est recommandée comme nous le fessons puisque les valeurs recommandées sont de 5 à 50 (dans notre cas il est à 185).
En pratique, nous pensons que nous devrions faire varier les paramètres et voir laquelle "a du sens". Bien sûr, c'est une suggestion qui n'a pas de base mathématique solide, mais après tout, le t-SNE est une méthode de visualisation, il n'y a donc pas vraiment de manière correcte ou optimale de le faire tout comme il n'y a pas d'hypothèse ou de test statistique impliqué. Cela dépend de ce que nous voulons montrer.

Notre divergence de Kullback-Leibler est de 1.0284, ce qui signifie qu'il faut en moyenne environ 1 bit de codage d'échantillons de P en utilisant un code optimisé pour Q plutôt que le code optimisé pour P. Nous comparerons ce résultat avec la t-SNE effectué sur les données en sortie de l'encodeur.

5. Réduction de dimension pour Fashion-MNIST avec un deep auto-encodeur

L'extension de l'auto-encodeur simple est le deep auto-encodeur, la seule différence est le nombre de couches cachées. Les couches cachées supplémentaires permettent à l'auto-encodeur d'apprendre mathématiquement des modèles sous-jacents plus complexes dans les données.
La première couche du Deep auto-encodeur peut apprendre des caractéristiques de premier ordre dans l'entrée brute (telles que des bords dans une image). La deuxième couche peut apprendre des caractéristiques de second ordre correspondant à des motifs d'apparition de caractéristiques de premier ordre (par exemple, en ce qui concerne les bords qui ont tendance à apparaître complexe comme détecter les contours ou coins). Les couches plus profondes du deep auto-encodeur ont tendance à apprendre des fonctionnalités encore plus élevées. Pour tout mettre ensemble : nous avons besoin de couches supplémentaires pour pouvoir traiter des données plus complexes comme Le jeux de données Fashion MNIST.

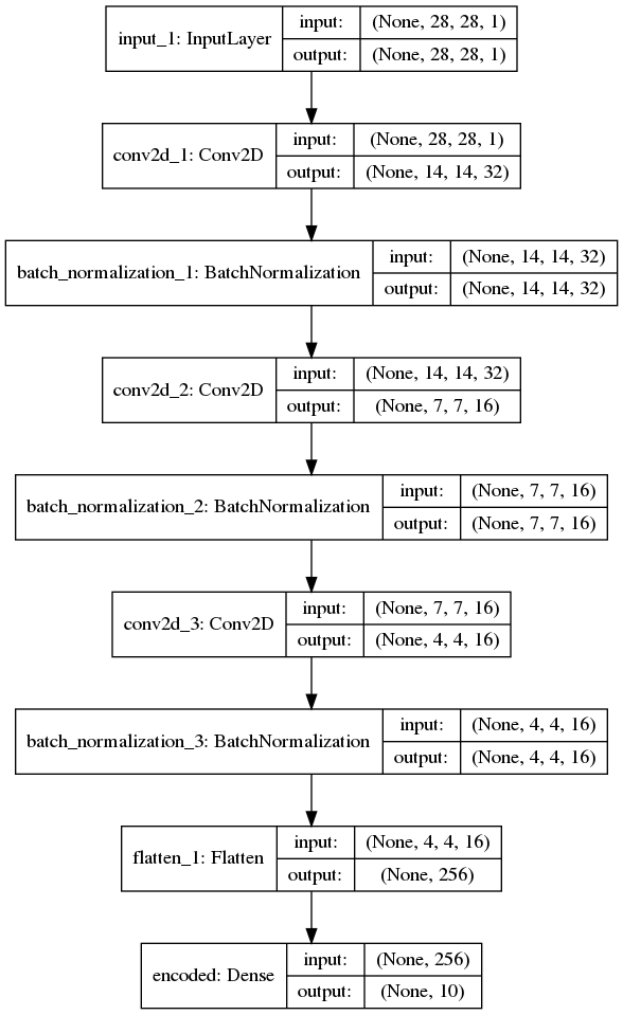

Figure 6: Graphe de la partie Encodeur.

