Rapport d'Apprentissage du Modèle

Membres de l'équipe:

Rémy THIBAUT

Quentin DELNEUF

Damien VAURETTE

Généré le 02/02/2025 à 14:29:30

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.

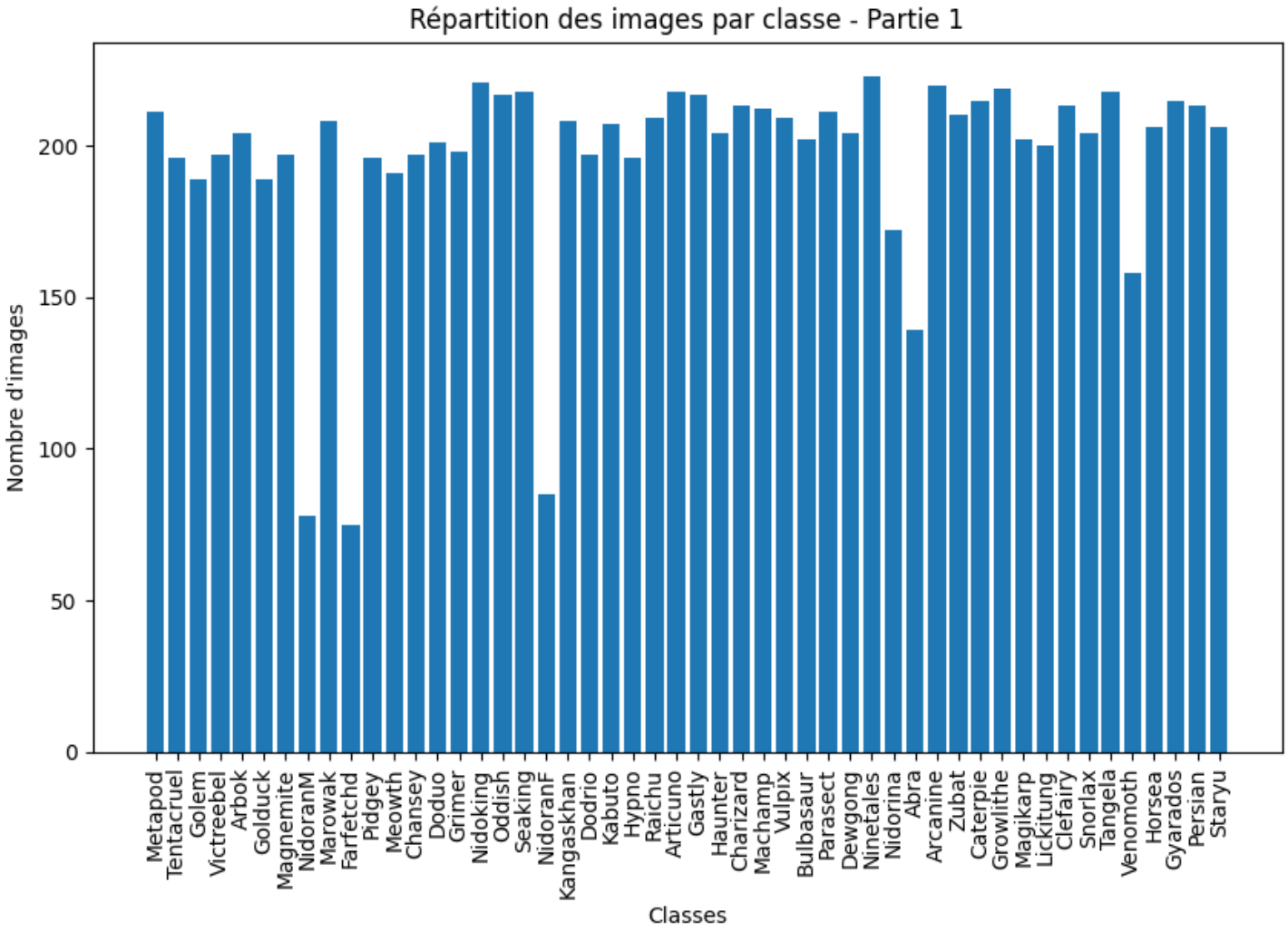
Paramètres de l'entraînement :

