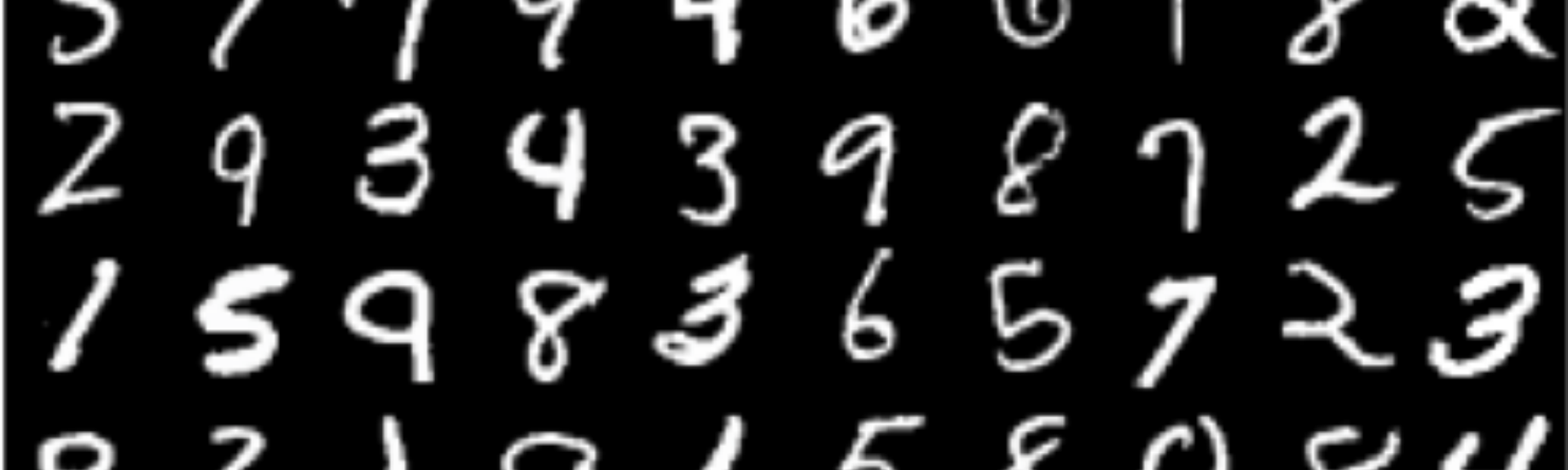
**Chiffres de reconnaissance à l'aide de réseau de neurones convolutifs (CNN)**



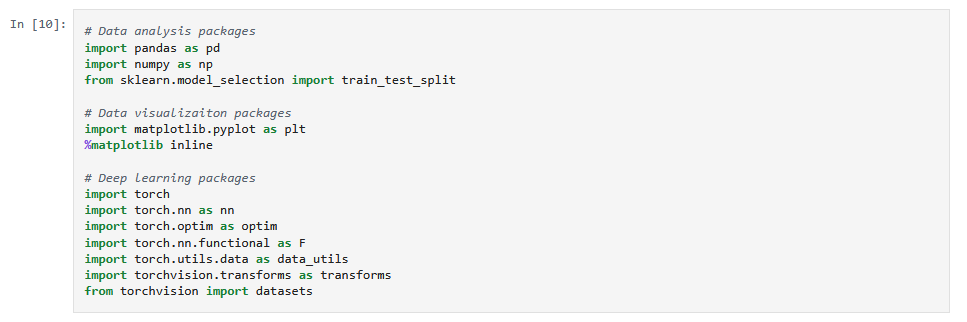
**Description**

L'objectif principal du dépôt est d'identifier les chiffres à partir d'un ensemble de données composé de dizaines de milliers de manuscrits en images, en utilisant des réseaux de neurones à convolution (CNN). Le jeu de données de chiffres manuscrits ([MNIST manuscrite chiffres dataset](https://www.kaggle.com/c/digit-recognizer/data) ) est utilisé pour entraîner et tester le modèle de CNN.

