Rapport d'Apprentissage du Modèle

Membres de l'équipe:

Rémy THIBAUT

Quentin DELNEUF

Damien VAURETTE

Généré le 21/01/2025 à 23:22:25

Le modèle utilisé pour cette tâche de classification d'images repose sur l'architecture EfficientNet pré-entraînée, spécifiquement le modèle EfficientNet-B2. EfficientNet est une architecture de réseau neuronal convolutif (CNN) optimisée pour atteindre un compromis optimal entre efficacité et performance. Cette architecture a été choisie pour sa capacité à fournir des résultats précis tout en étant relativement plus légère que d'autres architectures classiques comme ResNet ou VGG.

Le modèle a été ajusté pour cette tâche en modifiant sa couche de sortie. Par défaut, la couche de sortie d'EfficientNet-B2 contient 1408 unités, adaptées pour une tâche de classification sur un large nombre de classes. Pour cette application, cette couche a été remplacée par une nouvelle couche entièrement connectée (_fc) avec une sortie correspondant au nombre de classes dans l'ensemble de données, qui est dynamiquement déterminé à partir des sous-dossiers des images d'entraînement.

Librairies utilisées :

- PyTorch : Utilisé pour la construction, l'entraînement et la gestion du modèle. PyTorch est une bibliothèque flexible et puissante pour le calcul des gradients et l'optimisation des réseaux

neuronaux.

- EfficientNet-PyTorch : Une implémentation de l'architecture EfficientNet, permettant de charger directement des modèles pré-entraînés et de les adapter à des tâches spécifiques de classification.

- PIL (Python Imaging Library) et Torchvision : Utilisées pour les transformations d'images, telles que le redimensionnement et la normalisation, avant d'envoyer les images dans le modèle.

- TQDM : Une bibliothèque utilisée pour ajouter une barre de progression dans les boucles d'entraînement et de validation, facilitant le suivi de l'avancement de l'entraînement.

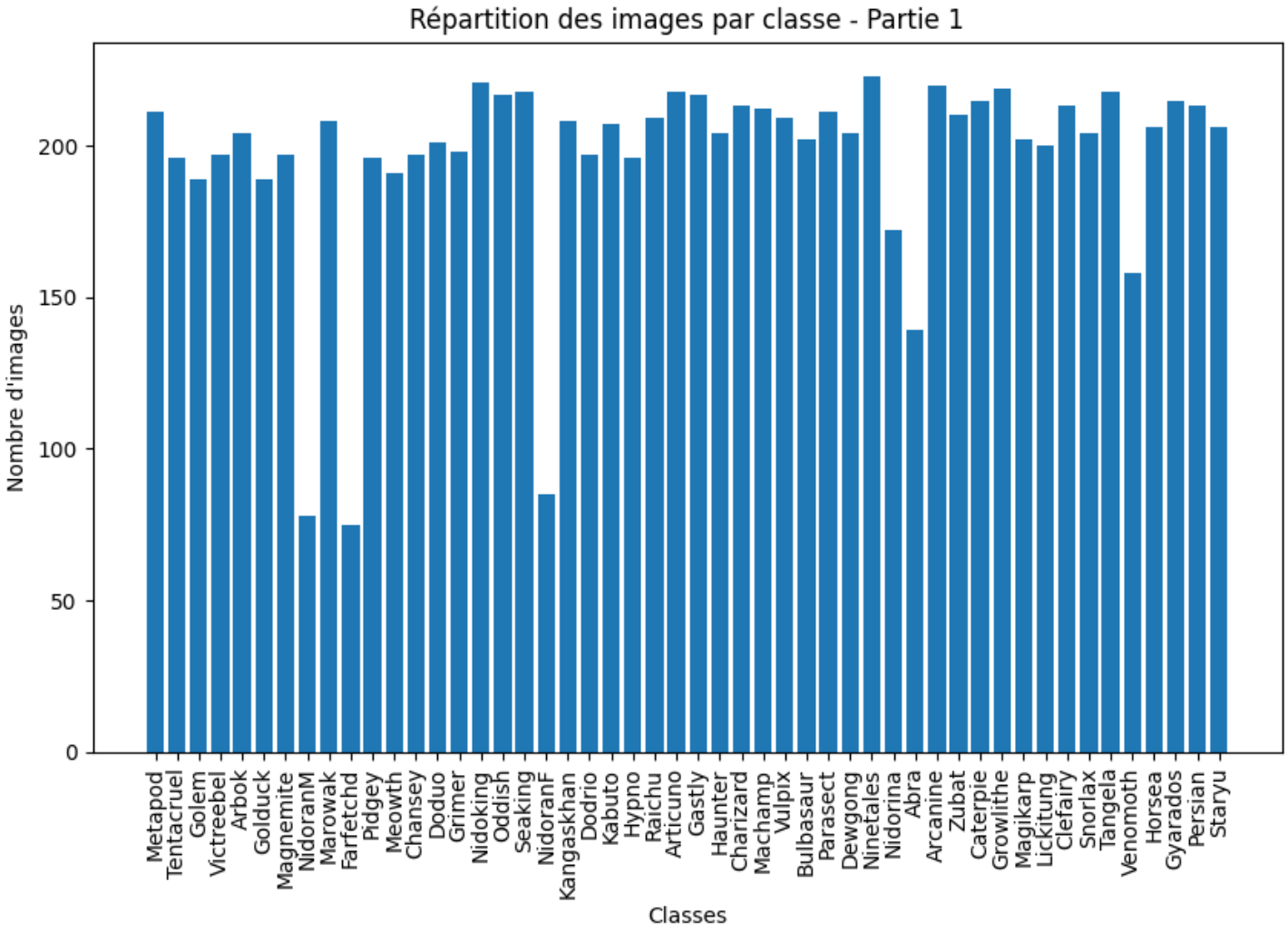
Paramètres de l'entraînement :