Le principe reste le même, mais en utilisant des images (vecteurs 3D) au lieu de vecteurs 1D aplatis (*flatten*). L'image d'entrée est sous-échantillonnée (*down-sampling*) pour donner une représentation latente de dimensions plus petites et forcer l'auto-codeur à apprendre une version compressée des images. Par conséquent, les AEC (auto-encodeurs convolutionnel) sont des extracteurs de fonctionnalités à usage général différemment des AE qui ignorent complètement la structure de l'image 2D. En fait, dans les AE, l'image doit être *flatten* dans un seul vecteur et le réseau doit être construit en tenant compte de la contrainte imposée au nombre d'entrées. En d'autres termes,

les AE introduisent une redondance dans les paramètres, forçant chaque caractéristique à être globale (c'est-à-dire qu'elle couvre tout le champ visuel), contrairement aux AEC.

Les auto-encodeurs convolutionnels abordent la tâche de définition de filtre sous un angle différent: au lieu de concevoir manuellement des filtres convolutionnels, nous laissons le modèle apprendre les filtres optimaux qui minimisent l'erreur de reconstruction. Ces filtres peuvent ensuite être utilisés dans toute autre tâche de vision par ordinateur.

Les auto-encodeurs convolutionnels (AEC) sont un type de réseaux de neurones convolutifs [6] (CNN): la principale différence entre l'interprétation commune de CNN et de AEC est que les premiers sont formés de bout en bout pour apprendre les filtres et combiner des fonctionnalités dans le but de classer leurs entrées. Les AEC ne sont utilisés que pour apprendre des filtres capables d'extraire des caractéristiques pouvant être utilisées pour reconstruire alors que les CNN sont généralement utilisés dans le cadre de l'apprentissage supervisé.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 28, 28, 1)	0
conv2d_1 (Conv2D)	(None, 14, 14, 32)	320
batch_normalization_1 (Batch	(None, 14, 14, 32)	128
conv2d_2 (Conv2D)	(None, 7, 7, 16)	4624
batch_normalization_2 (Batch	(None, 7, 7, 16)	64
conv2d_3 (Conv2D)	(None, 4, 4, 16)	2320
batch_normalization_3 (Batch	(None, 4, 4, 16)	64
flatten_1 (Flatten)	(None, 256)	0
encoded (Dense)	(None, 10)	2570
dense_1 (Dense)	(None, 256)	2816
reshape_1 (Reshape)	(None, 4, 4, 16)	0
conv2d_4 (Conv2D)	(None, 4, 4, 16)	2320
batch_normalization_4 (Batch	(None, 4, 4, 16)	64
up_sampling2d_1 (UpSampling2	(None, 8, 8, 16)	0
conv2d_5 (Conv2D)	(None, 8, 8, 16)	2320
batch_normalization_5 (Batch	(None, 8, 8, 16)	64
up_sampling2d_2 (UpSampling2	(None, 16, 16, 16)	0
conv2d_6 (Conv2D)	(None, 14, 14, 32)	4640
batch_normalization_6 (Batch	(None, 14, 14, 32)	128
up_sampling2d_3 (UpSampling2	(None, 28, 28, 32)	0
conv2d_7 (Conv2D)	(None, 28, 28, 1)	289

Total params: 22,731
Trainable params: 22,475
Non-trainable params: 256

Figure 7: Résumé de l'architecture.

De plus nous utilisons, la Batch Normalization [5]. En intégrant la normalisation à l'architecture de modèle, nous pouvons améliorer la vitesse d'apprentissage et accorder moins d'attention aux paramètres d'initialisation. La Batch Normalization agit en outre comme un régularisateur, réduisant (voire éliminant parfois) la nécessité du Dropout.

