Rapport IA - Projet SMART

Groupe 1
Derya AY - Juliette DEBRESSY - Paul CHOPINET

Le but du projet SMART est de créer une application en Python utilisant la vision par ordinateur pour reconnaître automatiquement un ensemble spécifique de produits.

Choix des hyperparamètres à utiliser

Hyperparamètre	Explication
Epoch	Nombre de fois que le modèle va passer sur l'ensemble des données pendant l'entraînement. Plus ce nombre est élevé, plus le modèle peut apprendre, mais cela peut aussi mener à un surapprentissage si le modèle est trop entraîné.
Seed	Pour initialiser le générateur de nombres aléatoires
LR0	C'est le taux d'apprentissage de départ, ou à quelle vitesse le modèle fait ses ajustements au début. Si ce nombre est trop grand, le modèle peut faire des erreurs trop grosses, mais s'il est trop petit, il mettra plus de temps à apprendre.
LRF	C'est un ajustement progressif du taux d'apprentissage pendant l'entraînement. En gros, au lieu de garder le même taux d'apprentissage pendant tout le processus, ce paramètre fait en sorte qu'il change de façon plus douce et dynamique au fil du temps.
Momentum	Cela permet au modèle de conserver un peu de son inertie", donc s'il commence à bien se diriger dans une direction, il va y rester. Cela aide à accélérer l'apprentissage et éviter qu'il ne parte dans une mauvaise direction
Weight decay	Permet la régulation et pénalise les poids trop importants dans le modèle pour éviter le sur-apprentissage (overfitting)
Batch	C'est la taille des groupes d'exemples avec lesquels on travaille à chaque fois.
Patience	Si le modèle n'améliore pas ses performances pendant un certain nombre de tours, l'entraînement va être arrêté avant qu'il ne commence à sur-apprendre
Imgsz	La taille des images avec lesquelles le modèle travaille
Box	C'est un paramètre qui définit l'importance des erreurs dans les boîtes de détection
Optimizer	AdamW est une méthode qui aide à optimiser les poids du

	modèle.

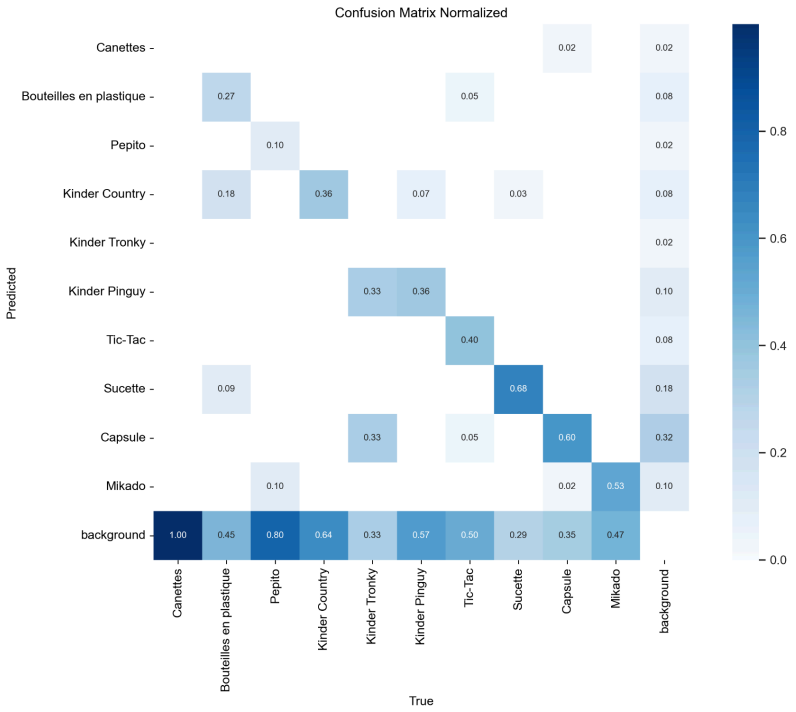
Experiment 1

Paramètres de l'expérience

Les hyper paramètres utilisés dans cette expérience sont les suivants :

