# INF7370 Apprentissage automatique – Hiver 2025

**Professeur : Mohamed Bouguessa**

## Travail pratique 2

**Envoyé le : 7 mars 2025**

**Date de remise : 7 avril 2025 à 9h30**

## Important

1. Le TP est noté sur 22 points.
2. Le travail peut être fait individuellement ou en équipe de deux personnes.
3. **Vous devez travailler seulement avec les deux fichiers : 1_Modele.py ET 2_Evaluation.py**
4. **Ne changé pas les noms de ces deux fichiers.**
5. Inclure votre code dans les endroits spécifiés dans ces deux fichiers.
6. Indiquer votre nom dans les deux fichiers.
7. **Remettre quatre fichiers :**
    a. 1_Modele.py
    b. 2_Evaluation.py
    c. Le fichier Model.keras, donc il faut sauvegarder et récupérer votre modèle avec la meilleure exactitude (*accuracy*) en format .keras.
    d. Votre rapport **en format pdf**. Le rapport doit suivre les directives mentionnées dans l'énoncé.
8. Remettre les fichiers dans un fichier zip identifié à votre nom. N'utiliser pas un nom générique (exemple : TP2).
9. Ne pas remettre les fichiers des images.
10. La date de remise est **le 7 avril 2025 à 9h30. Moodle ferme à 9h30. Aucun travail ne sera accepté à partir de cette heure et ce peu importe les raisons techniques.**

## Énoncé

Le but de ce travail est de développer un réseau de neurones à convolution (CNN) capable de distinguer entre six espèces marines :

1. Baleine
2. Dauphin
3. Morse
4. Phoque
5. Requin
6. Requin-Baleine

Vous devez utiliser le langage Python avec la librairie Keras pour l'implémentation d'un modèle de CNN. Il est recommandé d'utiliser Google Colaboratory comme environnement de développement, car ce dernier offre la possibilité d'utiliser un GPU gratuitement.

**Ensemble de données**

Pour entrainer votre modèle, il faut utiliser un ensemble de **30 000** images de six espèces marines. Le tableau suivant résume la répartition des images en six classes.

| | Baleine | Dauphin | Morse | Phoque | Requin | Requin-baleine |
|---|---|---|---|---|---|---|
| Données d'entrainement | 4 000 | 4 000 | 4 000 | 4 000 | 4 000 | 4 000 |
| Données de test | 1 000 | 1 000 | 1 000 | 1 000 | 1 000 | 1 000 |

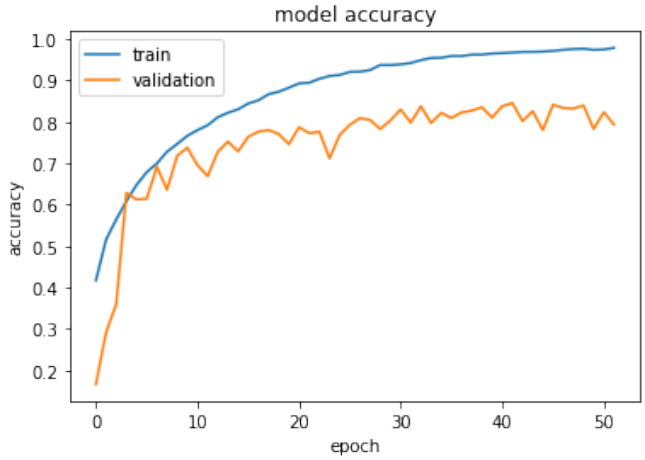
**Tâche 1 : Montage de l'architecture et entrainement du modèle**