Finalement, l'autre aspect de notre architecture est l'UpSampling [3] (car il faut bien que nous minimisions l'erreur pixel par pixel de manière binaire afin que les images d'entrée et sortie se ressemble aux maximum à la suite des convolutions). Nous pouvions utiliser l'opération de déconvolution (Transpose2D dans notre cas) même si théoriquement cela nous devrait de bon

résultat.
Il a était montré en pratique que l'UpSampling donnait des résultats meilleurs pour des formes complexes [7]. Le paramètre par défaut sur Keras est une interpolation "nearest", or nous avons décider d'utiliser l'interpolation bilinéaire. La principale différence en interpolation bilinéaire est qu'elle utilise 4 voisins les plus proches pour générer une surface de sortie et de pallier aux problèmes d'aliasing. Le calcul en deviens plus alourdi mais la taille de nos données ne sont pas grandes donc cela ne nous impacte pas beaucoup (3s l'epoch au lieu de 2s).
Voici la reconstruction de 10 exemples en sortie du auto-encodeur convolutionnel:

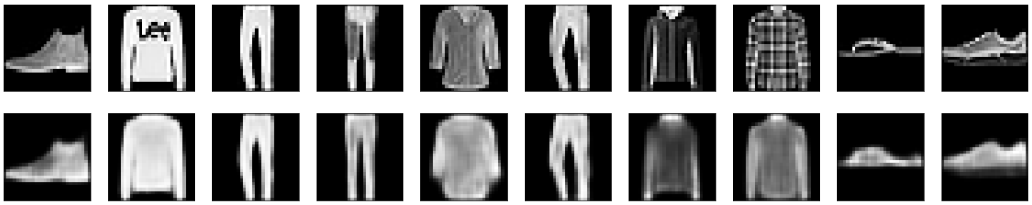

Figure 8: Reconstruction des images de Fashion MNIST.

Le PSNR est de 18 en moyenne et le MSE moyen de 0.015, la taille de notre vecteur d'espace latent est de 10 (ce qui correspond aux nombres de classes finalement) . La qualité de reconstruction est faible certes, mais L'ACP avec les 14 premières composantes était autour de 17 à titre de comparaison. Nous avons choisi comme optimizer Adam avec un learning rate 1e-03 et comme fonction coût l'entropie croisé binaire. Le but est de comparé chaque pixel un à un, c'est plus précis que le MSE car il sera plus sensible au forme des images complexes. Nous avons effectuer plusieurs test avec Adagrad puis Rmsprop pour en conclure que ces deux méthodes souffraient d'une instabilité sur les valeurs mais nous effectuons beaucoup d'epoch (100) et cela n'a pas de grande conséquence par la suite. En revanche SGD converge beaucoup trop lentement alors que notre nombre d'epochs est relativement élevée.

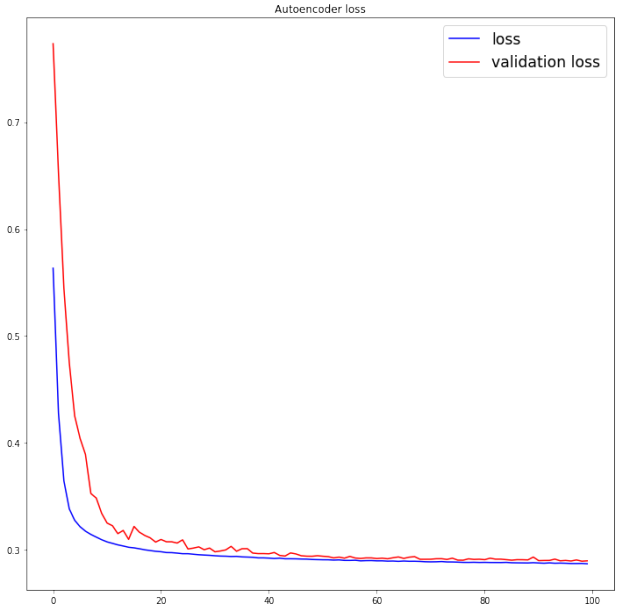

Figure 9: Courbe de décroissance de la fonction coût.

En validation, on remarque une certaine instabilité sur les valeurs de retours pour les 40 premières epochs alors que loss sur le jeux de données d'entraînement suis une décroissance logarithmique stable. Ensuite on remarque qu'il n'y a pas de problème de sur-apprentissage ou de sous-apprentissage (que l'on a pu noter en utiliant SGD comme optimizer).