- **epochs** : 100
- **lr0** : 0.0005
- **lrf** : 0.00961
- **momentum** : 0.96
- **weight_decay** : 0.00048
- **batch** : 8
- **patience** : 12
- **imgsz** : 640
- **optimizer** : AdamW

Tout au long de l'entraînement, on a vu que les pertes de boîte, de classification et DFL diminuait peu à peu, ce qui est bon signe. Mais, la perte de boîte est restée plutôt stable, ce qui indique que le modèle a un peu de mal à affiner les prédictions de localisation des objets. Par contre, la précision, le recall et la mAP50 ont progressé, bien qu'il y ait des oscillations qui indiquent que le modèle était encore en train de s'ajuster durant les premières itérations.



Le modèle est sous entraîné et confond les objets avec le fond.

Les problèmes que l’on a observés

- **Oscillations dans la perte de boîte** : La perte de boîte a montré quelques variations, ce qui pourrait signifier que le modèle a du mal à affiner les coordonnées des prédictions des objets.
- **Précision et recall qui se stabilisent trop tôt** : Même si la précision et le recall augmentent, elles semblent se stabiliser avant d'atteindre leur potentiel maximal. Cela pourrait indiquer qu’il faudrait mieux régulariser ou essayer d’autres techniques d’augmentation des données.
- **mAP50-95 plus faible que mAP50** : Le modèle a eu plus de mal à faire des prédictions correctes avec des critères plus stricts, ce qui mérite qu’on se penche dessus pour l'améliorer.

Pistes d’amélioration

- L’augmentation de l'épochs pourrait permettre au modèle de mieux converger. Aussi, une révision des taux d’apprentissage pourrait être bénéfique pour une convergence plus stable.
- L’augmentation des données pourrait être utile pour augmenter le dataset et ainsi rendre le modèle plus robuste.

Experiment 2

Paramètres de l'expérience

Les hyper paramètres utilisés dans cette expérience sont les suivants :

- **epochs** : 100
- **seed** : 42
- **lr0 (learning rate)** : 0.001
- **lrf (learning rate factor)** : 0.002
- **momentum** : 0.96
- **weight_decay** : 0.00048
- **batch size** : 16
- **patience** : 10
- **imgsz** : 640
- **plots** : True
- **close_mosaic** : 0
- **box loss coefficient** : 8.0
- **cls loss coefficient** : 0.41684
- **dfl loss coefficient** : 1.2

- **hsv_h** : 0.01469
- **hsv_s** : 0.7
- **hsv_v** : 0.44641
- **optimizer** : AdamW