- Optimiseur : Adam, utilisé pour la mise à jour des poids du modèle.

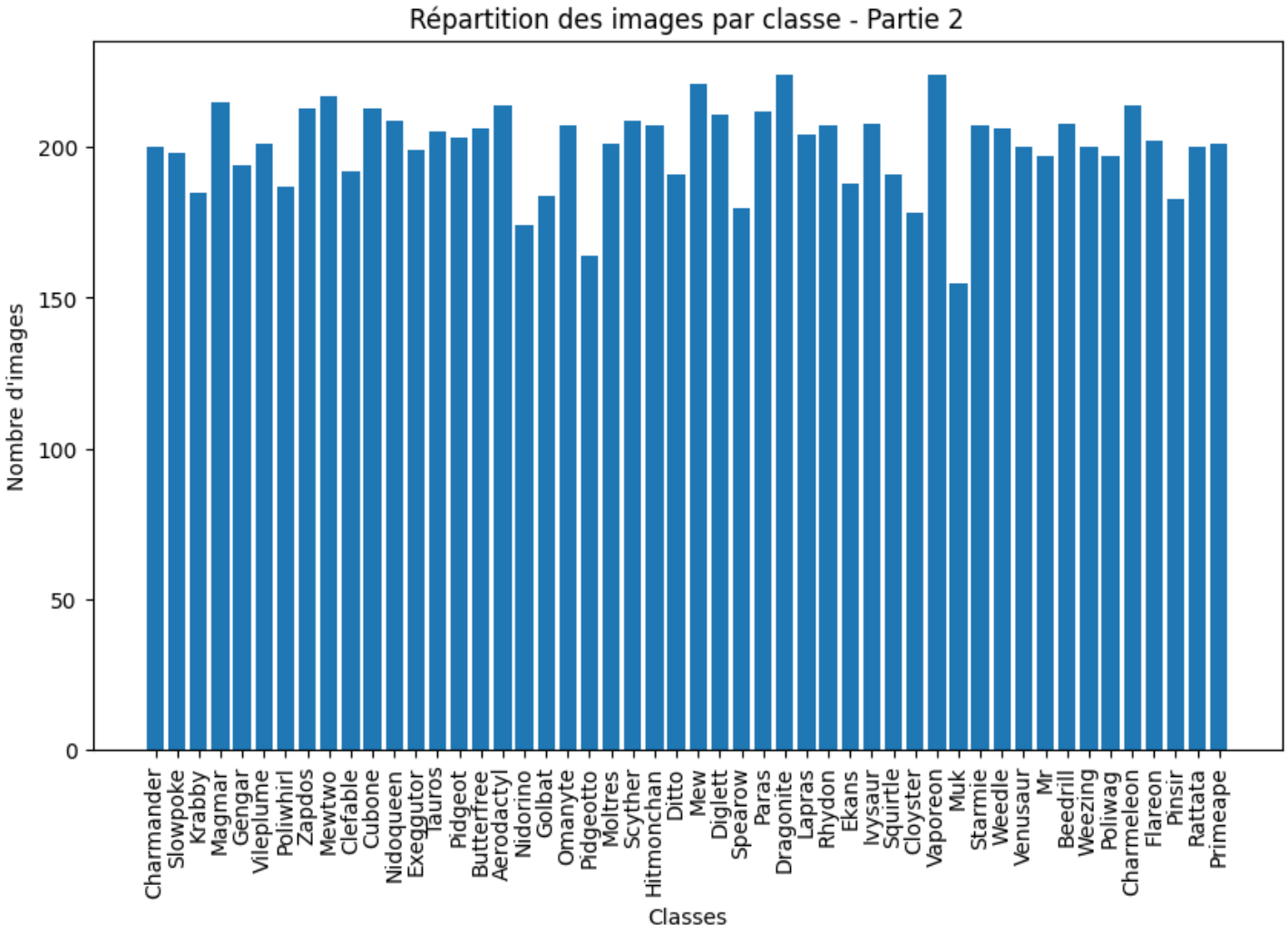
- Critère de perte : CrossEntropyLoss, adapté pour les tâches de classification multi-classes.