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

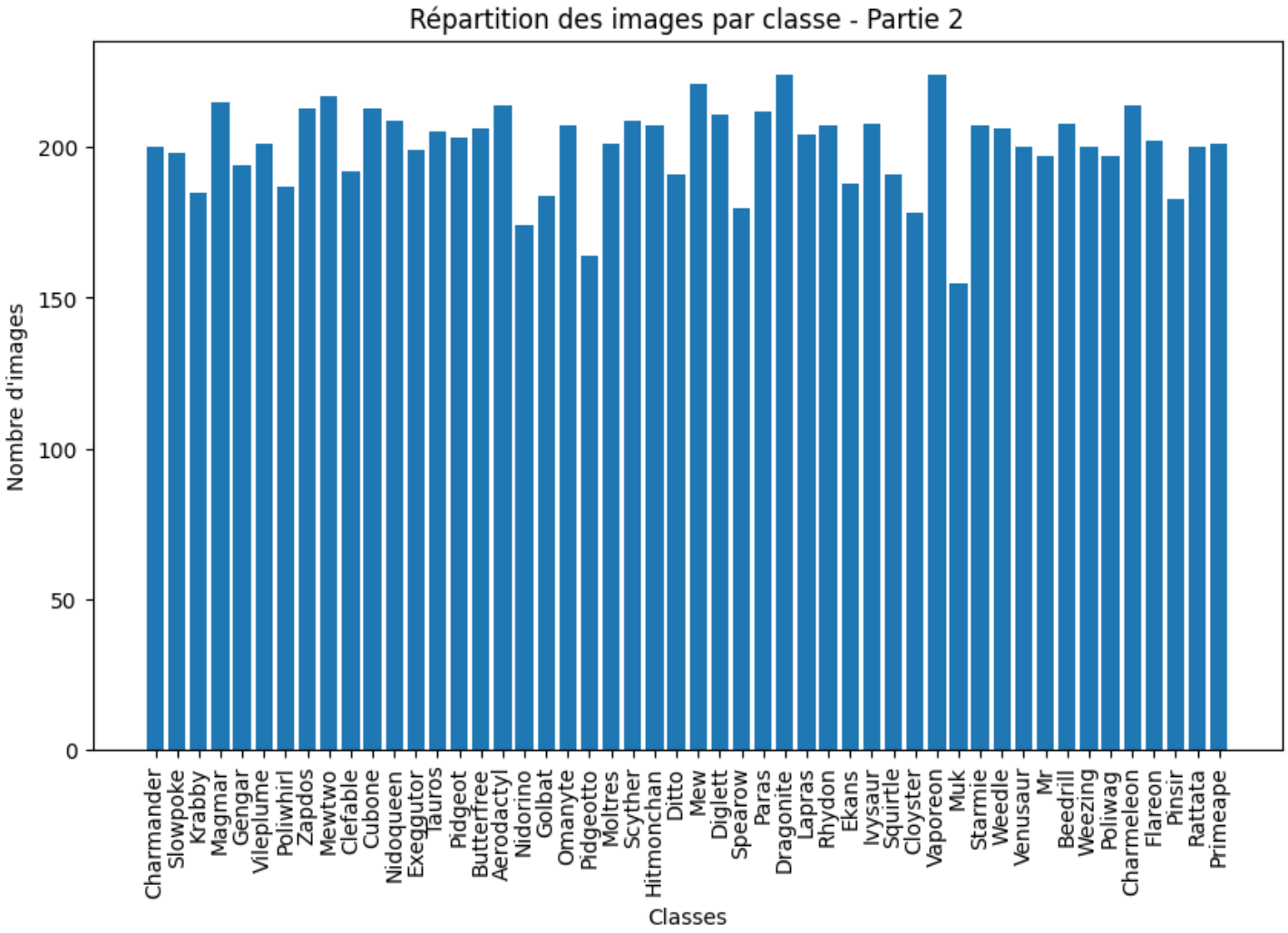
- Critère de perte : CrossEntropyLoss, adapté pour les tâches de classification multi-classes.

