Dorian

GROUTEAU

21904277

M2 VMI

27/10/2024

COMPTE RENDU

TP 5 Image :

Self Supervised Learning

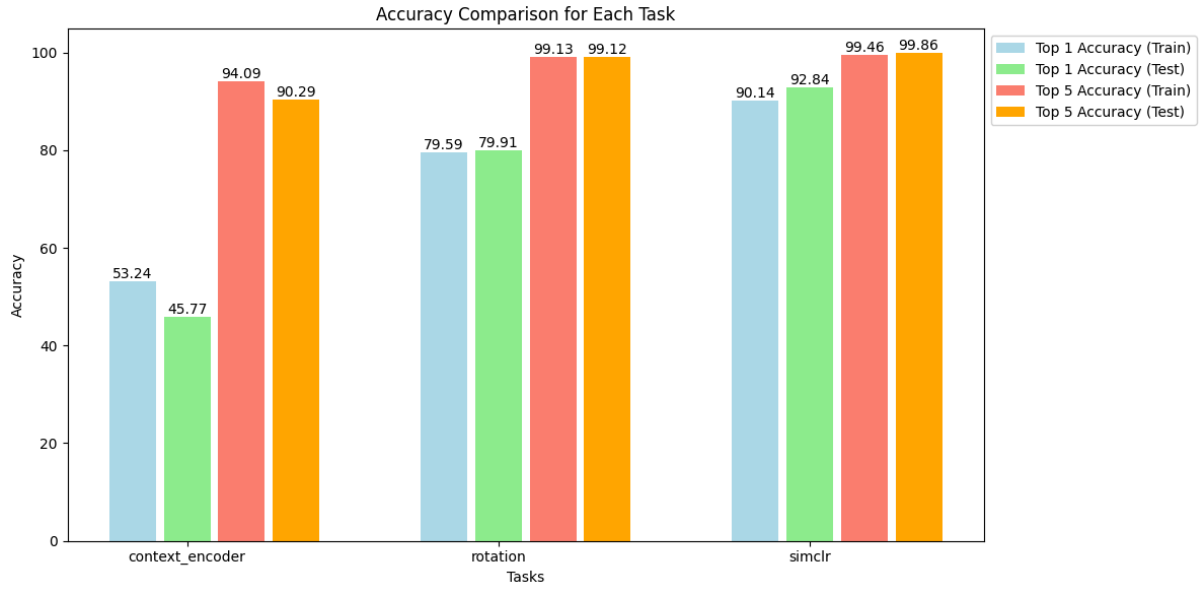
Github

L'apprentissage de représentations d'images par des méthodes auto-supervisées est devenu crucial dans le domaine de la vision par ordinateur. Nous comparerons les performances de divers modèles, tels que SimCLR, Rotation et Context Encoder.

Nous analyserons également les résultats obtenus sur le jeu de données STL10, afin d'explorer la robustesse des modèles face à des ensembles de données différents. Ce travail met en lumière l'impact des tâches prétextes sur l'apprentissage des représentations d'images.