- Scheduler : CyclicLR pour ajuster dynamiquement le taux d'apprentissage pendant l'entraînement.

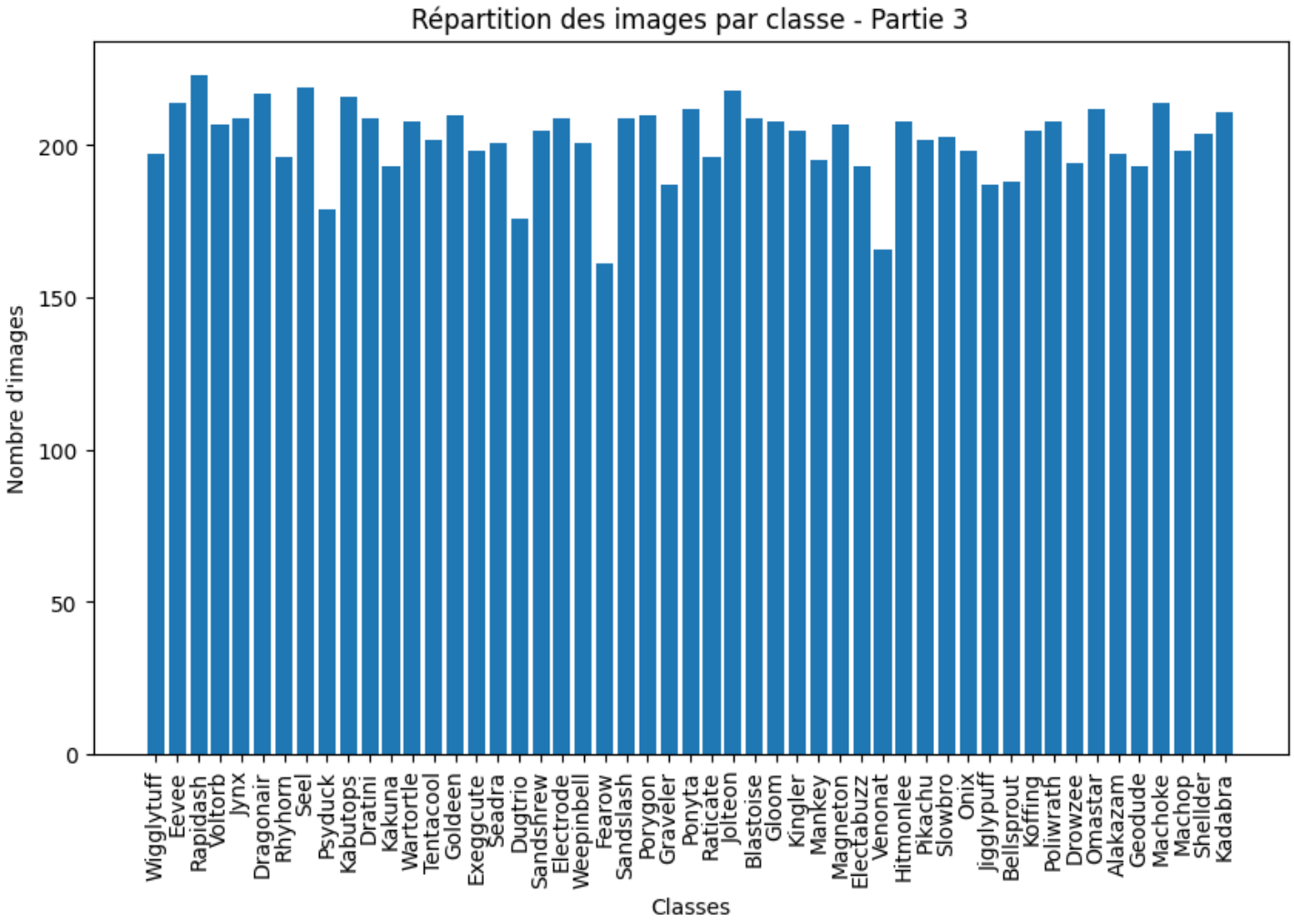
Répartition des Images par Classe - Partie 1