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

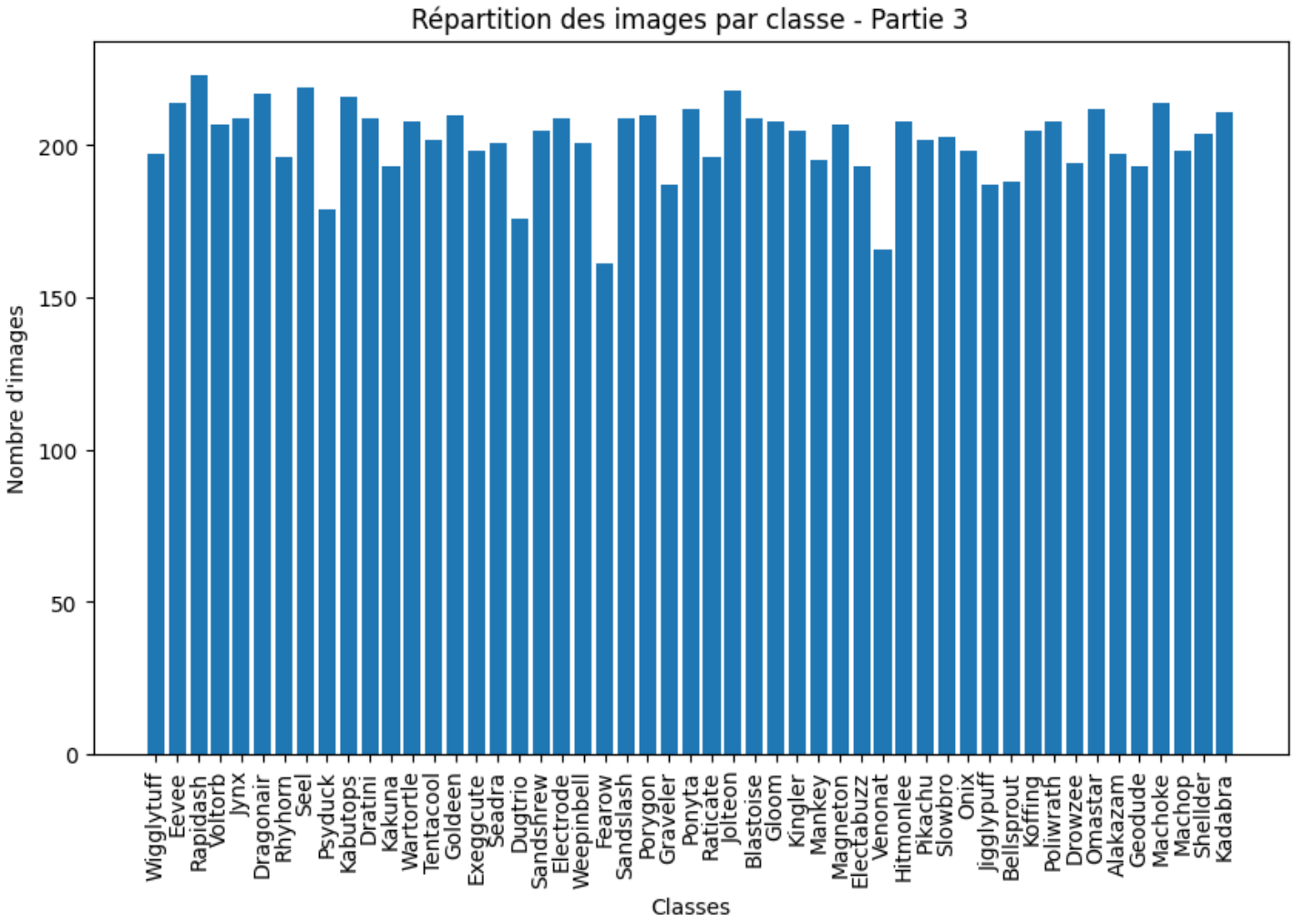
Répartition des Images par Classe - Partie 1