6. Clustering dans un espace latent de petite dimension

6.1. Clustering avec K-means

Nous allons d'abord effectuer un clustering sur les images de test non encodées et évaluer notre modèle suivant l'accuracy, la NMI, et l'ARI et ensuite sur les images encodées. Nous rappelons que nous avons choisi explicitement un espace latent très faible et aussi celui qui correspond finalement aux nombre de classes afin de voir les limites de l'auto-encodeur convolutionnel et aussi les performances de K-means. Ensuite, K-means a un point faible qui est la malédiction de la dimensionnalité donc nous avons décidé de pas prendre un vecteur de taille 188 ou 459 (ce qui correspond à 95% ou 99% de l'information contenu selon l'ACP).

Images	Accuracy	NMI	ARI
Encodées	56,35%	59,79%	43,60%
Non Encodées	48,64%	51,59%	35,15%

Lors de l'entraînement de l'auto-encodeurs, celui si n'a pas appris sur ce jeux de données puisqu'il servait de validation (ceux ne sont pas des images qui ont contribuait directement à la mise à jour des poids des différentes couches du réseau mais plutôt à l'optimisation de paramètres). L'accuracy a eu un gain de 16% et 15% pour la NMI puis 22% pour l'ARI.
La raison est que tout simplement derrière les valeurs rendu par l'encodeur se cache les caractéristiques discriminantes pour chaque classes, ce qui a pour finalité d'améliorer le clustering au niveau de la précision des classes mais aussi sur les deux autres aspects (NMI et ARI). La quantité d'information moyenne obtenue sur un cluster comparer à un autre est plus importante, cela signifie que l'encodeur améliore la répartition des données dans un cluster et de plus il y a une meilleure mesure de l'indépendance entre chaque cluster.

6.2. Visualisation avec t-SNE

La divergence KL est de 0.8863 soit un gain comparé aux jeux de test non encodée qui était de 1.0284. Ces 16% d'amélioration sont dûs à l'encodage de ces données comme vu précédemment grâce à l'amélioration du clustering en évaluant la NMI. Nos données étant mieux distribuées, nous pouvons mieux voir une séparation entre chaque classe comme l'indique ce nuage de points.