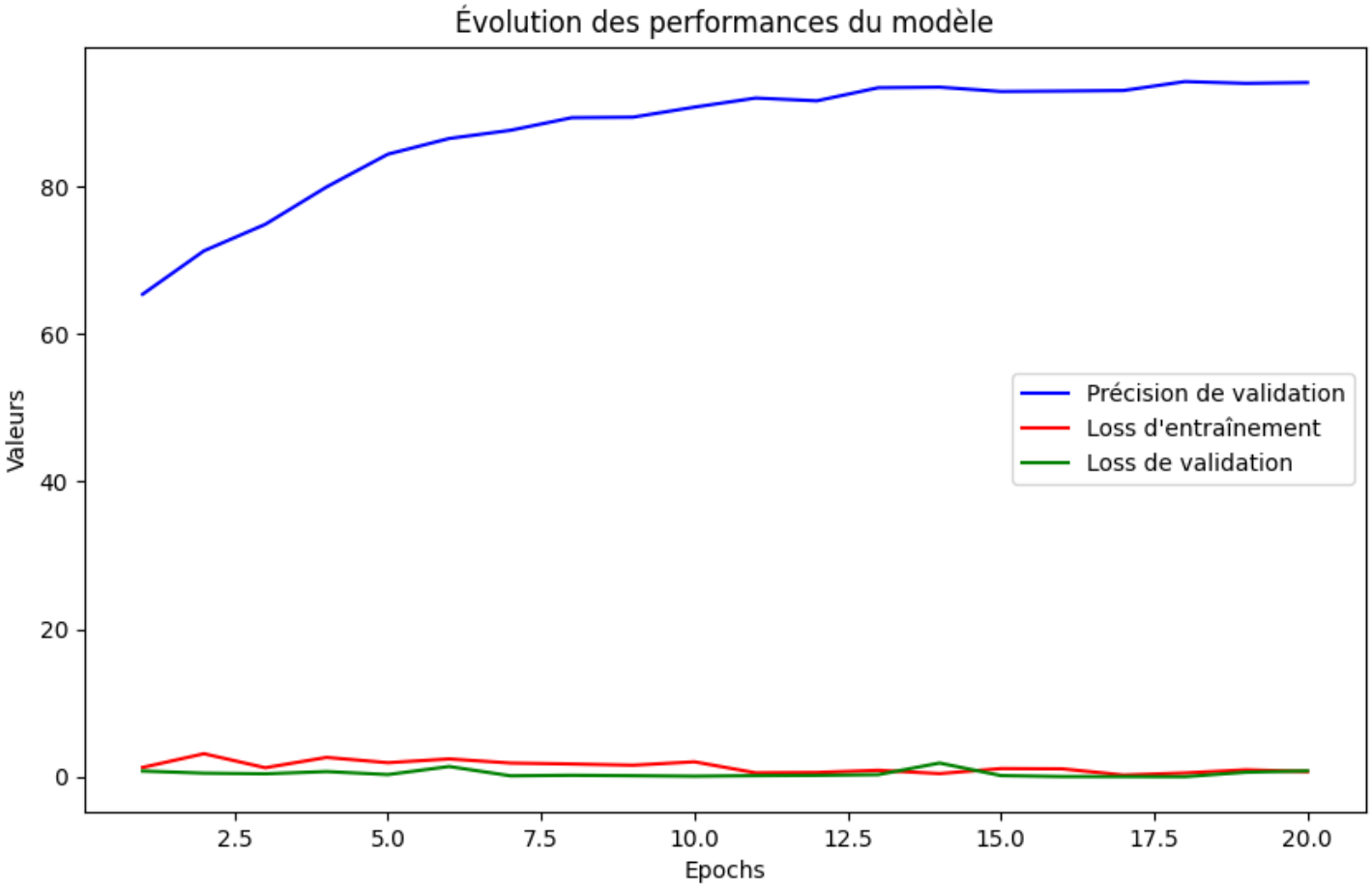
Répartition des Images par Classe - Partie 2