Nous décomposons ce processus selon les étapes suivantes :

* [Étape 1](https://github.com/andikarachman/CNN-Digit-Recognizer/blob/a969dd3a1d1e569fada370c3c03716124bc554cb/#step1): Qu’est-ce qu’un convolutional neural network?
* Étape 2: Importer Des Jeux De Données
* Étape 3: Indiquez les Données de Chargeurs pour l'ensemble de données Image
* [Étape 4](https://github.com/andikarachman/CNN-Digit-Recognizer/blob/a969dd3a1d1e569fada370c3c03716124bc554cb/#step3): Définir Une Architecture De Modèle
* [Étape 5](https://github.com/andikarachman/CNN-Digit-Recognizer/blob/a969dd3a1d1e569fada370c3c03716124bc554cb/#step4): Spécifier la Perte de la Fonction et de l'Optimiseur
* [Etape 6](https://github.com/andikarachman/CNN-Digit-Recognizer/blob/a969dd3a1d1e569fada370c3c03716124bc554cb/#step5): Former et de Valider le Modèle
* [Etape 7](https://github.com/andikarachman/CNN-Digit-Recognizer/blob/a969dd3a1d1e569fada370c3c03716124bc554cb/#step6): Tester le Modèle

Avant de passer à la section suivante, nous avons besoin d'importer tous les paquets nécessaires pour faire l'analyse en appelant le suivant:



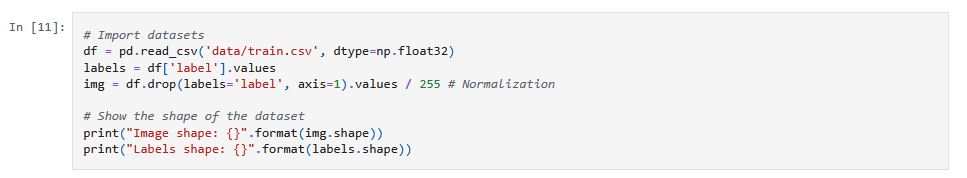
## **Étape 1: Importer Des Jeux De Données**

Un Convolutional Neural Network ([CNN](https://cs.nju.edu.cn/wujx/paper/CNN.pdf)), ou réseau de neurones convolutif en français, est une architecture spécifique de réseaux de neurones profonds utilisée en apprentissage automatique et notamment en **Deep Learning.**

Ces réseaux de neurones ont la capacité d’apprendre à extraire des caractéristiques locales, c’est-à-dire des structures présentes de manière répétée à travers l’image. Au cœur du réseau neuronal convolutif, différentes couches, comme la **couche de convolution,** jouent un rôle central. La convolution permet aux CNNs d’analyser et de comprendre les structures visuelles complexes présentes dans les données d’entrée. Les réseaux de neurones convolutifs sont l’un des modèles les plus réputés pour la classification d’images en raison de leurs performances exceptionnelles.

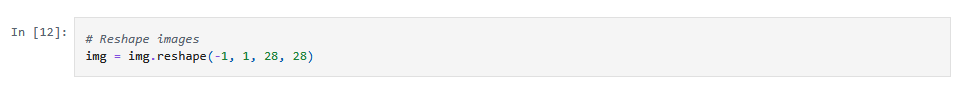
## **Étape 2: Importer Des Jeux De Données**

[MNIST manuscrite chiffres dataset](https://www.kaggle.com/c/digit-recognizer/data) est utilisé pour former et tester le modèle de CNN. Nous réalisons également des niveaux de gris de normalisation afin de réduire l'effet de l'éclairage des différences.



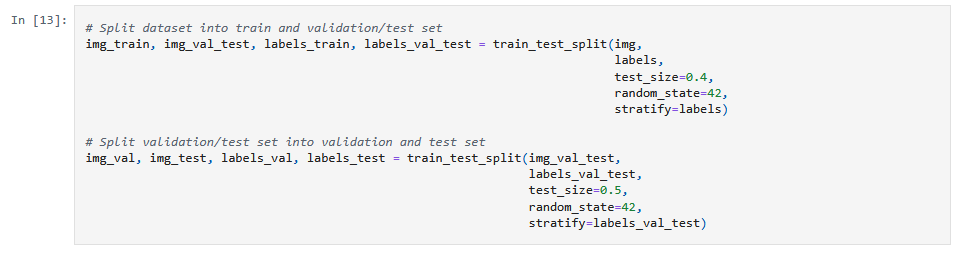
## **Etape 3: Indiquez les Données de Chargeurs pour l'ensemble de données Image**

Les images (28px x 28px) a été stockés dans des pandas dataframe comme 1D vecteurs de 784 valeurs. Nous remodeler toutes les données de 28x28 matrices. PyTorch exige une dimension supplémentaire au début, qui correspondent à des chaînes. MNIST images sont grises à l'échelle de sorte qu'il utiliser un seul canal. Pour les images RVB, il y a 3 canaux, nous avons remanié 784px vecteurs pour 3x28x28 matrices 3D.