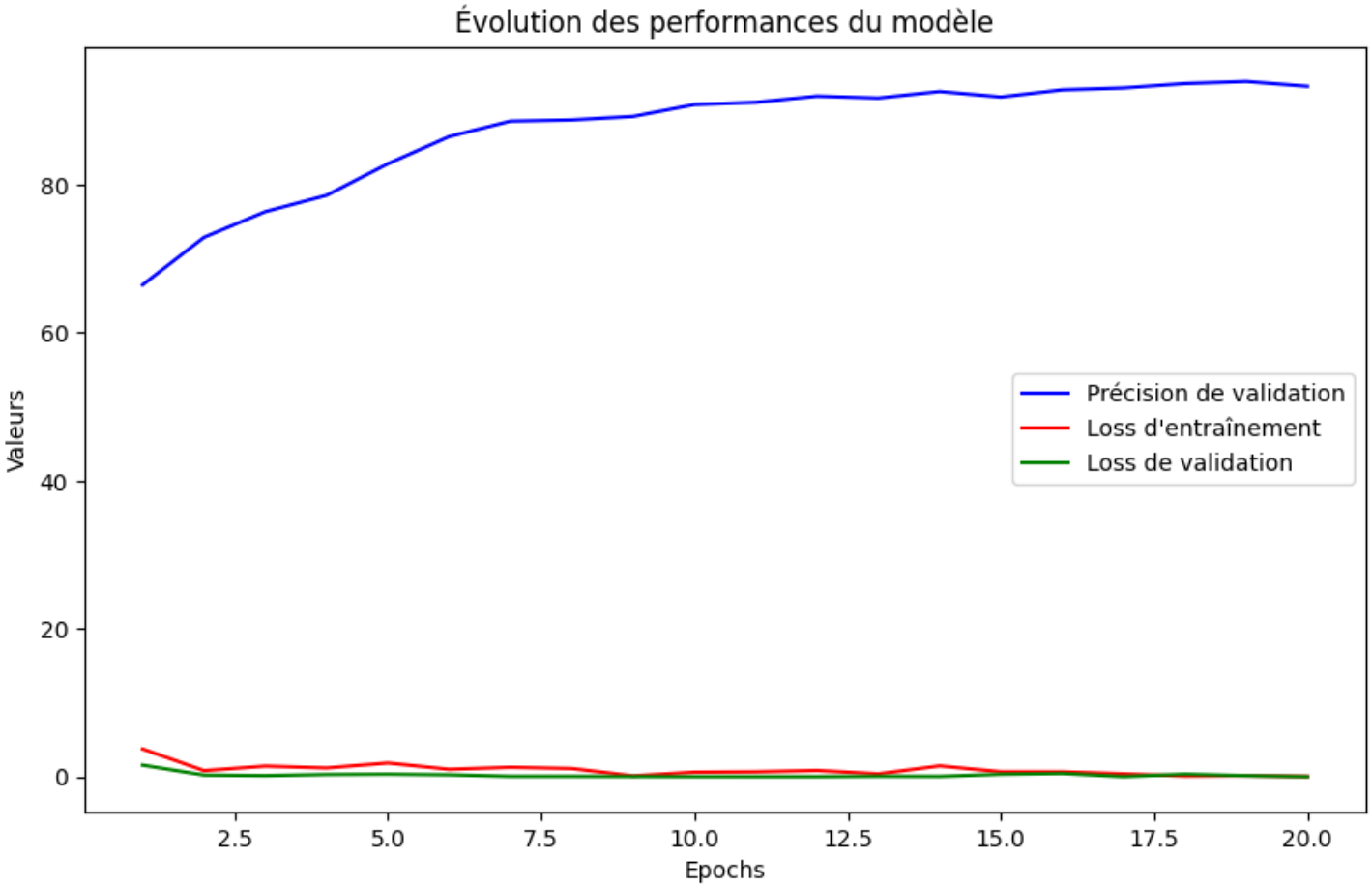
Répartition des Images par Classe - Partie 2



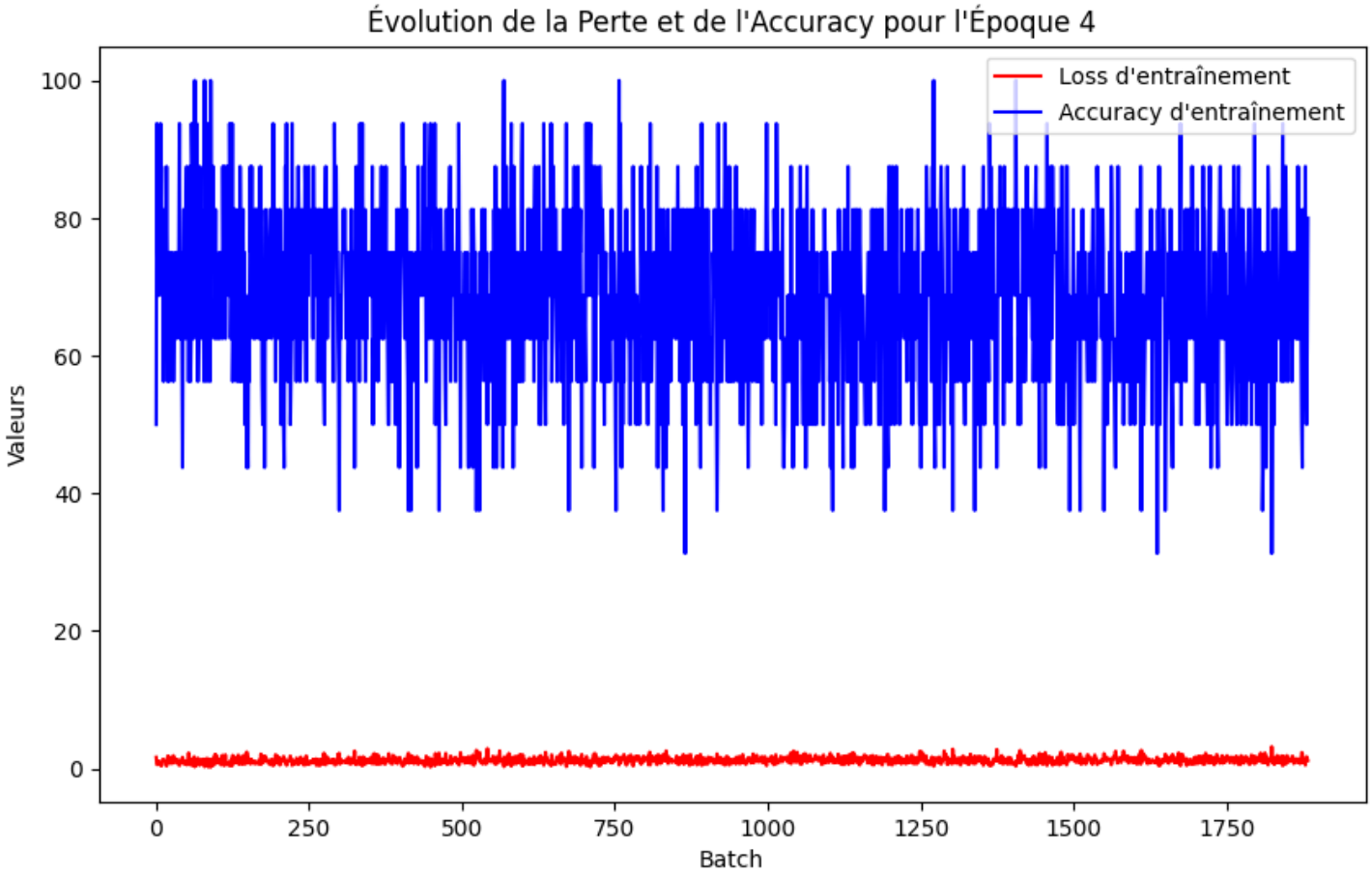
Répartition des Images par Classe - Partie 3



