

TP4 Apprentissage profond

Introduction aux modèles Autoencoders

Jérôme Buisine
jerome.buisine@univ-littoral.fr

13 décembre 2023

Durée : 3h

L'objectif de ce TP est la prise en main de la librairie Keras afin de mettre en place des architectures plus complexes comme les Autoencoders.

1 Ressources

Voici un ensemble de ressources qui peuvent vous être utiles durant ce TP :

- 1. [Documentation](#) officielle de l'outil Git ;
- 2. [Gitlab](#) : interface web pour la gestion de projets versionnés sous Git ;
- 3. [pyenv](#) : permet la gestion d'environnement Python.
- 4. [matplotlib](#) : librairie Python qui permet un affichage rapide de données.
- 5. [jupyter](#) : un outil de travail permettant une interaction rapide et visuelle avec une console Python.
- 6. [scikit-learn](#) : documentation de l'API `scikit-learn` utile pour le *machine learning*.
- 7. [Keras](#) : documentation de l'API Keras pour le *deep learning*.

2 Configuration de l'environnement

Créez un dossier à la racine de votre projet nommé `tp4-autoencoder`. Dans ce dossier et depuis votre terminal, lancez les commandes suivantes :

```
# spécifie l'environnement virtuel Python à Jupyter
ipython kernel install --user --name=ml-venv
# lance l'application Jupyter
jupyter-lab
```

3 Premier modèle AutoEncoder

Puis créez un nouveau notebook nommé « `autoencoder_mnist.ipynb` ». L'objectif est de proposer un premier réseau de neurones de type AutoEncoder pour afin d'encoder différents caractères manuscrits.

Pour rappel, la base de données `Mnist` sur laquelle nous allons travailler est disponible via Keras :

```
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
print("x_train : ", x_train.shape)
```

Tâche 1 : Comme dans le TP précédent, proposez une normalisation des données par la valeur maximale d'une images 8 bits, soit 255 afin que chaque valeur de pixel soit $\in [0, 1]$. Cette normalisation doit être faites sur les deux ensembles d'entrées : apprentissage et validation.

La dimension des données d'entrée à un modèle sera définie de la manière suivante :

```
input_img = keras.Input(shape=(784,))
```

Tâche 2 : Créer maintenant un modèle `Sequential` qui servira à encoder les données d'entrée (réduire leur dimension).

- Le paramètre déterminant la taille de l'espace latent z sera initialisé à 2 ;
- Il sera composé de 3 layers Dense : 128, 64, et un de la taille la réduction attendue (l'espace latent z) ;
- Les fonctions d'activation pour chacun des layers d'encodage seront des ReLu ;
- On nommera ce modèle `encoder`.

Tâche 3 : Créer maintenant un modèle `Sequential` qui servira à décoder les données depuis la dimension de l'espace latent z .

- Il sera composé de 3 layers Dense : 64, 128 et 784 ;
- Les fonctions d'activation pour chacun des layers cachés de décodage seront des ReLu, le layer de sortie aura lui une fonction d'activation Sigmoid pour borner les sorties dans un intervalle $[0, 1]$;
- On nommera ce modèle `decoder`.

Pour apprendre le modèle dans son intégralité et propager l'erreur de la sortie attendue, il est possible de combiner les deux modèles (c'est-à-dire à la fois d'encoder et décoder) :

```
autoencoder = keras.Model(input_img, decoder(encoder(input_img)))
```

Tâche 4 : Proposez un apprentissage de ce modèle avec un espace latent $|z| = 2$.

- Le modèle sera compilé avec un optimizer de type *adam* et une fonction de perte de type *binary_crossentropy* ;
- On utilisera 30 epochs et un batch size de 256 pour réaliser cet apprentissage ;
- On proposera des données de validation aux modèles.

