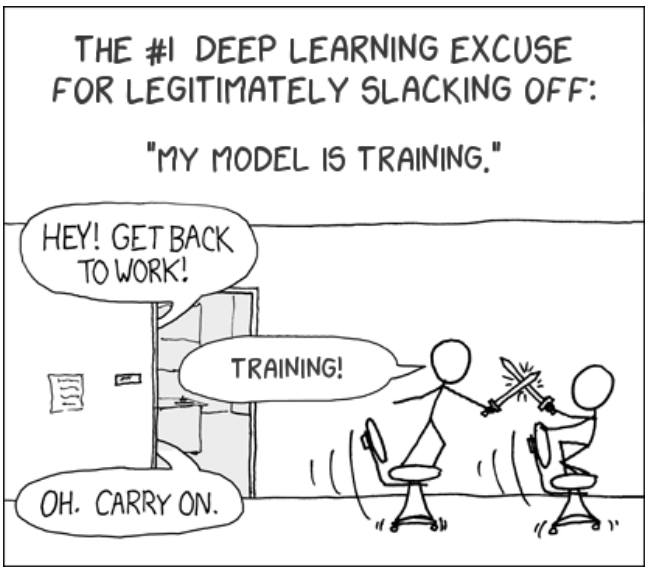
# Handwritten Digits Classification

I want to tell you a neural network joke, but it's deep.

## MNIST by Yann LECUN

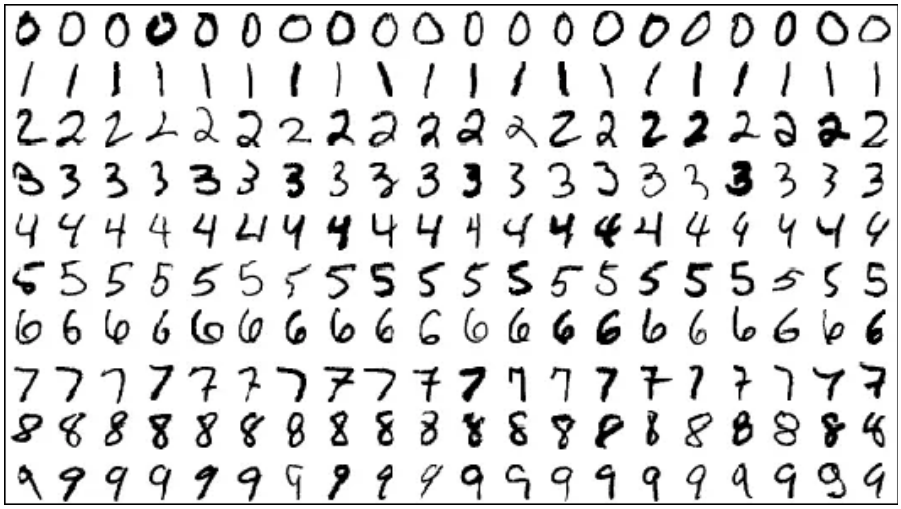



La base de données **MNIST (Modified National Institute of Standards and Technology database)** est une large base de données de **chiffres manuscrits**. Elle regroupe 60.000 images d'apprentissage et 10.000 images de test. Ces images sont en **noir et blanc, normalisées et centrées** pour tenir dans une "boîte" de 28 x 28 pixels.

La **reconnaissance de l'écriture manuscrite** est loin d'être une tâche simple. La base MNIST est devenue un **standard** pour l'**étude de performance des méthodes de reconnaissance de formes**. La collection est très utilisée car elle est issue du monde réel et est formatée de manière à **minimiser les efforts sur le prétraitement**.

Cette base de données a été créée en **"remixant" deux échantillons** des ensembles de données originaux du NIST (Special Database 1 et Special

Database 3). Ces deux ensembles se composent, respectivement, de chiffres écrits par **des lycéens américains** et **des employés du Bureau du recensement des États-Unis**.