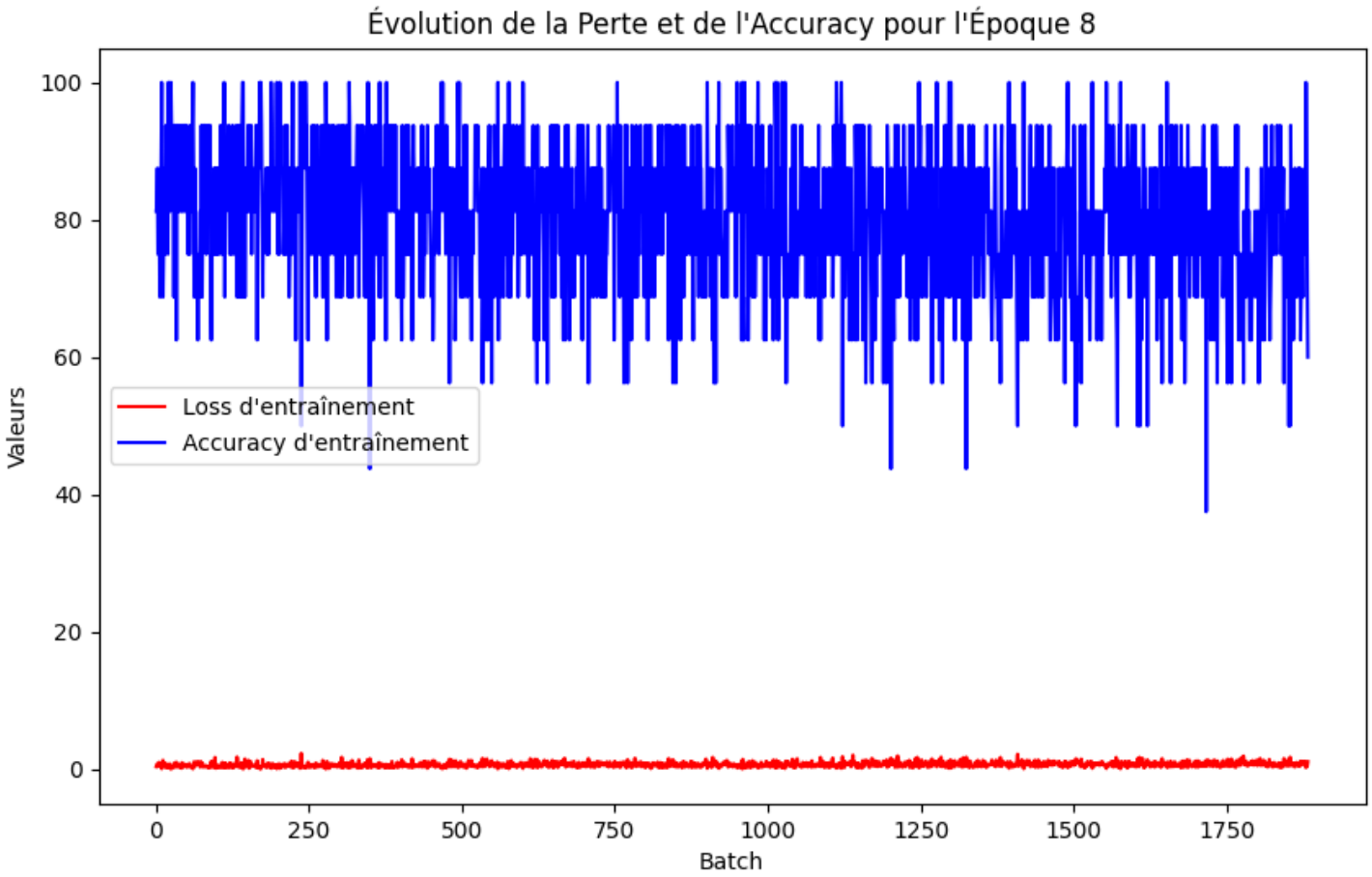
Évolution des Performances du Modèle