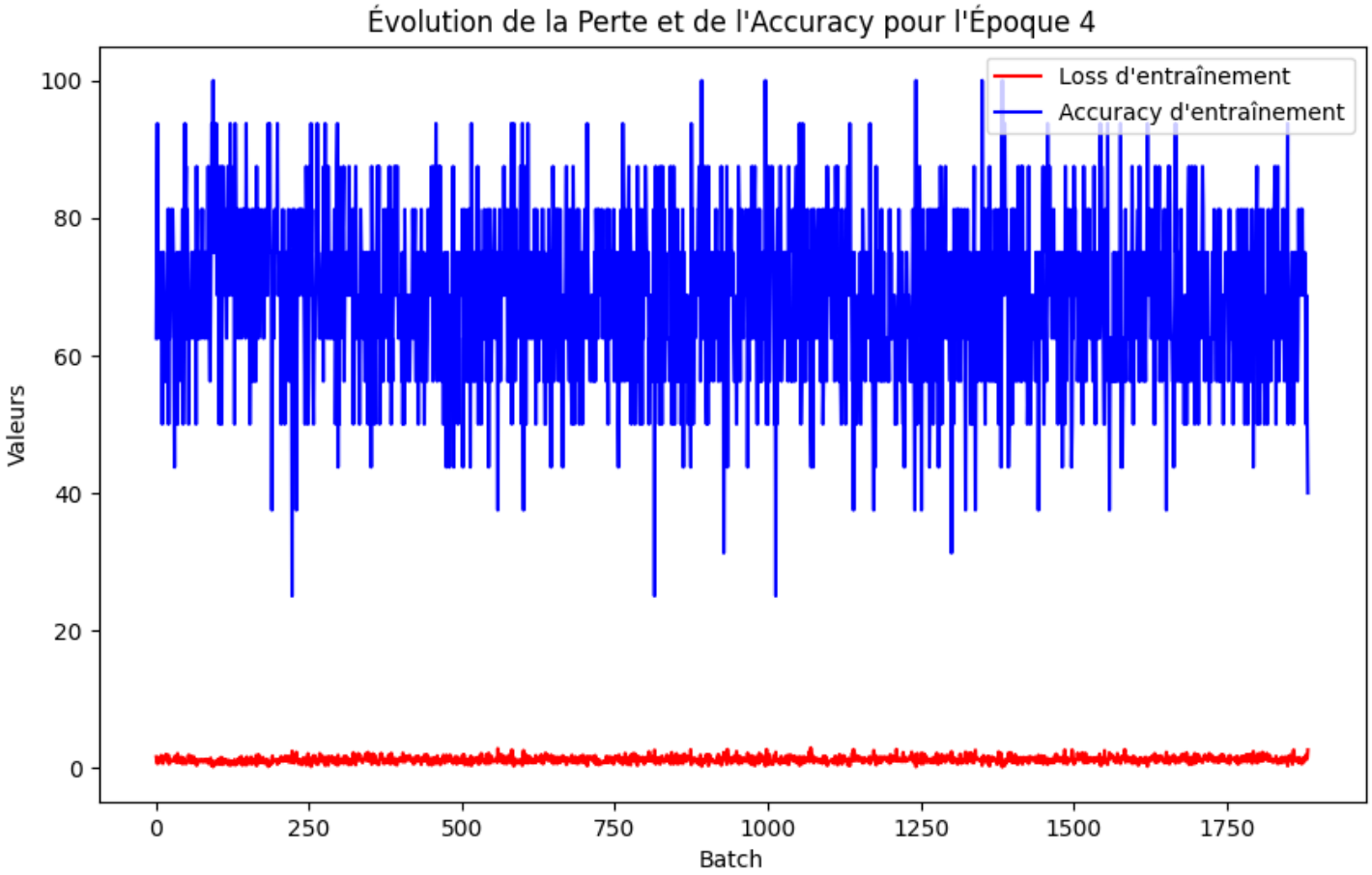
Répartition des Images par Classe - Partie 3