Les créateurs de la base de données ont **documenté des méthodes testées** sur celle-ci. Dans leur article original, ils utilisent une **machine à vecteurs de support** pour obtenir un **taux d'erreurs de 0,8%**. La table ci-dessous est une liste de **quelques-unes des méthodes d'apprentissage automatique** utilisées sur le jeu de données et leurs taux d'erreurs, par type de classifieur :

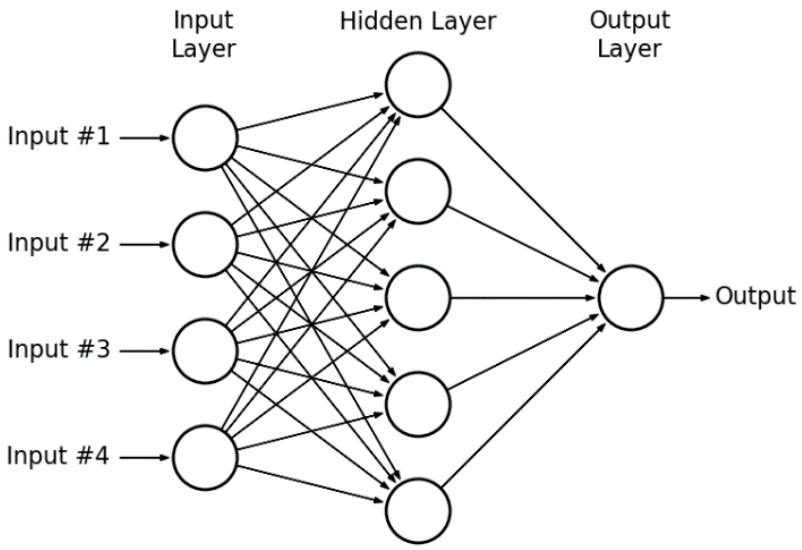
| Type | Classifieur | Error rate (%) |
|---|---|---|
| **Linear Classifier** | Pairwise linear classifier | **7.6** |
| **K-Nearest Neighbors** | K-NN with non-linear deformation | **0.52** |
| **Random Forest** | Fast Unified Random Forests for Survival, Regression, and Classification | **2.8** |
| **Deep neural network (DNN)** | 2-layer 784-800-10 | **1.6** |

| | | |
|---|---|---|
| **Convolutional neural network (CNN)** | 6-layer<br>784-50-100-500-1000-10-10 | **0.27** |

Fascinés par cette problématique de détection de formes, vous vous lancez dans le **développement d'un outil de classification des chiffres manuscrits**. Vous êtes déterminés à battre le **record actuel de 0.18 % d'erreur**, détenu par le Département d'ingénierie des systèmes et de l'information de l'Université de Virginie.

Cependant, vous gardez en tête que **certaines images de l'ensemble de données de test sont à peine lisibles**, même par un humain, empêchant ainsi d'atteindre des taux d'erreurs de test de 0 %.

# Phase 1 : Perceptron Multicouches




**Avant d'ouvrir votre IDE** et de construire plein de PMC super performants, vous **révisez les notions importantes** d'un RNA.

1. Il existe différents types de couches dans un réseau de neurones artificiels. Quelles sont **les types de couches** pouvant **composer** un **Perceptron multicouches** ?

2. Définissez et différenciez les notions d'**Epochs**, d'**Iterations** et de **Batch size**.

3. Qu'est ce que l'hyper-paramètre **learning rate** ? Quelles sont les conséquences d'un learning rate **trop bas** ou **trop élevé** ?

4. Définissez la **Batch normalization** et argumentez son utilisation.

5. Qu'est-ce que l'algorithme d'**optimisation d'Adam** ?

Une fois **sûrs de vos bases**, vous vous lancez dans ce challenge et faites une première contribution à la reconnaissance de chiffres manuscrits en utilisant Keras :

6. Explorez et testez différentes combinaisons d'**architectures** et d'**hyper-paramètres** du PMC. Prenez soin de comparer vos modèles et notez les meilleurs résultats.