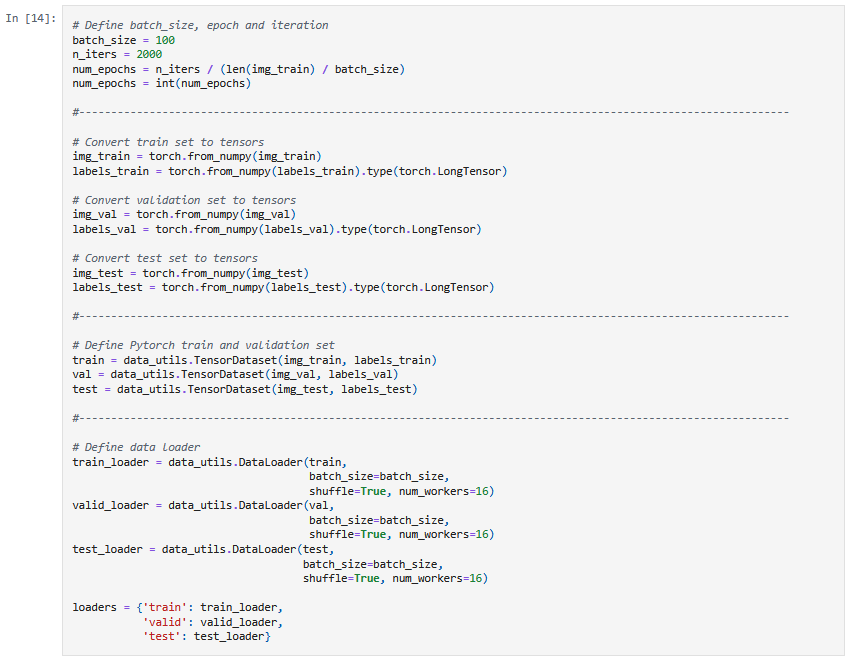


Ensuite, nous avons divisé le train en trois parties :

* 60% de l'ensemble de données de devenir le train
* 20% de l'ensemble de données deviennent l'ensemble de validation
* 20% de l'ensemble de données de devenir le jeu de test



Nous utilisons la cellule de code ci-dessous pour écrire trois [données chargeurs](http://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader) pour la formation, la validation et de test des jeux de données de chiffres d'images.



## **Étape 4: Définir Une Architecture De Modèle**

Dans notre CNN architecture, la première couche a une entrée de la forme de (28, 28, 1) et la dernière couche de sortie 10 classes. Dans les deux premiers convolutifs couches, nous avons choisi de jeu de 32 filtres avec kernel\_size de 5. Après ces couches, les données sont sous-échantillonnées à l'aide d'un max de mise en commun de la couche avec un pas de 1. Les deux prochaines convolutifs couches ont 64 filtres avec kernel\_size de 3. Ensuite, les données sous-échantillonnées à l'aide d'un max de mise en commun de la couche avec la foulée de 2.

Nous avons appliqué abandon de 0.5. Décrochage est une méthode de régularisation, où une proportion de nœuds dans la couche au hasard ignoré (la définition de leurs wieghts à zéro) pour chaque échantillon. Elle tombe au hasard d'une propotion du réseau et des forces du réseau pour apprendre les fonctionnalités de manière distribuée. Cette technique améliore également la généralisation et réduit le surapprentissage.

Nous utilisons des "Rectified Linear Unit " comme notre fonction d'activation. Rectified Linear Unit est utilisé pour ajouter de la non-linéarité du réseau.

Entièrement connecté couche est placée à l'extrémité du réseau. Il combine des caractéristiques locales de la précédente convolutifs couches. Ensuite, le 2ème entièrement connecté couche est destinée à produire de la finale de la taille de sortie, qui prédit des classes.