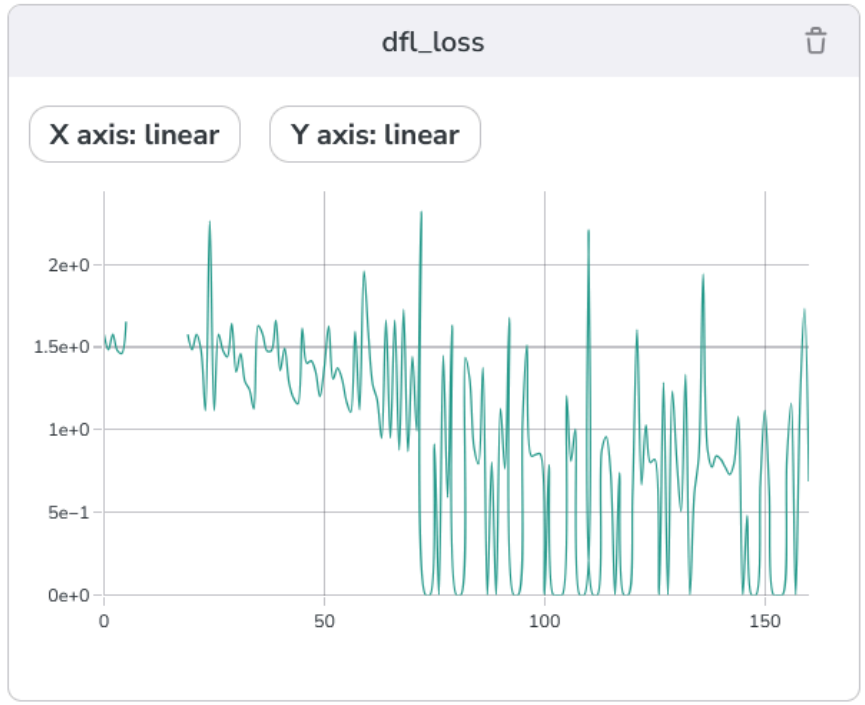
Observations

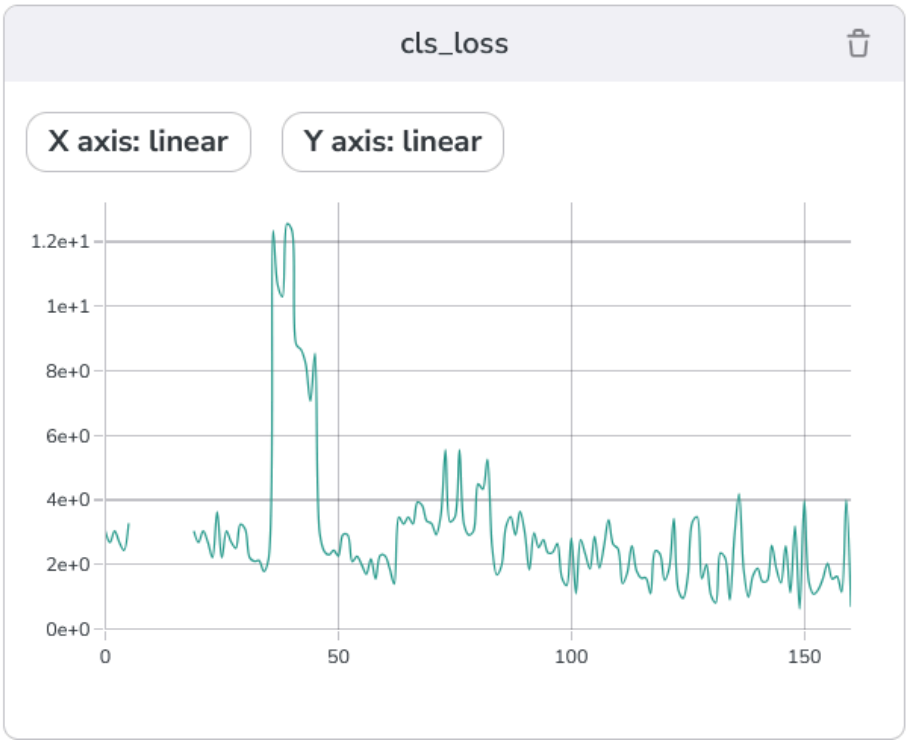
Dans cet entraînement, nous avons décidé d'augmenter les données en fonction de la teinte, la saturation et la luminosité (HSV). Ils modifient les couleurs de l'image de manière aléatoire pendant l'entraînement pour rendre le modèle plus robuste à des variations de couleur. Par exemple :

- **hsv_h** : ajuste la teinte des images.
- **hsv_s** : ajuste la saturation (intensité des couleurs).
- **hsv_v** : ajuste la luminosité (clarté).

Les pertes de boîte (box_loss) diminuent régulièrement, ce qui est une bonne indication que le modèle apprend à mieux prédire les positions des objets. Mais, les pertes de classification (cls_loss) montrent une certaine instabilité, avec de petites augmentations par moments, montrant que le modèle a encore des difficultés avec la classification des objets dans certaines situations.

L'experiment commence environ après le step 70





- Learning rate trop élevé (lrf) : Une valeur légèrement plus élevée pour lrf = 0.003 par rapport à l'entraînement précédent peut causer une instabilité en fin d'entraînement.

- Taille de batch modeste (batch=16) : Des tailles de batch plus petites peuvent générer des oscillations plus importantes dans la perte.
- Après le step 50, la cls_loss semble stabilisée avec des oscillations autour de petites valeurs (inférieures à 4). Ces petites fluctuations sont normales.
- Les résultats semblent satisfaisants si cette perte reste stable et basse. Une oscillation persistante peut signaler une difficulté à trouver un équilibre.