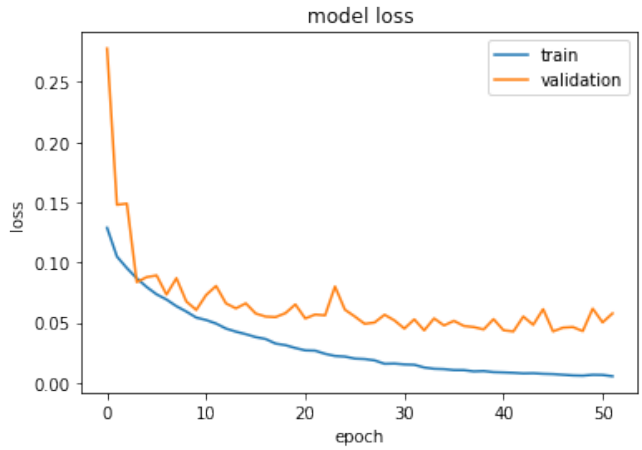
En premier lieu il faut élaborer ***votre propre*** architecture du CNN (le modèle) et l'entrainer sur les images fournies dans le dossier `donnees/entrainement`. Trouver l'architecture qui vous permet d'avoir les meilleurs résultats. Il est à noter que, au final (à la Tâche 2), **l'exactitude (accuracy) minimale** du CNN que vous devez élaborer est de **82% sur les données de test.**

Il est à noter que les données d'entrainement (24 000 images au total) doivent être divisées en deux parties : Training et Validation. Vous avez la liberté de choisir les proportions de chaque ensemble. Ici, je souhaite mentionner que le but derrière la division des données d'entrainement en deux sous-ensembles (Training et Validation) est pour de travailler avec un échantillon qui permet de trouver la meilleure architecture selon l'exactitude (*accuracy*) obtenue via les données de validation.

Dans votre rapport, vous devez indiquer les informations suivantes :

- La taille de l'ensemble de Training ainsi que la taille de l'ensemble de Validation.
- Est-ce que vous avez effectué un prétraitement des données (*data augmentation* par exemple).
- L'optimisateur utilisé avec les paramètres associés à cet optimisateur.
- La taille du lot (*batch size*) d'entrainement.
- Le nombre d'époques (*number of Epochs*) et l'arrêt précoce s'il y a lieu.
- Le nombre de couches utilisées avec le type les paramètres de chaque couche. Par exemple :
    - Le nombre de couches de convolution avec la taille et le nombre des filtres utilisés.
    - Le nombre de couches d'échantillonnage (*pooling*) ainsi que la taille du filtre.
    - Des informations sur la couche complètement connectée : nombre de couches et le nombre de neurones par couche.
- Dropout : Oui/Non?
- Le type des fonctions d'activations.
- Le temps total d'entraînement en minutes.
- L'erreur minimale commise lors de l'entrainement (*Minimum Loss*).
- L'exactitude maximale de l'entraînement (*Maximum Accuracy*).
- La courbe d'exactitude par époque (Training vs Validation) ainsi que la courbe de perte (*loss*). À titre d'exemple, les figures suivantes illustrent nos résultats.




- Justifier vos choix en indiquant les facteurs ayant contribué à l'amélioration de l'entrainement.
- → **Fournir un graphique qui relate l'architecture du modèle que vous avez élaboré**. Voir un exemple à la page 6 de ce document.

**IMPORTANT**

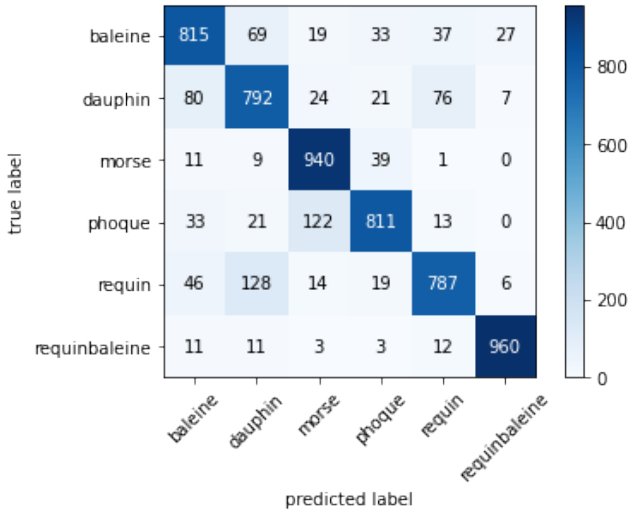
Il faut utiliser le code du fichier 1_Modele.py (de l'exemple MNIST – disponible dans moodle) comme point de départ pour développer votre modèle. Insérer votre code dans les endroits indiqués à cette fin dans le fichier 1_Modele.py. Il faut ajuster l'architecture de votre CNN dans les deux fonctions feature_extraction et fully _connected. Insérer des commentaires dans le code afin d'expliquer vos choix.