Il sera possible d'exploiter après apprentissage les deux modèles indépendamment :

```
encoded_imgs = encoder.predict(x_test) # z latent space
decoded_imgs = decoder.predict(encoded_imgs)
```

Vous devriez obtenir des images reconstruites proches de :

Aussi, il vous sera possible de visualiser l'espace latent z après un encodage des images :

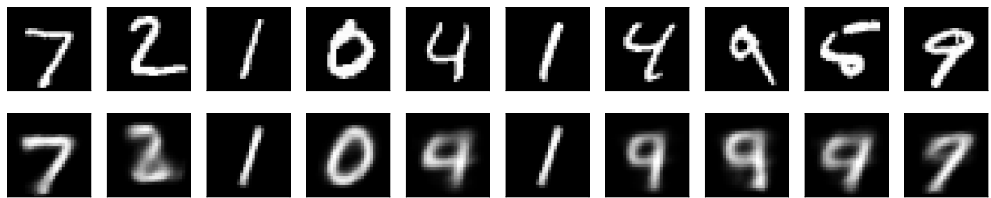


FIGURE 1 – Échantillons d'images reconstruites de la base de test

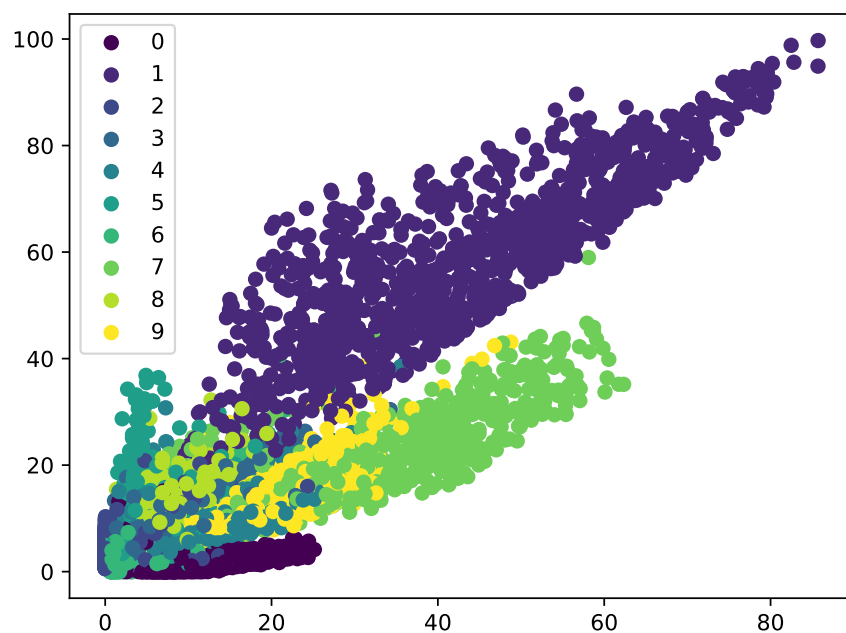


FIGURE 2 – Espace latent $|z| = 2$

Questions

- Quels sont les performances d'encodage et de restitution de ce modèle ?
- Quelles sont pour vous les limites actuelles du modèle ?

4 Vers un espace latent plus complexe

Nous allons maintenant chercher à améliorer les performances de restitution de notre modèle. Pour cela, nous allons jouer avec le paramètre z , déterminant la taille de l'espace latent.

Tâche 5 : Proposez un modèle autoencodeur avec un espace latent $|z| = 3$

- Les paramètres d'apprentissage seront les mêmes que précédents ;
- Le modèle apprendra sur la même base d'apprentissage également.

Avec ce nouveau modèle, vous devriez obtenir des reconstructions proches de :

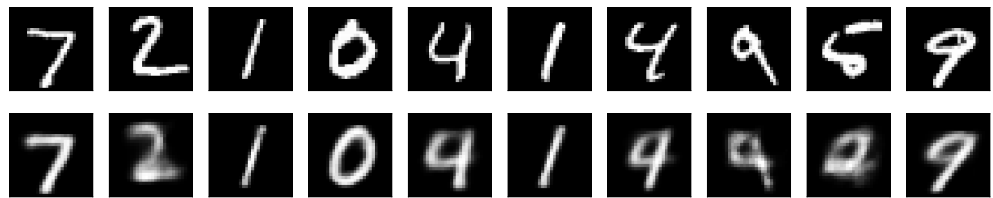


FIGURE 3 – Échantillons d'images reconstruites de la base de test

Aussi, il vous sera possible de visualiser¹ l'espace latent z après encodage des images :