Ce qui a fonctionné

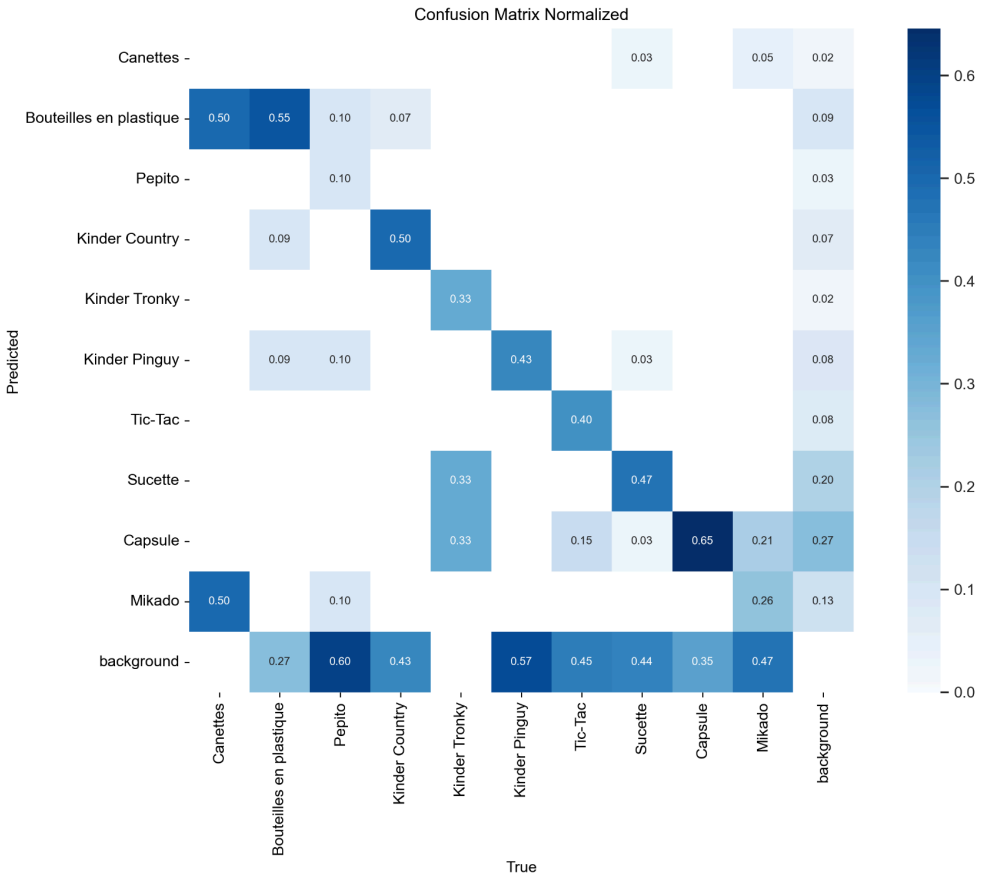
- **Réduction des pertes** : Les pertes pour les différentes métriques (box, cls, dfl) sont en diminution constante, ce qui indique que l'optimisation est en bonne voie.
- **Amélioration du mAP** : Le mAP50 et le mAP50-95 augmentent globalement, ce qui montre que le modèle apprend progressivement à mieux généraliser sur les données de validation.
- **Stabilité de l'apprentissage** : L'usage de l'optimiseur **AdamW** avec une **stratégie d'augmentation de la learning rate** est efficace pour entraîner des réseaux de neurones complexes, ce qui peut être un facteur de la stabilité observée.

Ce qui n'a pas marché ou ce qui pourrait être amélioré

- **Pertes encore relativement élevées** : Bien que les pertes diminuent, elles ne chutent pas aussi rapidement que prévu. Cela pourrait être un signe que le taux d'apprentissage (learning rate) est trop bas, ou que la convergence est trop lente pour un nombre d'époques relativement faible.
- **Fluctuations de mAP** : Le mAP50-95 montre des variations qui suggèrent un manque de stabilité dans l'apprentissage. Cela peut être dû à des choix des hyperparamètres comme **lr0**, **momentum**, ou **batch size**, qui peuvent nécessiter un ajustement pour améliorer la performance de façon consistante.