**Tâche 2 : Évaluation du modèle**

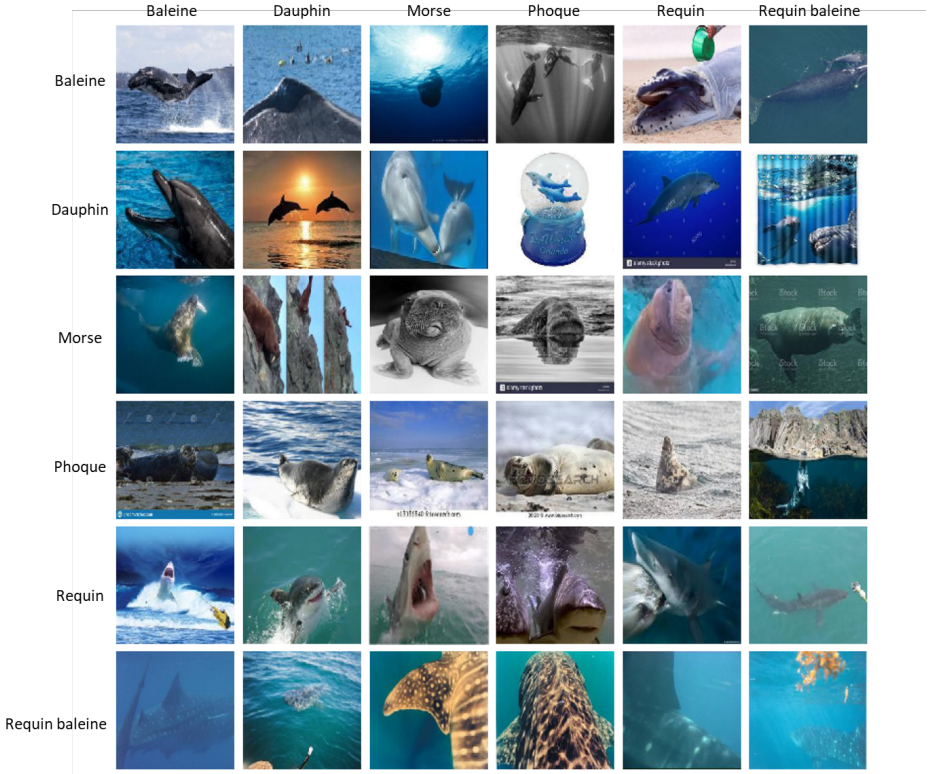
Le but de cette partie est d'évaluer le modèle élaboré à la Tâche 1. À cette fin, il faut utiliser les données de test disponible dans `donnees/test`. C'est un ensemble de 6 000 images contenant 1 000 images de chaque classe.

Dans votre rapport, vous devez indiquer les informations suivantes :

- L'exactitude (*accuracy*) du modèle développé dans la Tâche 1 **sur les données de test**.
    - **Rappel : Il faut atteindre une exactitude minimale de 82% sur les données de test en exécutant 2_Evaluation.py**
- La matrice de confusion. À titre d'exemple, voici une matrice de confusion.




- Extraire une image mal classée pour chaque combinaison d'espèces. Voici un exemple.



**IMPORTANT**

Il faut utiliser le code du fichier 2_Evaluation.py (de l'exemple MNIST) comme point de départ. À noter que l'exemple de MNIST se rattache à deux classes seulement, alors que le problème du TP se rattache à six classes. Un ajustement est donc nécessaire. En plus, dans le code fourni il manque quelques éléments comme la Matrice de confusion, et l'extraction d'une image mal classée pour chaque combinaison d'espèces. Insérer votre code dans les endroits indiqués à cette fin dans le fichier 2_Evaluation.py.