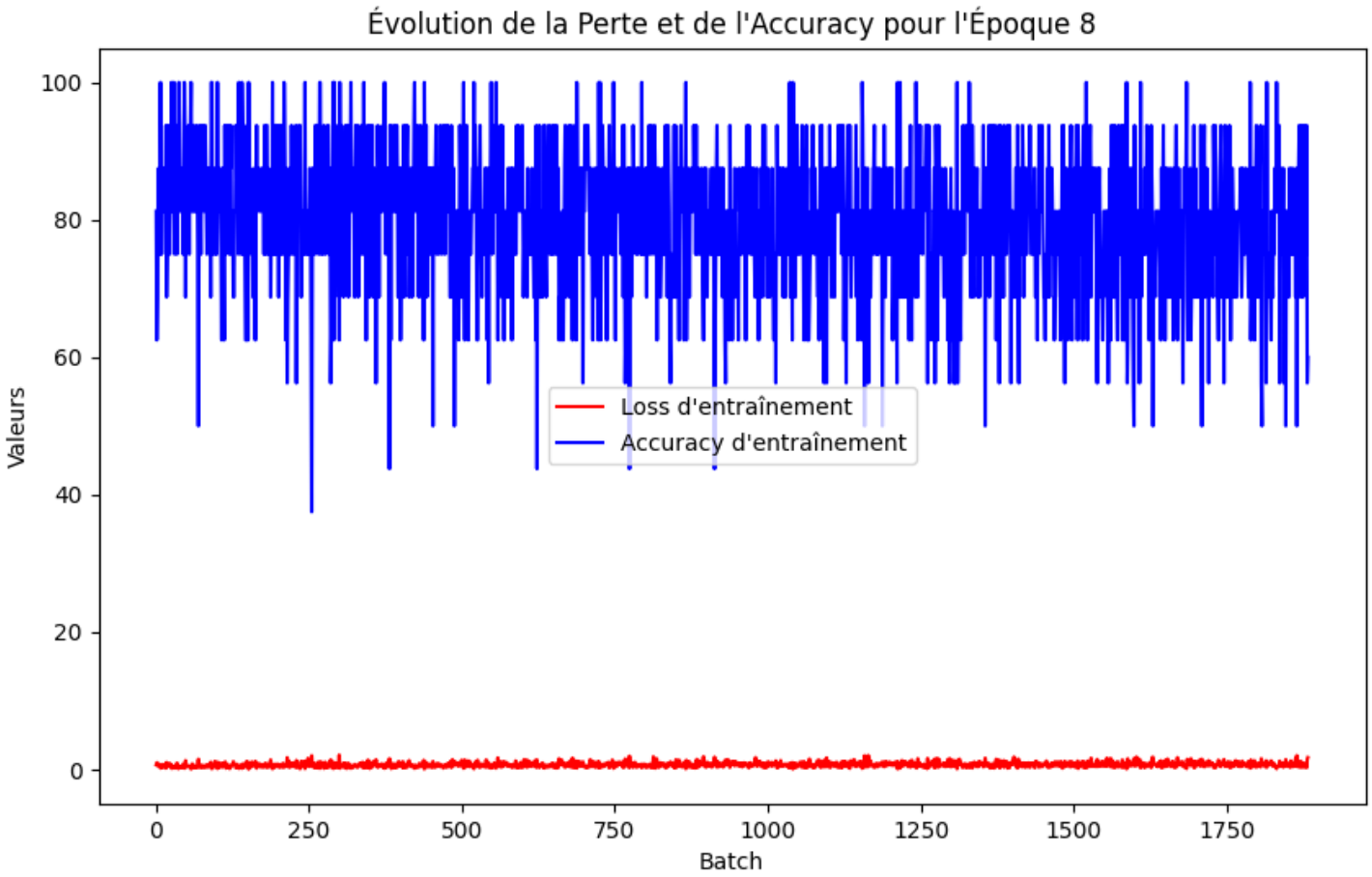
Évolution des Performances du Modèle



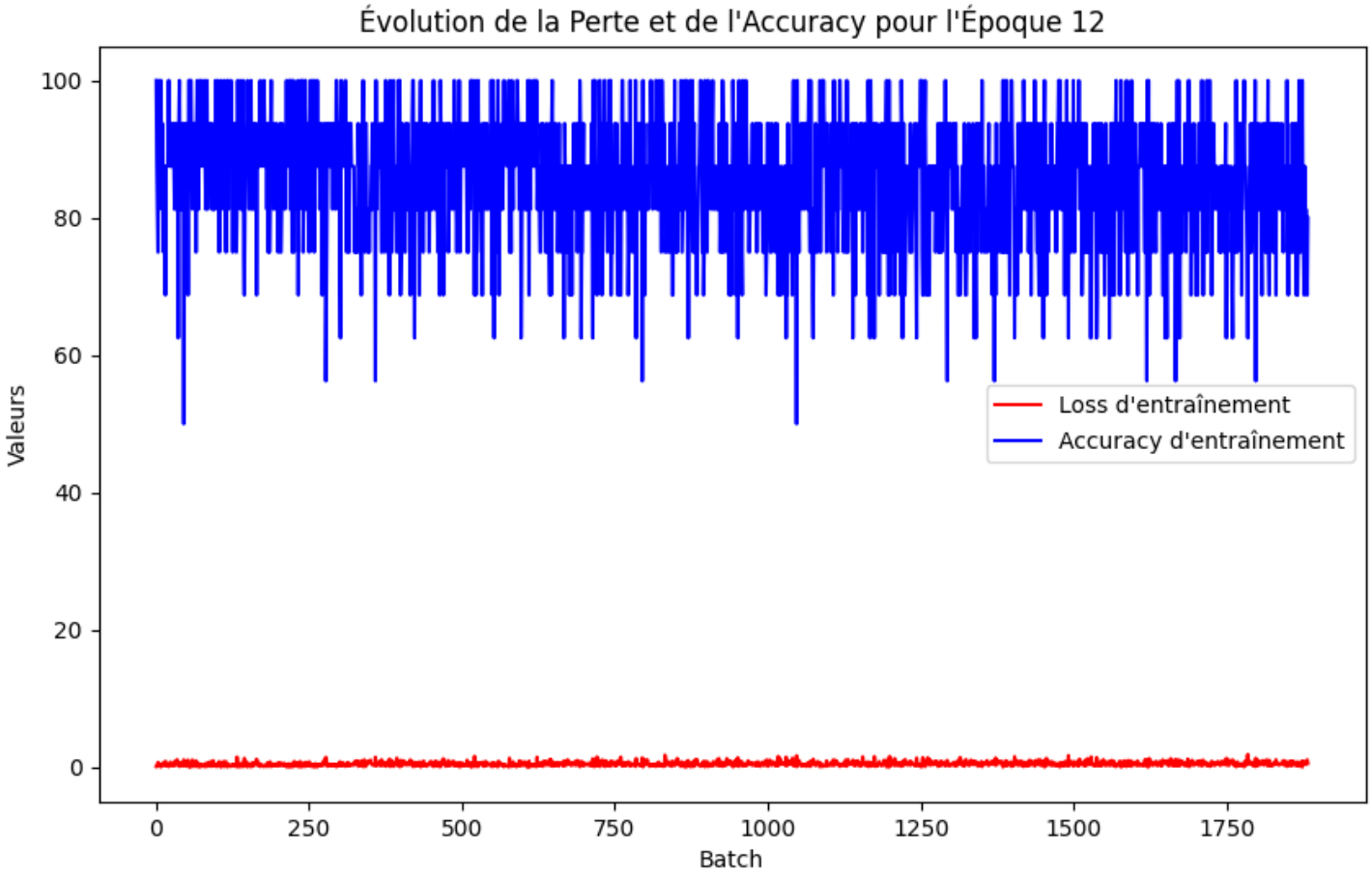
Évolution de la Perte et de l'Accuracy pour l'Époque