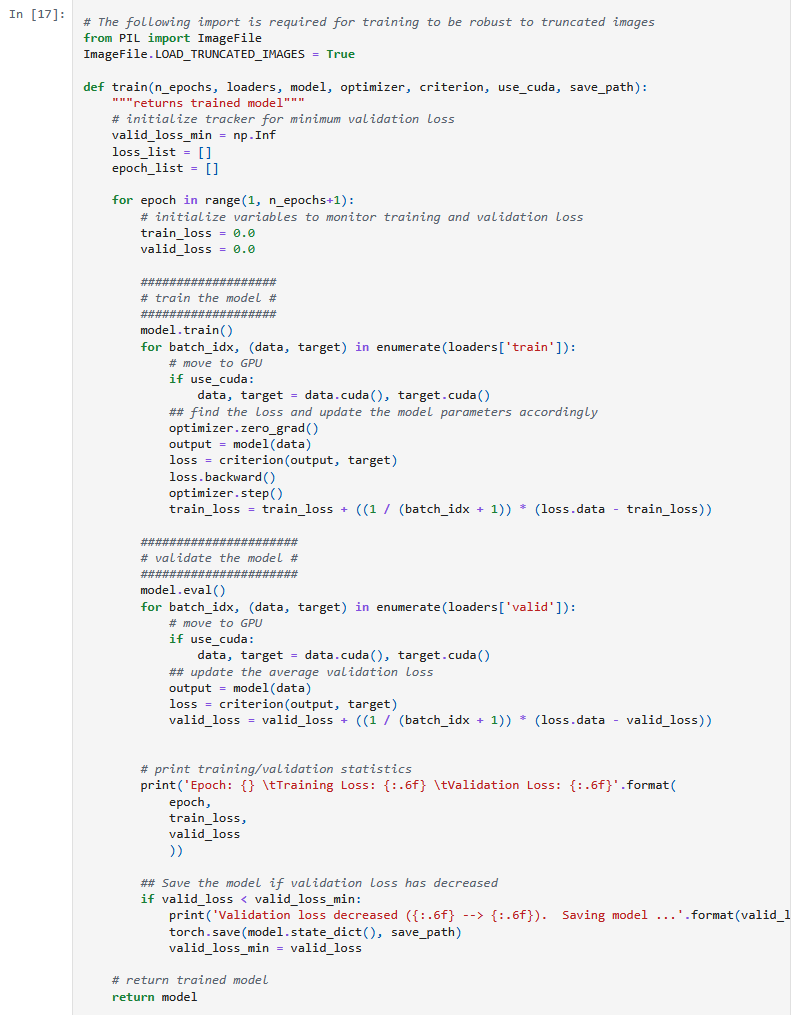
## **Étape 5: définir la Perte de la Fonction et de l'Optimiseur**

Nous utilisons CrossEntropyLoss que notre perte de fonction et RMSprop que notre optimiseur.

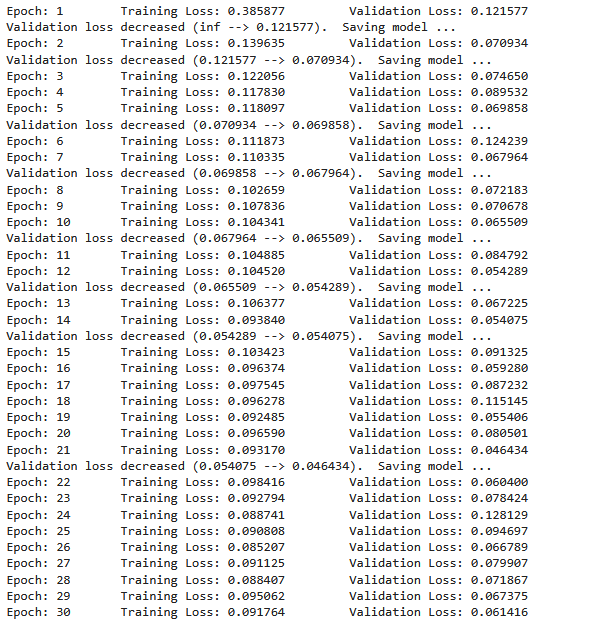


## **Etape 6: Former et de Valider le Modèle**

Nous sauvegardons les paramètres finaux du modèle dans le chemin du fichier'cnn\_digit\_recognizer.pt'.







## **Etape 7: Tester le Modèle**

Nous essayons de votre modèle sur le jeu de données de test. Nous utilisons la cellule de code ci-dessous pour calculer et imprimer le test de la perte et de la précision.





Test Loss: 0.041995

Test Accuracy: 98% (8301/8400)

### **Visualiser L'Échantillon De Test Résultats**