Pistes d’amélioration

- **Augmenter les époques** : Pourrait permettre de mieux converger, surtout si les pertes diminuent progressivement mais de manière trop lente.
- **Augmenter la patience** en fonction du nombre d'epoch



Experiment 3

Paramètres de l’expérience

- **"epochs"**: 150,
- **"seed"**: 42,
- **lr0**: 0.001
- **lrf**: 0.003
- **momentum**: 0.96
- **weight_decay**: 0.00048
- **batch**: 16
- **patience**: 30
- **imgsz**: 640
- **plots**: True
- **close_mosaic**: 0
- **box**: 8.0
- **cls**: 0.41684
- **dfl**: 1.2
- **hsv_h**: 0.01469
- **hsv_s**: 0.6

- **hsv_v**: 0.44641
- **optimizer**: "AdamW"

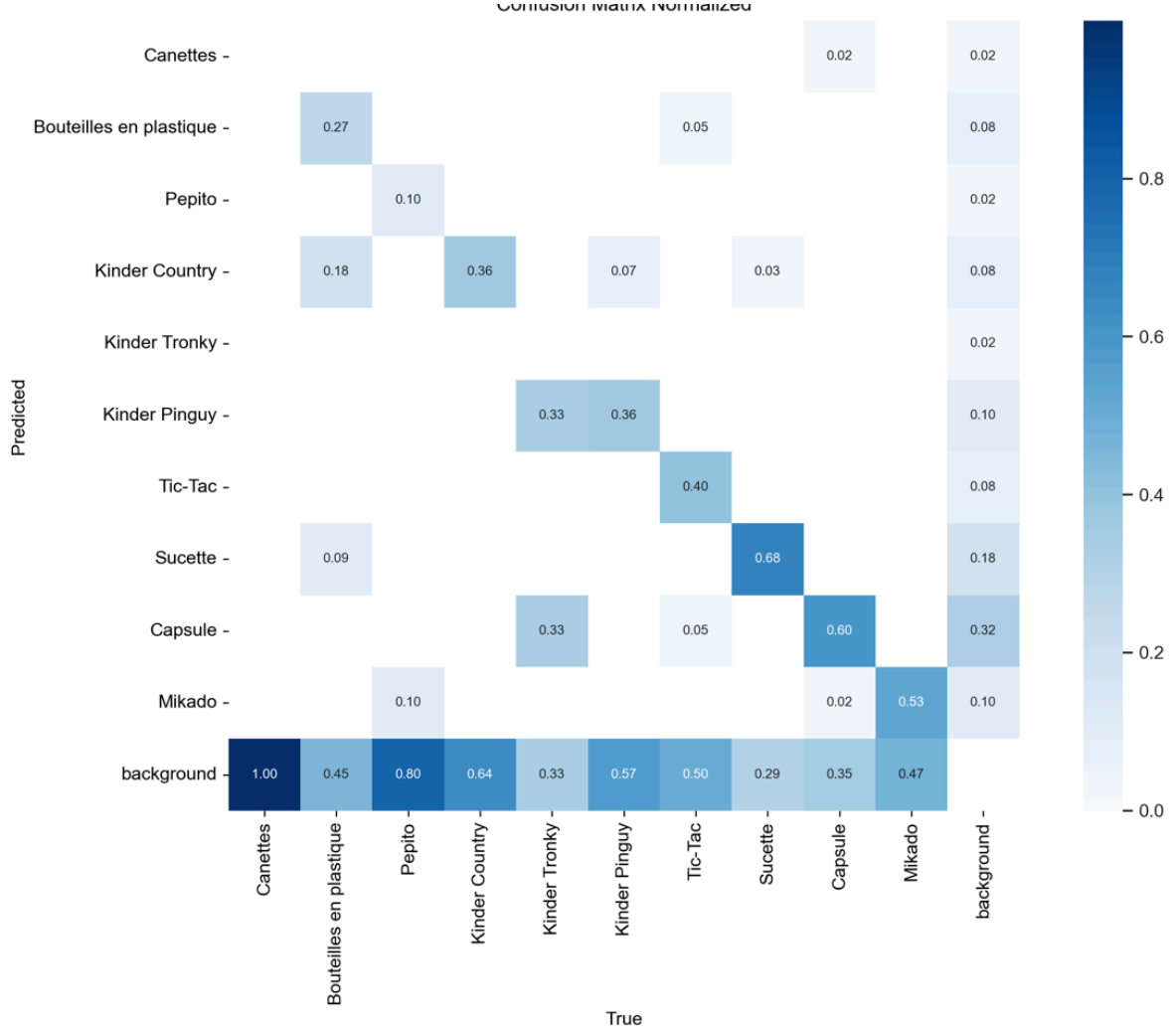
Ce qui a fonctionné

Pistes d’amélioration

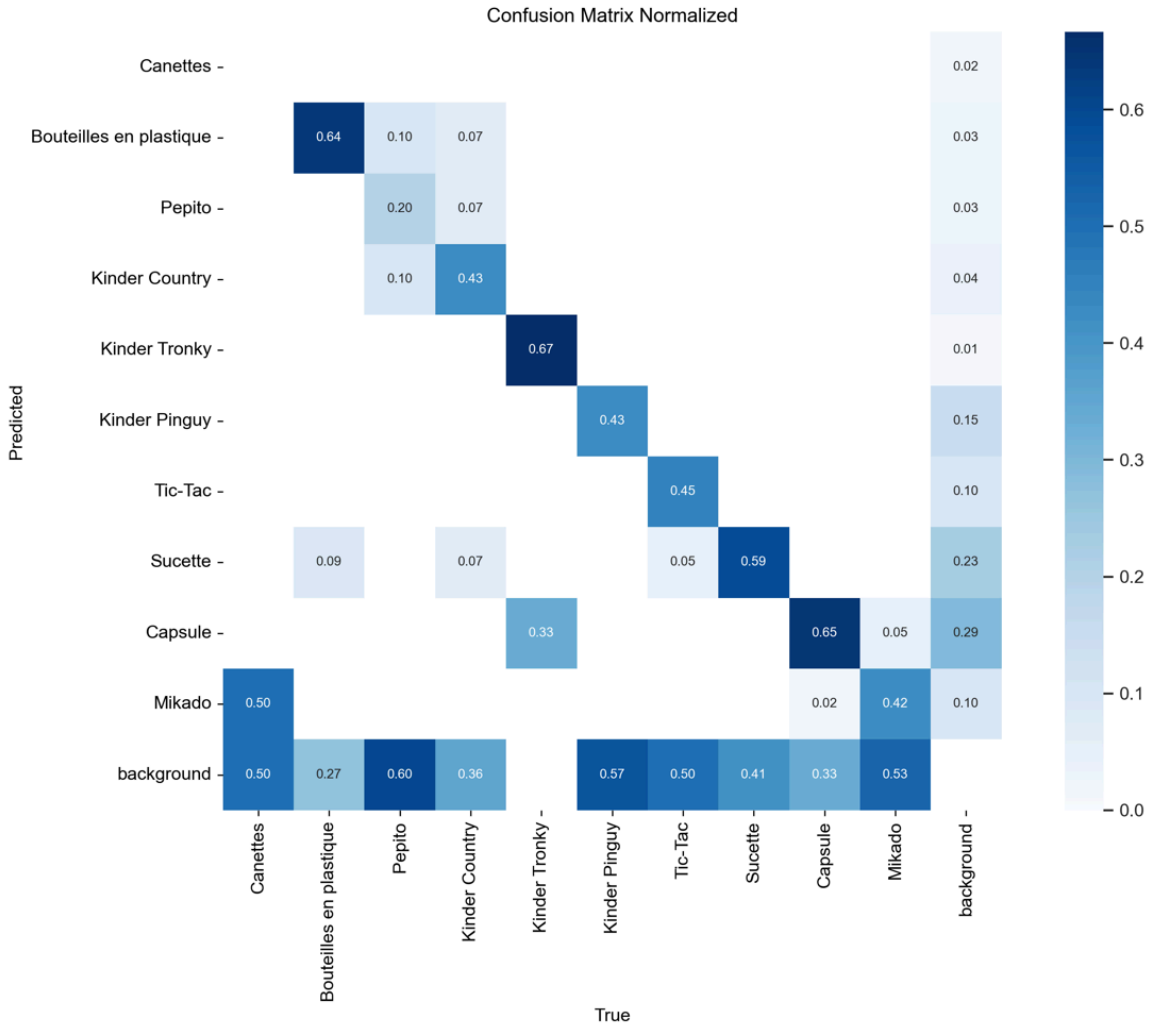
La fitness augmente de manière stable pendant une cinquantaine d’epochs puis commence à osciller, et ne s’améliore plus. Il faudrait lancer une phase exploratoire, avec plus d’epochs, en augmentant le learning_rate, et en diminuant la patience. On est peut-être dans un minimum local.

