# Intelligence Artificielle 1

Cours 4 - Introduction à l'AI - Prog. et Décisions

# Table des matières

1 Introduction à des tâches programmables reliées à l'intelligence artificielle (2/2)
- La représentation graphique des données : résultats d'entraînement et d'évaluation
- La sauvegarde et chargement de modèles ML
  - Sauvegarde
  - Chargement

# Introduction à des tâches programmables reliées à l'intelligence artificielle (2/2)

L'intelligence artificielle (AI) disponible dans les produits est fonctionnelle grâce à de la programmation.

Pour faire de la recherche appliquée en intelligence artificielle, il est nécessaire de programmer pour pouvoir générer des résultats d'expériences.

# Introduction à des tâches programmables reliées à l'intelligence artificielle (2/2)

Tâches courantes (en gras est abordé dans cette diapo) :

- Importation des données, séparation des ensembles de données (train/test sets) et nettoyage des données
- La compilation d'un modèle préétabli avant l'entraînement/l'évaluation (étapes et distinctions brèves, vu en détail au cours AI 3) :
  - La topologie du modèle (couches, etc.)
  - La fonction de perte (Loss function)
  - L'optimisateur (Optimizer)
  - Métrique (Metric)
- L'entraînement d'un modèle à partir de données
- L'évaluation d'un modèle à partir de données
- **La représentation graphique des données et résultats d'évaluation**
- **La sauvegarde et chargement de modèles ML**

# La représentation graphique des données : résultats d'entraînement et d'évaluation

Il est très pertinent de représenter graphiquement les résultats d'entraînement et d'évaluation pour pouvoir faire des conclusions sur la performance d'un modèle (décisions administratives de haut niveau).

# La représentation graphique des données et/ou résultats d'évaluation

Listing: https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/multi-class_classification_with_MNIST.ipynb?hl=en

```
1  # Copyright 2020 Google LLC. https://www.apache.org/licenses/LICENSE-2.0
2  def plot_curve(epochs, hist, list_of_metrics):
3      """Plot a curve of one or more classification metrics vs. epoch."""
4      # list_of_metrics should be one of the names shown in:
5      # https://www.tensorflow.org/tutorials/structured_data/imbalanced_data#defin
6      plt.figure()
7      plt.xlabel("Epoch")
8      plt.ylabel("Value")
9
10     for m in list_of_metrics:
11         # Show accuracy as 0 to 100% for better reading
12         if 'acc' in m:
13             x = hist[m] * 100
14         else:
15             x = hist[m]
16         plt.plot(epochs[1:], x[1:], label=m)
17
18     plt.legend()
19     plt.savefig(list_of_metrics[0] + '.png')
20     plt.show()
```