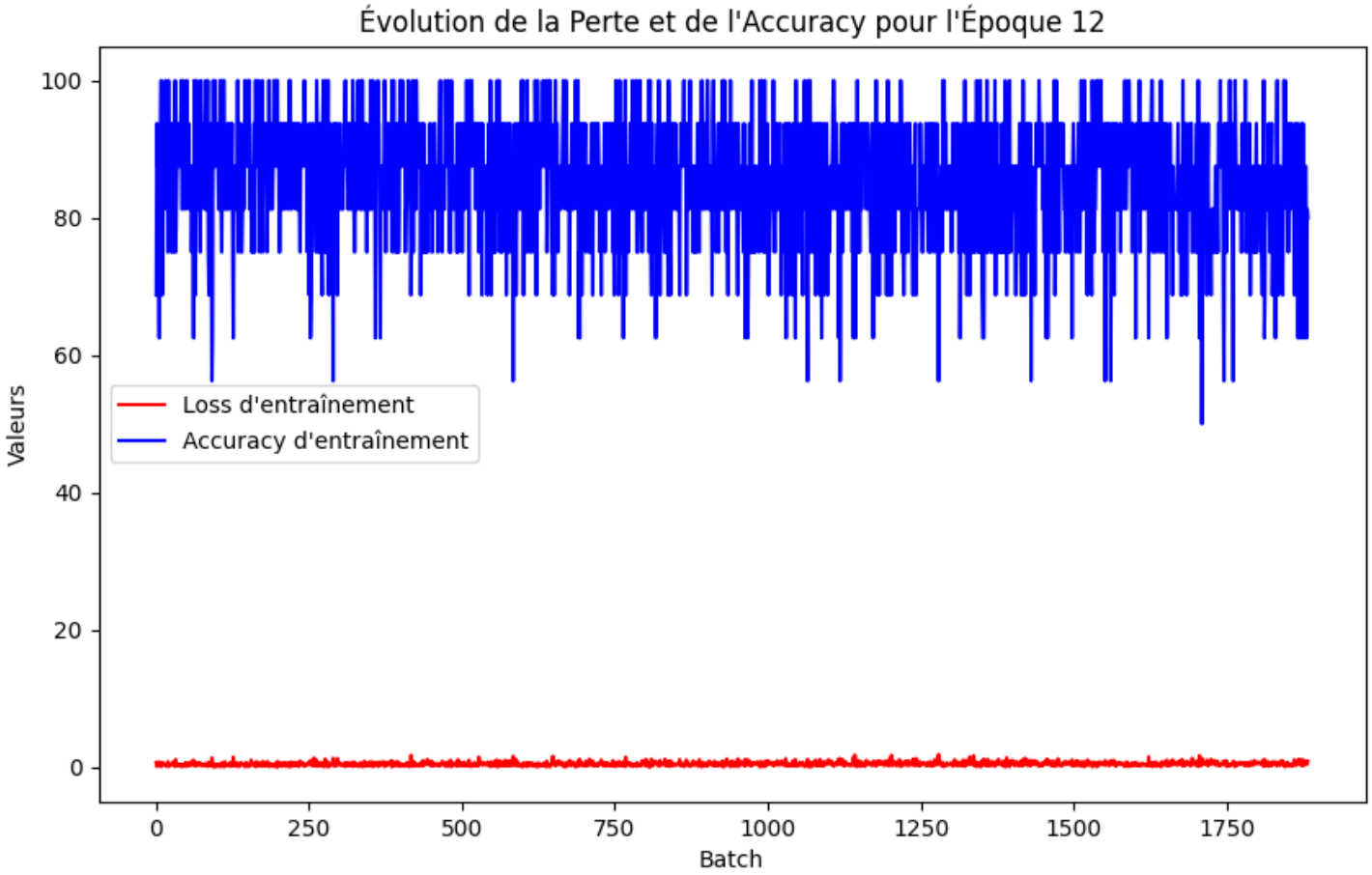
Évolution de la Perte et de l'Accuracy pour l'Époque