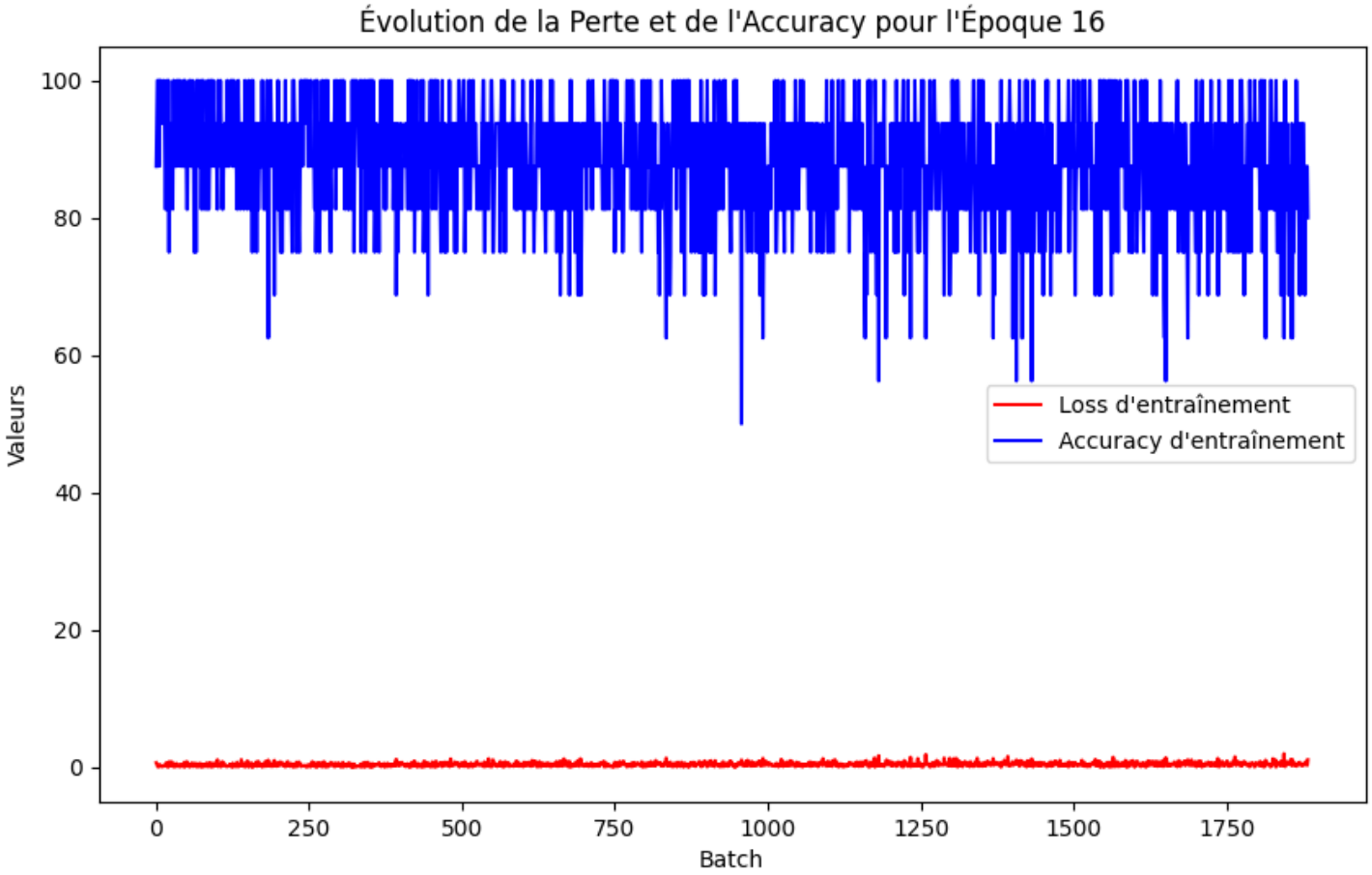
Évolution de la Perte et de l'Accuracy pour l'Époque



