Rapport d’exercice de code python/c++

Difficultés rencontrées :

T D'entrée de jeu, je fus confronté à un environnement entièrement nouveau. En effet, je venais d'acquérir un nouvel ordinateur fonctionnant sous Windows 11. La première épreuve fut la configuration et l'installation des différents outils et bibliothèques indispensables à mon travail, tels que Visual Studio et diverses bibliothèques Python. Cette étape s'est avérée chronophage. La seconde difficulté résidait dans le besoin de me familiariser de nouveau avec des outils que j'avais précédemment utilisés, tels que Spyder et Visual Studio. Ce, tout en apprenant à les interfacer avec d'autres, comme l'API Python. La dernière difficulté a été de découvrir un certain nombre de choses que je n’avais jamais faite auparavant tel que la connexion entre des scripts python et C++, ainsi que la génération d’application multi-langages.

Challenges :

De nombreuses tâches du test m'étaient familières, bien que réalisées dans des langages distincts ou sans interaction entre eux. L'un des défis majeurs, ainsi qu'une décision cruciale, a été d'utiliser l’API Python pour établir une connexion entre les scripts C++ et Python. J'ai opté pour cette stratégie en raison de sa mise en œuvre simplifiée, ne nécessitant aucune installation supplémentaire, et par conséquent, évitant des problèmes potentiels de compatibilité. Cette approche offrait également une flexibilité appréciable dans mon code Python, me permettant d'invoquer aisément diverses fonctions.

Toutefois, ce choix s'est avéré contre-productif vers la fin, car je n'ai pas pu définir de méthode pour interconnecter l'application générée avec le script Python à l'exécutable c++. J'ai donc manqué de vision sur la finalité du projet, à savoir une application entièrement autonome.

Performances :

Les différents modèles que j’ai pu étudier sont ceux présents sur : [Keras Applications](https://keras.io/api/applications/)

J’ai testé différents modèles sur un jeu d’images d’animaux, et une chose que j'ai rapidement observé que le temps d'inférence sur GPU surpassait largement celui du CPU lorsqu'exécuté directement depuis le script Python. Cette efficacité accrue est attendue, les GPU étant optimisés pour effectuer une multitude de calculs en parallèle.

Néanmoins, lors de l'exécution via un script C++, le temps d'inférence initial sur GPU est nettement supérieur à celui du CPU (par exemple, jusqu'à 4-5 fois plus long pour le VGG16). Après cette première inférence, les temps convergent vers ceux observés en utilisant directement le script Python. Cette disparité initiale peut être attribuée à la gestion de TensorFlow du GPU, qui varie selon qu'il soit invoqué depuis Python ou C++. Par conséquent, l'utilisation du GPU ne semble pas justifiée pour un faible volume d'images.

Le script Python pourrait être optimisé à divers égards. Actuellement, la majorité des fonctions dépendent du modèle utilisé, nécessitant une série de conditions pour choisir la fonction appropriée. De plus, le modèle en cours d'utilisation est systématiquement obtenu grâce à la méthode get\_config, ce qui est redondant. Une récupération directe de l'entrée de la fonction C++ pourrait éviter ces calculs. Ces améliorations n'ont pas été mises en œuvre faute de temps.

Améliorations :

Il y a une marge conséquente pour bonifier cette application. Par exemple, un paramétrage optimisé pourrait être envisagé pour chaque modèle. Il serait également envisageable de mettre en œuvre une fonction pour sélectionner le modèle le plus performant en fonction du cas d'application, voire de faire tourner plusieurs modèles simultanément pour augmenter la fiabilité des prédictions.

Une autre piste d'amélioration serait de former les modèles soi-même pour mieux en saisir les spécificités, permettant ainsi de choisir le modèle le plus adapté à chaque situation.

De plus une interface graphique serai très appréciable pour pouvoir utiliser cet outil de manière plus simple et intuitive pour n’importe quel type d’application.

Et l’amélioration à faire en premier lieu, serai de terminer le bouclage entre le script python et l’exécutable c++, qui à l’heure actuelle, ne fonctionne que si on a toutes les dépendances nécessaires.