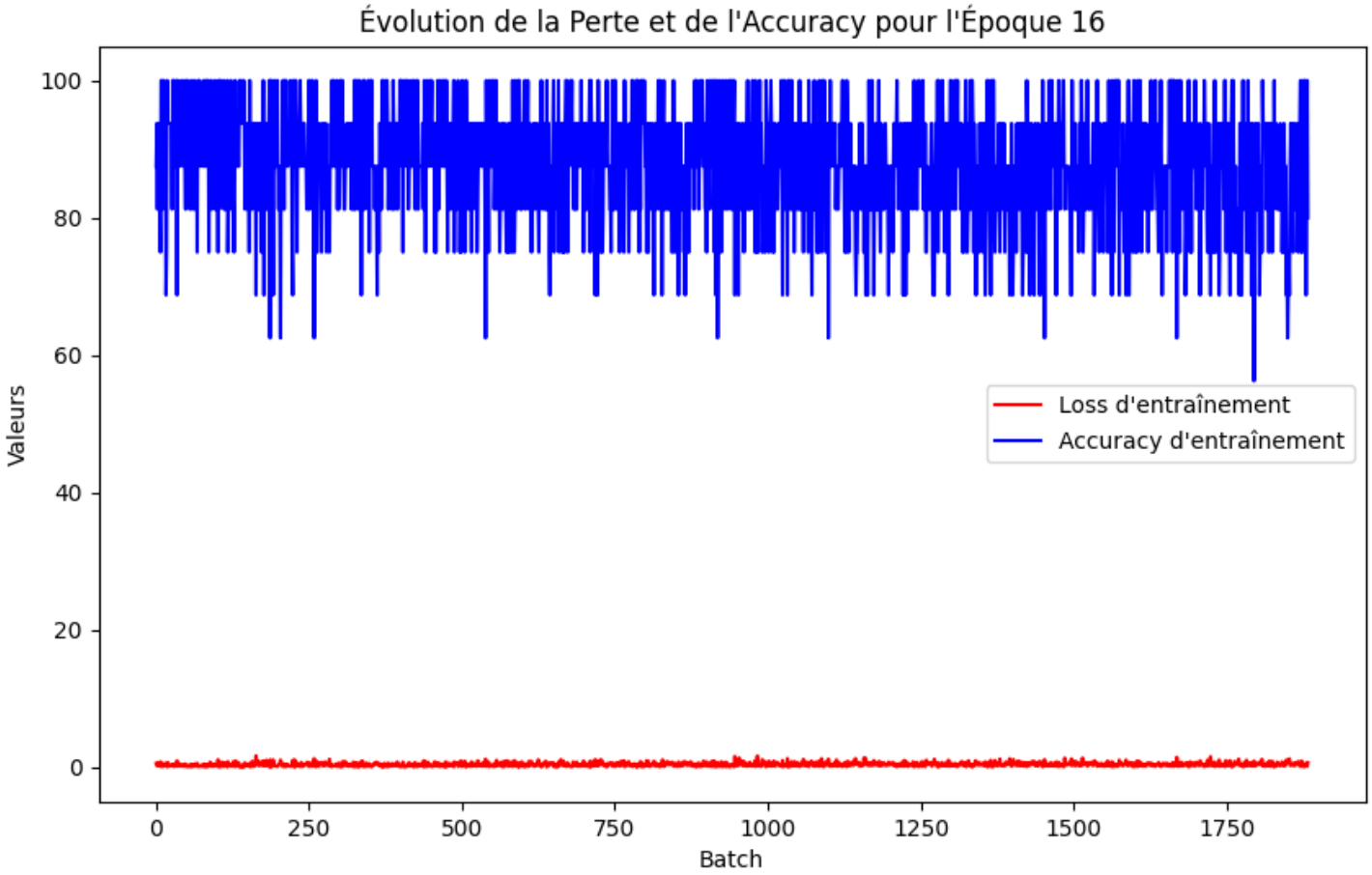
Évolution de la Perte et de l'Accuracy pour l'Époque