# La représentation graphique des données : résultats d'entraînement et d'évaluation

NB : Ça diffère d'un problème à un autre

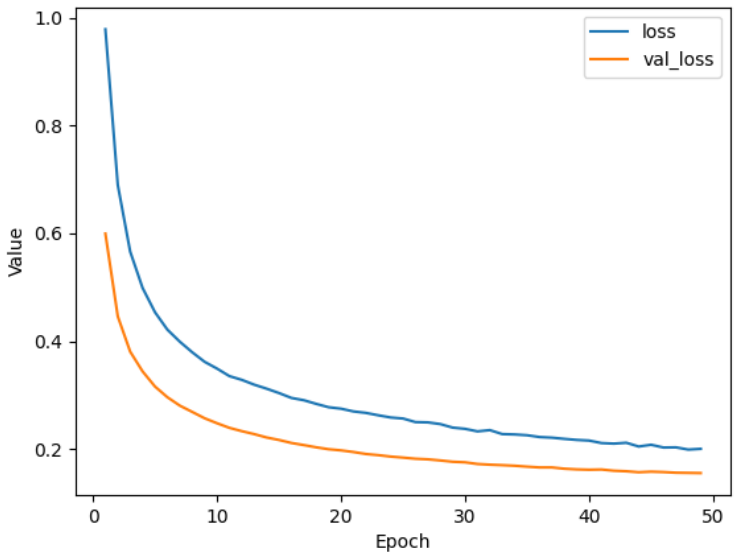


Figure: Entraînement - Perte (loss) - Le plus minimal, le meilleur

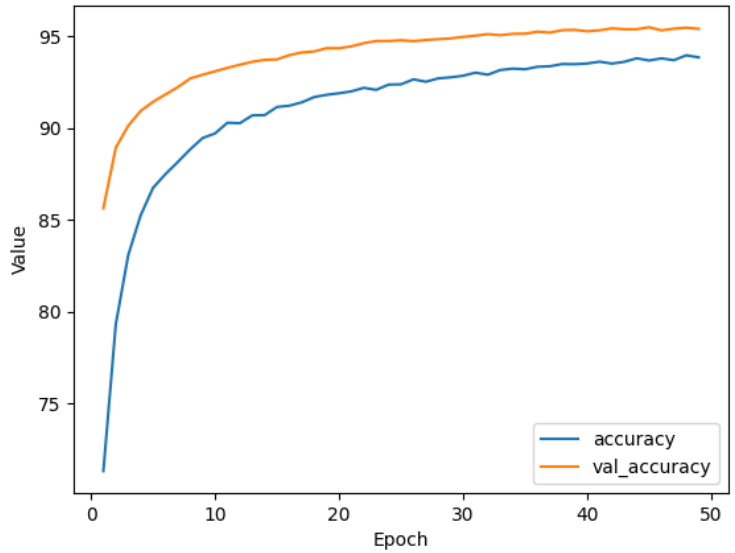


Figure: Évaluation - Précision en % (accuracy) - Le plus haut, le meilleur

# La représentation graphique des données : résultats d'entraînement et d'évaluation - Overfitting/Underfitting

**Underfitting :** Lorsqu'un modèle est trop simple pour capturer la complexité des données parce qu'il n'y a pas assez de données et/ou puisque l'entraînement était trop court au niveau des itérations (epochs).
**Overfitting :** Lorsqu'un modèle est entraîné avec trop de données et/ou pendant trop d'itérations (epochs), il commence à apprendre du bruit et des entrées de données inexactes de notre ensemble de données.

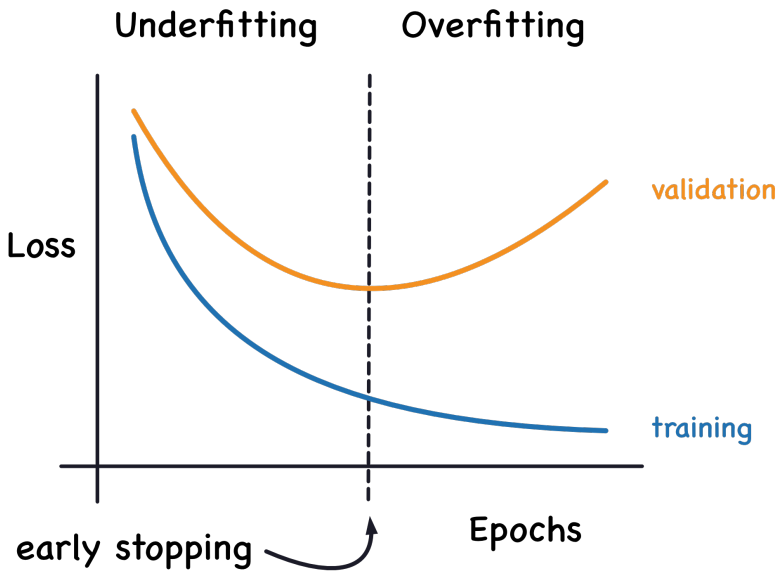


Figure: https://sourestdeeds.github.io/blog/overfitting-and-underfitting/

# La représentation graphique des données : résultats d'entraînement et d'évaluation - Biais/Variance

**Biais (Bias) :** Le biais fait référence à l'erreur due à des hypothèses trop simplistes dans l'algorithme d'apprentissage.