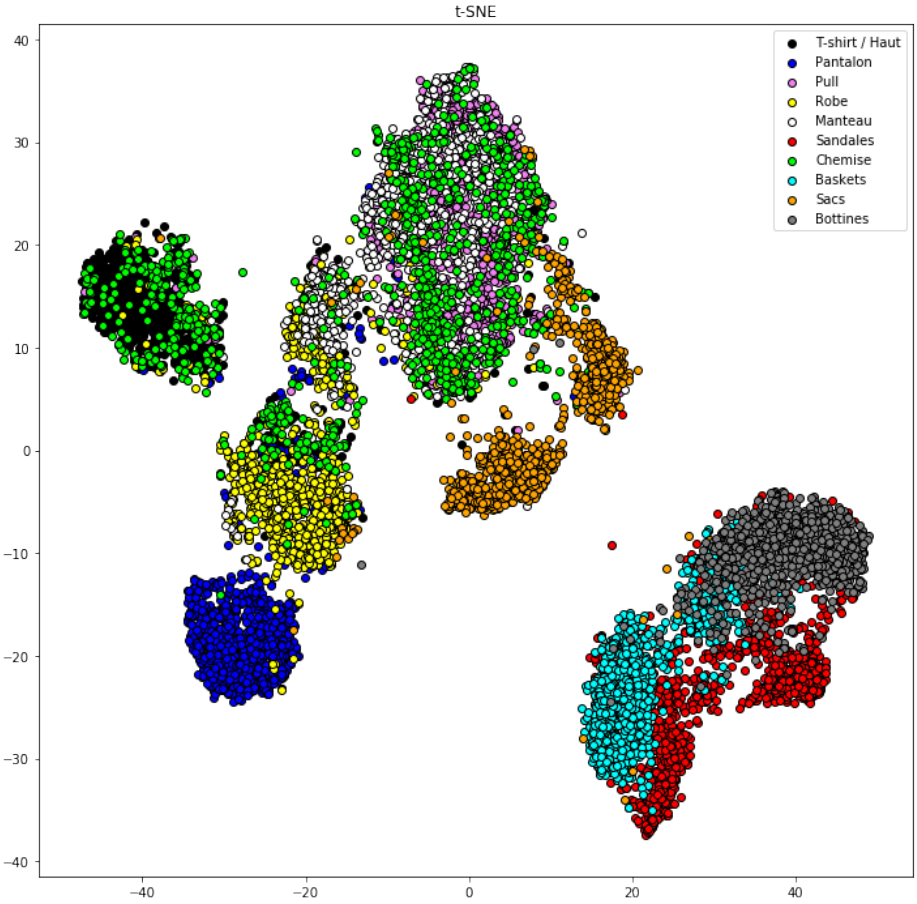

Figure 10: t-SNE sur les images encodées.

Les *T-shirt/Haut* sont mieux discriminés qu'auparavant ainsi que les chaussures entre eux. Il y a un regroupement plus claires des différents types de chaussures.

7. Conclusion

Nous avons appris beaucoup à travers ce mini projet puisqu'il fait intervenir techniques de visualisation et de clustering. Nous avons aussi pu voir les différents caractéristiques des auto-encodeurs et leurs variantes. À la suite de l'exercice sur les encodeurs, nous avons constater que les auto-encodeurs pouvaient nous aider à améliorer les résultats d'un clustering dans le cadre d'un jeux de données dont on ne connaîtrait pas la ou les classes associées. Ensuite la t-SNE, nous permettra de visualiser le sens de nos données à travers la répartition des points. De plus, nous avons constater que la divergence de Kullback-Leibler et sa relation avec l'information mutuelle était utile pour évaluer nos résultats. Pour la suite, nous aurions pu construire une architecture en utilisant un autoencodeur variationnel en plus des couches convolutionnels puis essayer plusieurs paramètres pour K-means afin de retenir celui qui nous aurait donner la meilleur précision. Finalement, c'était un projet qui nous a permis d'explorer, visualiser et classifier le jeu de donnnées Fashion-MNIST qui est plus complexe que MNIST.

References

[1] Baldi, P., Hornik, K., 12 1989. Neural networks and principal component analysis: Learning from examples without local minima", ne. Neural Networks 2, 53–58.

[2] Cao, Y., Wang, L., 2017. Automatic selection of t-sne perplexity. CoRR abs/1708.03229.
URL http://arxiv.org/abs/1708.03229

[3] Chen, J., Adams, A., Wadhwa, N., W. Hasinoff, S., 11 2016. Bilateral guided upsampling. ACM Transactions on Graphics 35, 1–8.

[4] Hinton, G. E., Zemel, R. S., 1994. Autoencoders, minimum description length and helmholtz free energy. In: Cowan, J. D., Tesauro, G., Alspector, J. (Eds.), Advances in Neural Information Processing Systems 6. Morgan-Kaufmann, pp. 3–10.
URL http://papers.nips.cc/paper/798-autoencoders-minimum-description-length-an
pdf

[5] Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167.
URL http://arxiv.org/abs/1502.03167

[6] LeCun, Y., Haffner, P., Bottou, L., Bengio, Y., 1999. Object recognition with gradient-based learning. In: Shape, Contour and Grouping in Computer Vision. Springer-Verlag, London, UK, UK, pp. 319–.
URL http://dl.acm.org/citation.cfm?id=646469.691875

[7] Odena, A., Dumoulin, V., Olah, C., 2016. Deconvolution and checkerboard artifacts. Distill.
URL http://distill.pub/2016/deconv-checkerboard

[8] Rumelhart, D. E., Hinton, G. E., Williams, R. J., 1986. Learning internal representations by error propagation. In: Rumelhart, D. E., Mcclelland, J. L. (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations. MIT Press, Cambridge, MA, pp. 318–362.

[9] van der Maaten, L., 2013. Barnes-hut-sne. CoRR abs/1301.3342.
URL http://arxiv.org/abs/1301.3342