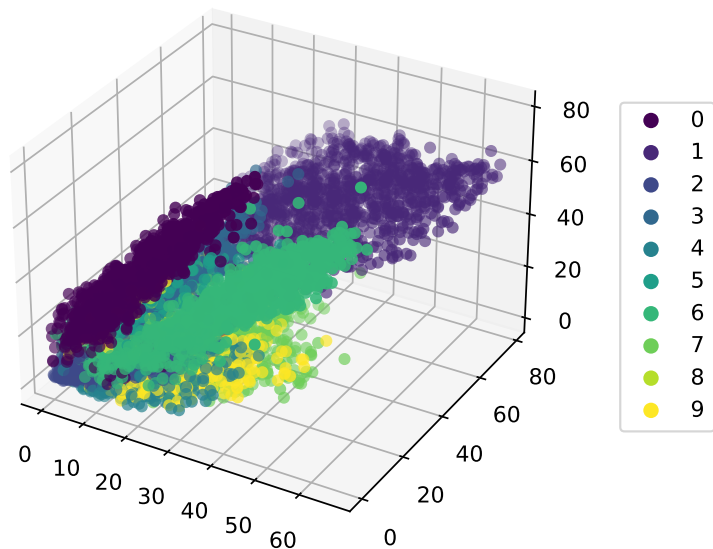


FIGURE 4 – Espace latent $|z| = 3$

Questions

- Que pouvez-vous remarquer des nouvelles reconstructions obtenues ?
- Que remarquez-vous au niveau de l'espace latent ?

Tâche 6 : Proposez maintenant un niveau d'encodage moins drastique, avec notamment des paramètres $|z| = 16$ et/ou $|z| = 32$.

- L'objectif est d'augmenter la capacité d'encodage du modèle ;
- Vous pouvez visualiser les courbes d'apprentissage de ces modèles et vérifier leurs performances d'apprentissage/validation.

1. Si besoin, vous pouvez vous référer sur des [exemples](#) de la documentation de matplotlib.

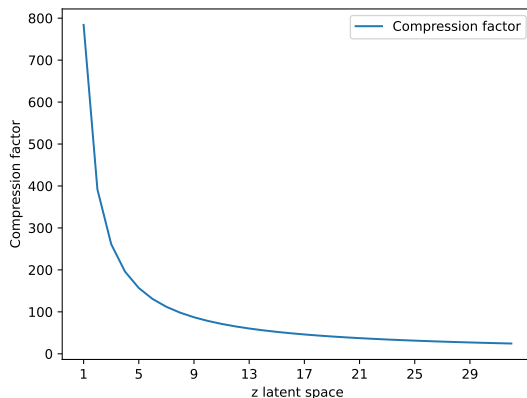
Indications :

- ♦ L'espace latent étant de plus grande dimension, il ne sera pas possible de visualiser sa séparation. Toutefois, il est possible de cibler certaines valeurs de cet espace latent pour s'en faire une idée.

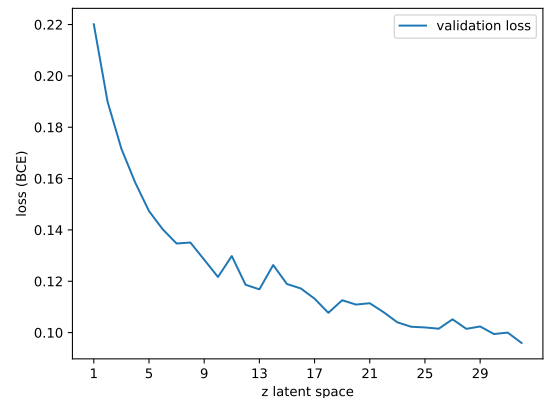
Bonus

Tâche 7 : Proposez d'évaluer quels pourraient être les meilleurs paramètres d'encodage de ce genre de modèle sur ce problème.

- proposez d'entraîner des modèles autoencodeur sur 10 époques avec $|z| \in [1, 32]$;
- Vous pouvez utiliser des indicateurs de performances (fonction de perte sur les données de validation) pour visualiser ces paramètres.



(a) Facteur de compression de z



(b) Évolution de la loss sur l'ensemble de validation

5 Remise des travaux

N'oubliez pas de réaliser un commit de vos travaux. Taguez également votre projet avec le tag « tp4 » et soumettez-le sur le serveur Gitlab. Il fera office de rendu.