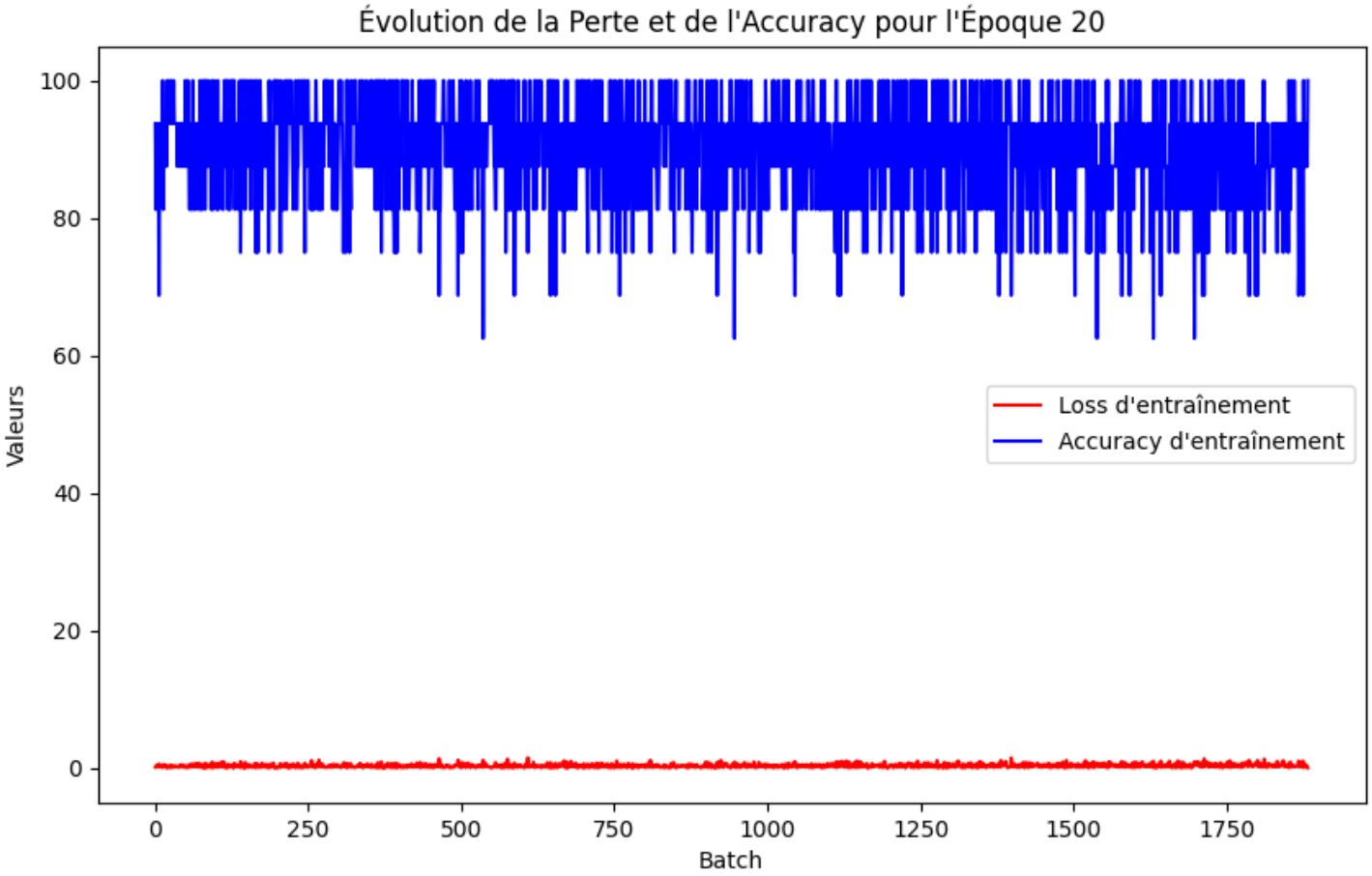
Évolution de la Perte et de l'Accuracy pour l'Époque



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	3.7303	1.5558	66.48
2.0	0.8158	0.2135	72.89
3.0	1.4237	0.1388	76.36
4.0	1.1791	0.2919	78.57
5.0	1.8434	0.3344	82.83
6.0	0.9930	0.2498	86.51
7.0	1.2660	0.0453	88.59
8.0	1.0988	0.0453	88.76
9.0	0.1079	0.0249	89.23
10.0	0.6050	0.0071	90.83
11.0	0.6570	0.0027	91.13
12.0	0.8357	0.0031	91.97
13.0	0.3772	0.0563	91.71
14.0	1.4537	0.0345	92.59
15.0	0.6583	0.3251	91.85
16.0	0.6534	0.4319	92.82
17.0	0.3746	0.0081	93.08
18.0	0.0912	0.3434	93.67
19.0	0.1486	0.1319	93.95
20.0	0.0208	0.0003	93.30