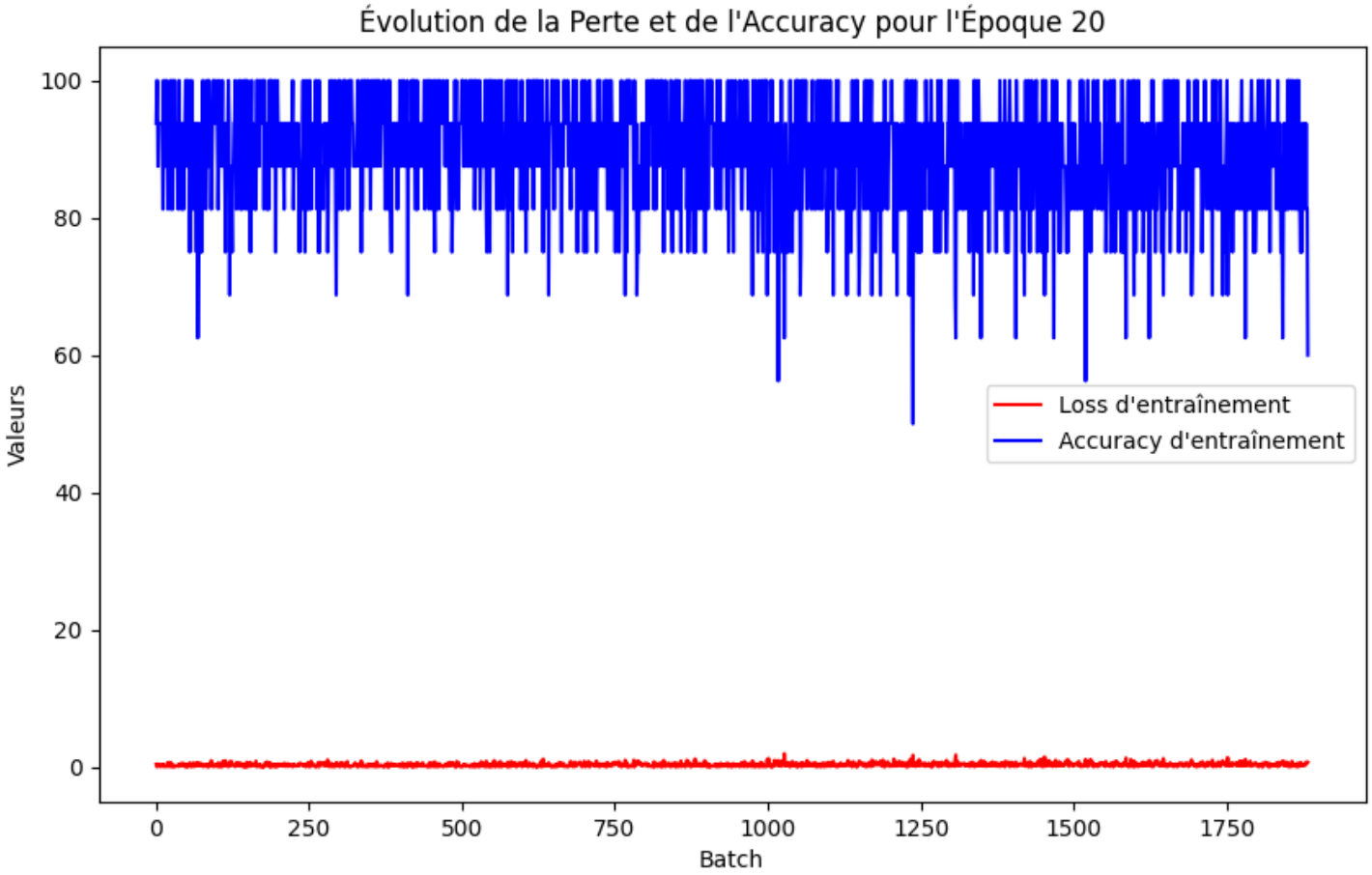
Évolution de la Perte et de l'Accuracy pour l'Époque



Évolution de la Perte et de l'Accuracy pour l'Époque



Évolution de la Perte et de l'Accuracy pour l'Époque



Statistiques d'Apprentissage

Epoque	Perte d'Entraînement	Perte de Validation	Précision de Validation (%)
1.0	1.2573	0.7701	65.40
2.0	3.1180	0.4823	71.28
3.0	1.2176	0.3962	74.87
4.0	2.6188	0.6984	79.94
5.0	1.9002	0.2926	84.39
6.0	2.4104	1.3831	86.51
7.0	1.8536	0.1256	87.62
8.0	1.7269	0.1962	89.33
9.0	1.5608	0.1395	89.41
10.0	2.0210	0.0746	90.76
11.0	0.5264	0.1468	92.00
12.0	0.5740	0.1892	91.62
13.0	0.8695	0.2850	93.40
14.0	0.4252	1.8557	93.47
15.0	1.1044	0.1552	92.89
16.0	1.0698	0.0249	92.94
17.0	0.2272	0.0235	93.02
18.0	0.5069	0.0108	94.23
19.0	0.9313	0.6109	93.99
20.0	0.6959	0.7904	94.09