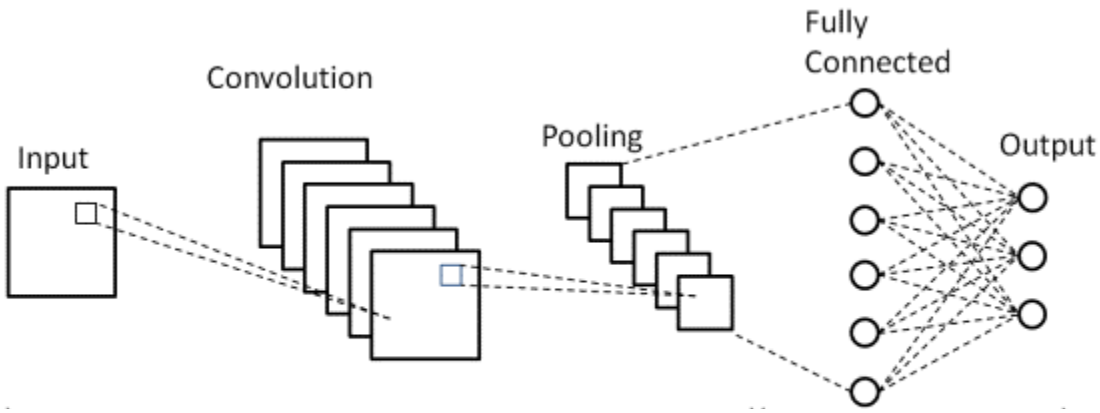
7. Après l'utilisation d'une **couche dense**, les **données sont transformées**. Cela à souvent pour effet de produire des **valeurs totalement dispersées**. Remédiez à ce problème en ajoutant une **couche de normalisation**.

8. Surveillez le **surapprentissage de vos modèles** en visualisant la loss en fonction des **epochs**. Qu'est ce que le **Early stopping** ? S'il y a du surapprentissage, utilisez des **couches de régularisation**.

9. **Évaluez** vos modèles avec différentes **métriques de classification** (matrice de confusion et rapport de classification).

10. **Concluez sur cette première tentative**. Quel est le **modèle** construit générant le **taux d'erreurs le plus bas** ?

## Phase 2 : Réseau neuronal convolutif




En réalisant des recherches sur les réseaux de neurones artificiels, vous tombez sur les super **réseaux de neurones convolutifs**. Les **CNN** désignent une sous-catégorie des RNA et sont à ce jour un des **modèles de classification d'images** réputés être **les plus performants**.

1. Réalisez **une veille** sur les **réseaux de neurones artificiels de type convolutifs**. Quel est l'**architecture typique d'un CNN** ?

2. Donnez le principe de fonctionnement **d'une couche convolutive**. Qu'est ce qu'un filtre de convolution ?

3. Comment un **filtre de convolution** est-il appliqué à une image en entrée ? Qu'est ce qui en résulte ? En quoi est-il utile pour la **détection d'objets** ?

4. Quelle est la **fonction d'activation** utilisée par un CNN ? Pourquoi est-elle **la plus adaptée** pour ce type de réseaux de neurones ?

5. Qu'est ce qui arrive à la **Feature Map** lorsque celle-ci est donnée **en paramètre** à la **fonction d'activation d'un CNN** ?

6. Donnez le **principe** de fonctionnement **d'une couche de Pooling**. Il existe différentes **opérations de Pooling**, citez en au moins **deux**.

7. Quels sont les **avantages** de l'utilisation d'une **couche de Pooling** ?

8. La **dernière couche** d'un CNN est une **couche entièrement connectée**. Expliquez son fonctionnement. Qu'est ce que **reçoit** la couche entièrement connectée ?

9. **Détaillez les raisons** pour lesquelles un **réseau de neurones convolutif** est **préféré** à un réseau de neurones dense pour une **tâche de classification d'images**.

Une fois le **réseau neuronal convolutif compris**, vous vous lancez maintenant dans son **application** sur le MNIST Database avec **Keras**. Etant donné que le CNN est plus adapté à la problématique, vous **espérez avoir de meilleurs résultats**.

10. Explorez et testez **différentes architectures** et **hyper-paramètres** d'un CNN. Prenez soin de **comparer** vos modèles et **notez** les meilleurs résultats.

11. Évitez **une dispersion trop importante** de vos données en utilisant une **couche de normalisation**.

12. Surveillez le surapprentissage de vos modèles en visualisant la loss en fonction des epochs. S'il y a du surapprentissage, utilisez des **couches de régularisation**.

13. Évaluez vos modèles avec les différentes **métriques de classification** (matrice de confusion et rapport de classification).

14. **Concluez** sur cette seconde phase. Quel est le **taux d'erreurs** le plus bas que vous pouvez obtenir ? Avez-vous obtenu de **meilleures performances** qu'avec un MLP ?