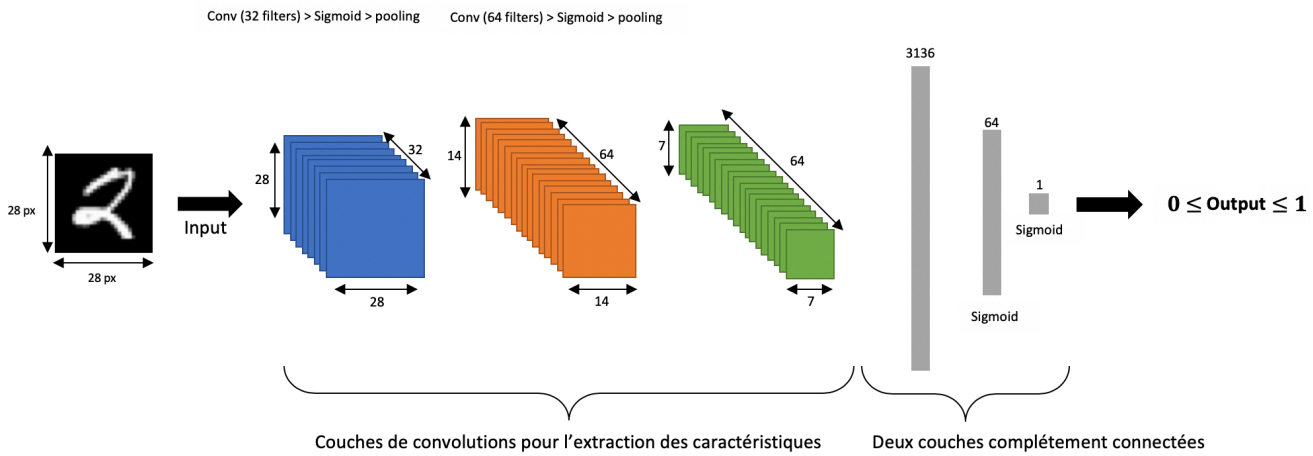
**Tâche 3 : Conclusion**

Conclure votre rapport en discutant, d'une façon générale, les problèmes rencontrés ainsi que les démarches possibles qui peuvent considérés pour améliorer votre modèle.

**Squelettes de code**

Afin de vous faciliter la tâche, nous avons préparé le code pour un problème de deux classes. Il y a deux fichiers : un pour la création du modèle et un autre pour l'évaluation du modèle. Le code est associé à l'élaboration et la validation d'un CNN pour la classification des chiffres manuscrits. Spécifiquement, le modèle est dédié à distinguer entre deux classes : les images associées au chiffre 2 et les images associées au chiffre 7. Ces images ont été extraites de l'ensemble de données MINST qui représente une banque d'image des chiffres manuscrits de 0 à 9.

Nous avons élaboré le code pour bâtir un CNN pour deux classes seulement : les images représentant le chiffre 2 et les images représentant le chiffre 7. Ce code va vous aider à réaliser le TP. Bien évidemment, il faut, l'adapter pour le problème du TP. Veuillez passer en revue les deux fichiers de code et exécutez-les sur Colab. À la page suivante (page 6), vous trouverez un graphique qui relate l'architecture du modèle que nous avons élaboré pour la classification des images '2' et des images '7'. **Utiliser un style de graphique semblable pour élaborer le graphique qui relate votre propre architecture utilisée pour le TP.**




**Architecture globale du CNN utilisée pour la classification des images '2' et '7'.**

## Google Colab

Pour utiliser Google Colab, vous avez besoin d'un compte de Google (Gmail). Vous devez activer le GPU pour votre session avant de lancer le code :

Sous le menu Runtime →Change Runtime → Hardware Accelerator → GPU

Sachez toutefois que les fichiers que vous générez ou téléchargez seront supprimés lorsque la session sera déconnectée. La session ne dure que 6 heures au maximum.

Pour télécharger les images sur Colab, il est recommandé de les télécharger en tant que fichier zip, puis de les décompresser sur Colab. Vous pouvez utiliser la commande suivante pour les décompresser :

```
! unzip donnees.zip
```

Si vous ne voulez pas télécharger vos fichiers à chaque fois, vous pouvez placer les images sur Google Drive et connectez Colab à MyDrive par la commande* :

```
from google.colab import drive
drive.mount ( ' /content /drive ' )
```

Sachez, toutefois, qu'avec cette méthode, l'entrainement sera plus lent, car les fichiers devront être chargés de MyDrive par lot (*batch*).

* Au lancement de cette commande, Colab va vous présenter un lien pour authentifier l'accès à MyDrive. Il faut copier le code généré par Google dans Colab afin de connecter le Drive.