**Variance :** La variance est l'erreur due à la sensibilité du modèle aux fluctuations des données d'entraînement.

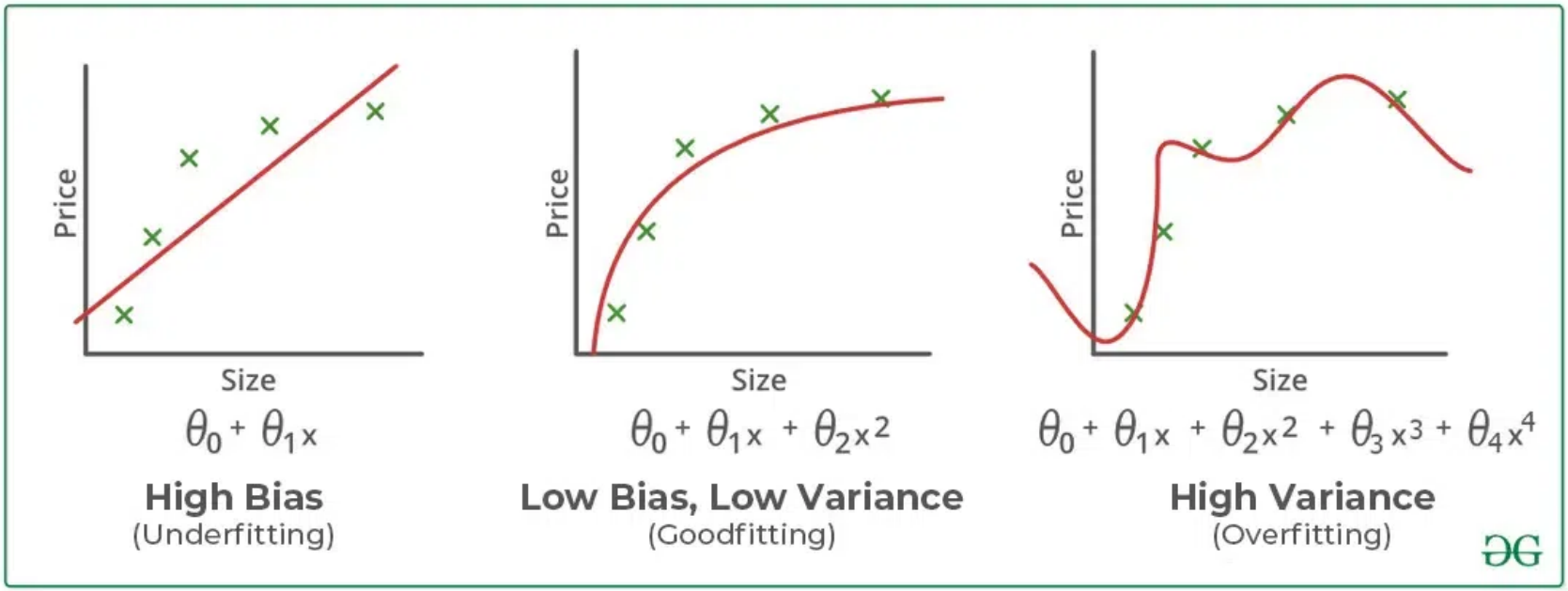


Figure: `https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/`

# La sauvegarde et chargement de modèles ML

Après avoir sauvegardé le modèle, il est possible de l'utiliser sur n'importe quel appareil qui a assez de mémoire sans avoir à l'entraîner à nouveau puisque les poids sont sauvegardés.

Un IoT peut être utilisé! Les modèles chargés demandent seulement l'espace de stockage embarqué et une RAM minimale pour son exécution.

Les prédictions sont quasi instantanées (à moins que le modèle soit vraiment complexe).

# Sauvegarde du modèle

Listing: https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/multi-class_classification_with_MNIST.ipynb?hl=en

```
1  # Copyright 2020 Google LLC. https://www.apache.org/licenses/LICENSE-2.0
2
3  learning_rate = 0.003
4  epochs = 50
5  batch_size = 4000
6  validation_split = 0.2
7
8  # Establish the model's topography.
9  my_model = create_model(learning_rate)
10
11 # Train the model on the normalized training set.
12 epochs, hist = train_model(my_model, x_train_normalized, y_train,
13                            epochs, batch_size, validation_split)
14 # Plot a graph of the metric vs. epochs. (Omitted for simplicity)
15
16 # Eval model
17 my_model.evaluate(x=x_test_normalized, y=y_test, batch_size=batch_size)
18 # Save model for quick reuse in another app (without training again)
19 my_model.save('mnist_trained.keras')
```

# Chargement du modèle

```
1  # Steve Levesque, All rights reserved
2  import tensorflow as tf
3  import numpy as np
4  from matplotlib import pyplot as plt
5
6  def show_mnist_graphic_number(img):
7      img = np.array(img, dtype='float')
8      pixels = img.reshape((28, 28))
9      plt.imshow(pixels, cmap='gray')
10     plt.show()
11
12 # Get data of MNIST
13 (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
14 # Load model
15 new_model = tf.keras.models.load_model('mnist_trained.keras')
16 # Show the model architecture
17 new_model.summary()
18 # Show a prediction
19 img1 = x_train[1]
20 print(new_model.predict(np.reshape(img1, (1, 28, 28))))
21 show_mnist_graphic_number(img1)
```

# Bibliographie


- https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/multi-class_classification_with_MNIST.ipynb?hl=en
- https://stephenallwright.com/loss-function-vs-cost-function/
- https://stephenallwright.com/l1-loss-function/
- https://stephenallwright.com/interpret-mae/
- https://sourestdeeds.github.io/blog/overfitting-and-underfitting/
- https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/