Tableau récapitulatif des experiments

Experiment 1



Experiment 3



Label	Exp1	Exp3	Evolution
Canettes	0	0	0
Bouteilles en plastique	0,27	0,64	+0,37
Pepito	0,10	0,20	+0,10
Kinder Country	0,36	0,43	+0,07
Kinder Tronky	0	0,67	+0,67
Kinder Pinguy	0,36	0,43	+0,07
Tic Tac	0,40	0,45	+0,05
Sucette	0,68	0,59	-0,09
Capsule	0,60	0,65	+0,05
Mikado	0,53	0,42	-0,11

On peut voir que le modèle a mieux appris lors de la troisième expérience par rapport à la première, avec une amélioration sur plusieurs catégories. Cela montre qu'il commence à mieux capter les tendances et caractéristiques des données. Cependant, l'apprentissage reste limité, car les résultats ne sont pas encore optimaux. Il serait intéressant de relancer l'entraînement avec un plus grand nombre d'époch, par exemple 410, et une patience de 100 afin d'observer une meilleure évolution et permettre au modèle d'atteindre un niveau de performance plus satisfaisant.

Difficultés rencontrées

L'une des principales difficultés a été de trouver les meilleurs hyperparamètres pour entraîner le modèle efficacement. Il a fallu tester plusieurs configurations pour essayer d'améliorer les performances, ce qui a pris du temps et nécessité plusieurs ajustements. De plus, le choix du nombre d' epochs et de la patience a été un vrai défi, car un mauvais réglage pouvait entraîner soit un sous-apprentissage, soit un sur-apprentissage du modèle.

Un autre point compliqué était l'analyse des résultats et leur interprétation. Il n'était pas toujours évident de comprendre pourquoi certaines valeurs évoluent dans un sens plutôt qu'un autre, ce qui nous a poussés à creuser davantage et à comparer différentes expériences. Enfin, le manque de temps nous a empêchés d'explorer encore plus de pistes, notamment en testant des architectures plus complexes ou des jeux de données plus variés.

Problèmes liés à CUDA

L'intégration de CUDA pour l'accélération GPU a été un problème récurrent, entre incompatibilités entre les versions des drivers Nvidia et du toolkit CUDA, les erreurs de mémoire GPU sans raison apparente

Conclusion générale

Par manque de temps, nous n'avons pas pu pousser l'entraînement aussi loin que nous l'aurions voulu. Cependant, on constate que le modèle a mieux appris lors de la troisième expérience par rapport à la première. L'évolution des valeurs montre une amélioration globale, mais il aurait été intéressant de lancer l'entraînement sur un plus grand nombre d'époch, comme **410**, avec une patience de **100** afin d'observer comment le modèle évolue sur une durée plus longue.

Cela aurait permis d'affiner l'apprentissage et d'obtenir des résultats encore plus stables et précis. À l'avenir, il serait pertinent de consacrer plus de temps à l'entraînement et d'explorer d'autres ajustements, comme l'optimisation des hyperparamètres, afin d'améliorer encore la performance du modèle.

Un lien vers Picsellia qui pointe sur le meilleur model_version atteint:
https://app.picsellia.com/0192f6db-86b6-784c-80e6-163debb242d5/model/019493d3-d97b-71a9-9051-3d558aedf5f4/version/0194eba1-907c-702c-a977-6e2dc388f252