## Outil de détection de chiffres manuscrits

---

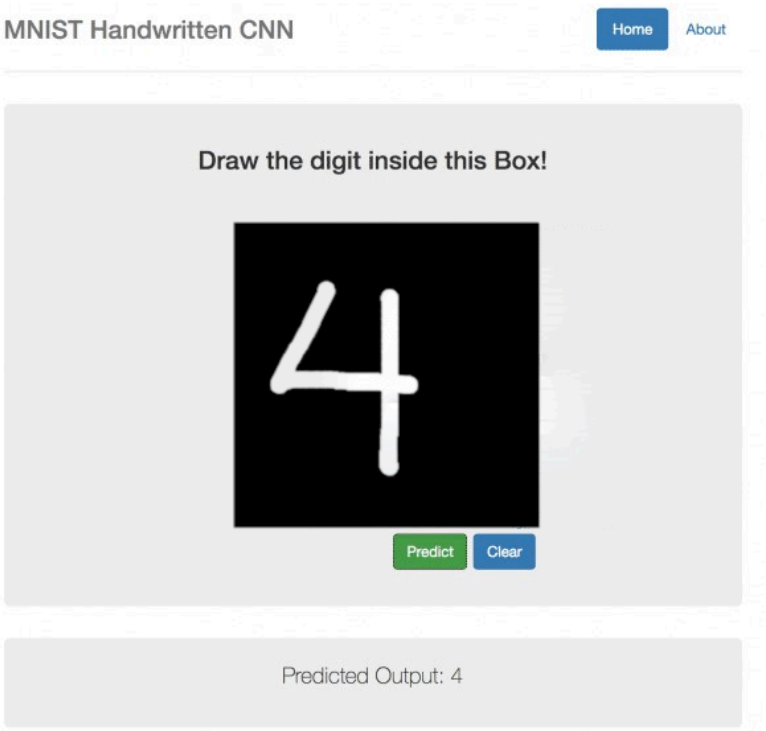
De l'analyse que vous avez réalisée, vous ne garder que le **modèle avec les meilleures performances**. Vous êtes contents et vous vous lancez dans la phase suivante : la **construction d'une interface graphique utilisateur (GUI)** et le déploiement de votre solution dans application web à l'aide du framework **Flask**. La **priorité** de cette application est **son aspect fonctionnel**, l'esthétique n'est que **secondaire**. Sur cette interface, l'utilisateur pourra :

➔ Mettre l'image d'un chiffre manuscrite (format MNIST).

➔ Avoir le résultat de la prédiction du modèle (avec la probabilité des différentes classes).


## (BONUS)...avec du CSS et du Javascript

Pour aller plus loin, vous souhaitez donner la possibilité à l'utilisateur **d'écrire lui-même des chiffres** sur l'interface graphique et de **les envoyer à votre modèle** pour effectuer une **prédiction**. Une telle implémentation nécessite l'utilisation combinée du langage de style CSS et du langage de programmation JavaScript appliqués à votre application web **Flask**. L'interface graphique doit avoir un **cadre** dans lequel l'utilisateur pourra **dessiner un chiffre**, un bouton pour envoyer ce chiffre à votre modèle et afficher la prédiction en temps réel.




## Compétences visées

➔ Apprentissage profond

## Rendu

L'évaluation de ce projet se fera sur deux aspects :

1. Une présentation explicative de votre travail sous forme de diapositives.
2. Un repository github public nommé **digits-classification**, contenant les éléments suivants :
    a. Un **notebook Python** **propre et commenté** (introduction, titres des sections, interprétation des visuels, justification des résultats, conclusion, etc) contenant le procédé de développement de

votre outil, du nettoyage à la modélisation des données, en passant par l'analyse exploratoire. **Pensez à répondre à la problématique.** Vous pouvez avoir au maximum deux notebooks, un pour l'exploration et l'autre pour la modélisation de données.

b. Un **script.py** de votre outil de détection de chiffres manuscrits déployé sur Flask.

c. Un fichier **README.md** présentant le contexte du projet, les données et leur analyse, les algorithmes utilisés et une conclusion sur votre travail. Pensez à inclure la veille réalisée.

## Base de connaissances

- THE MNIST DATABASE of handwritten digits
- Batch Normalization in Convolutional Neural Networks
- Neural Networks Part 8: Image Classification with Convolutional Neural Networks
- Convolutional Neural Networks (CNNs) explained
- Pooling Layer — Short and Simple