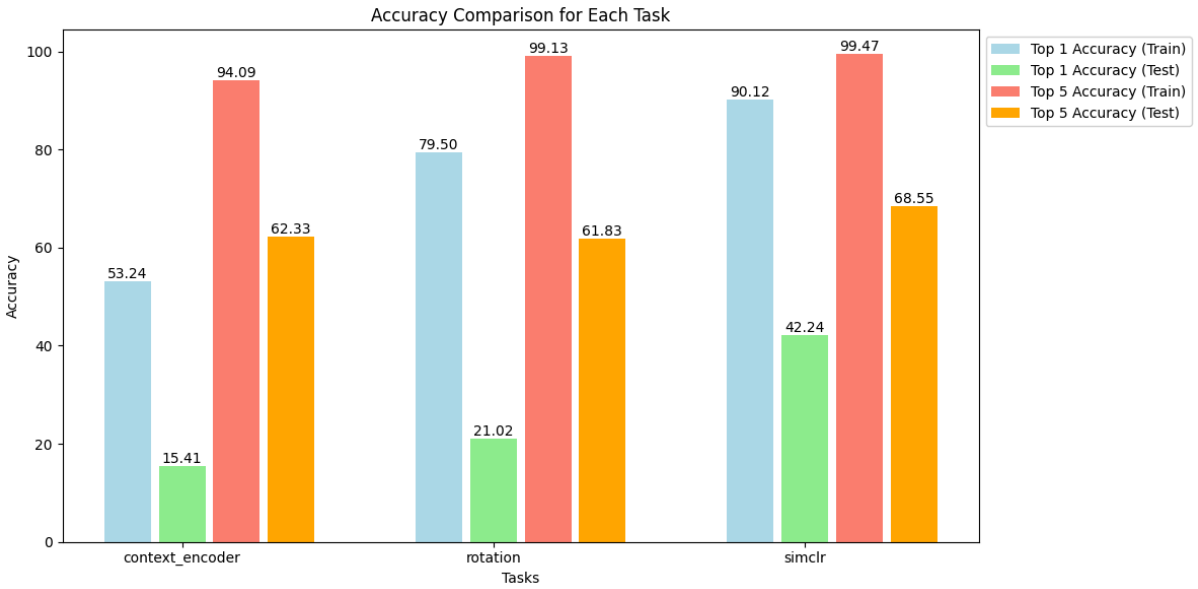
Question 3 :

En utilisant un histogramme, nous avons visualisé les performances de chaque modèle en termes de Top-1 et Top-5 accuracy sur les données d'entraînement et de test. Les résultats montrent que SimCLR surpasse nettement les autres approches, avec une Top-1 accuracy de **92.84%** et une Top-5 accuracy de **99.86%** sur le jeu de test, comparativement aux scores de **79.91%** et **99.12%** pour la tâche de rotation, et **45.77%** et **90.29%** pour le Context Encoder.



Question 4 :

Nous avons choisi d'évaluer les modèles sur le jeu de données STL10. Après avoir récupéré la partie test de ce dataset, nous avons effectué une évaluation similaire à celle de la question 3. Les résultats obtenus révèlent des performances globalement inférieures, avec SimCLR affichant les meilleurs résultats, soit **42.24%** pour Top-1 accuracy et **68.55%** pour Top-5 accuracy. Ces résultats mettent en évidence les difficultés d'application des modèles pré-entraînés sur des ensembles de données différents de ceux utilisés pour l'entraînement initial.



Question 5 :

J'ai utilisé le dataset STL10 non étiqueté pour créer une tâche prétexte qui apprend à prédire la position relative d'un patch d'image par rapport à un autre. Voici les étapes clés de mon approche :

1. J'ai défini des transformations pour convertir les images en tenseurs et les transformer en niveaux de gris. Ensuite, j'ai chargé le dataset STL10 avec les données non étiquetées et l'ai divisé en ensembles d'entraînement et de test, en attribuant 80 % des données à l'entraînement et 20 % aux tests.
2. J'ai créé une fonction create_patches pour découper chaque image en 9 patchs. Cette fonction applique un léger décalage aléatoire (jitter) lors de la découpe afin d'augmenter la diversité des données d'entrée.
3. Dans la fonction select_pair, je sélectionne aléatoirement un patch d'ancre et un patch de requête adjacents. Je calcule ensuite la position relative de la requête par rapport à l'ancre et lui attribue une étiquette en fonction de cette position.
4. J'ai défini une classe PretextDataset pour gérer le chargement des images, la découpe en patchs et la sélection des paires.
5. J'ai créé un modèle simple, PretextModel, qui se compose d'un encodeur à deux couches de convolution, suivi d'une couche linéaire qui prédit la position relative des patchs.
6. Enfin, j'ai mis en place la fonction train pour entraîner le modèle prétexte sur l'ensemble d'entraînement, en utilisant la perte crossentropy pour ajuster les poids du modèle en fonction de l'exactitude des prédictions.

Question 6 :

J’ai construit un classifieur à partir du modèle pré-entraîné afin d'évaluer les performances sur le dataset STL10 :

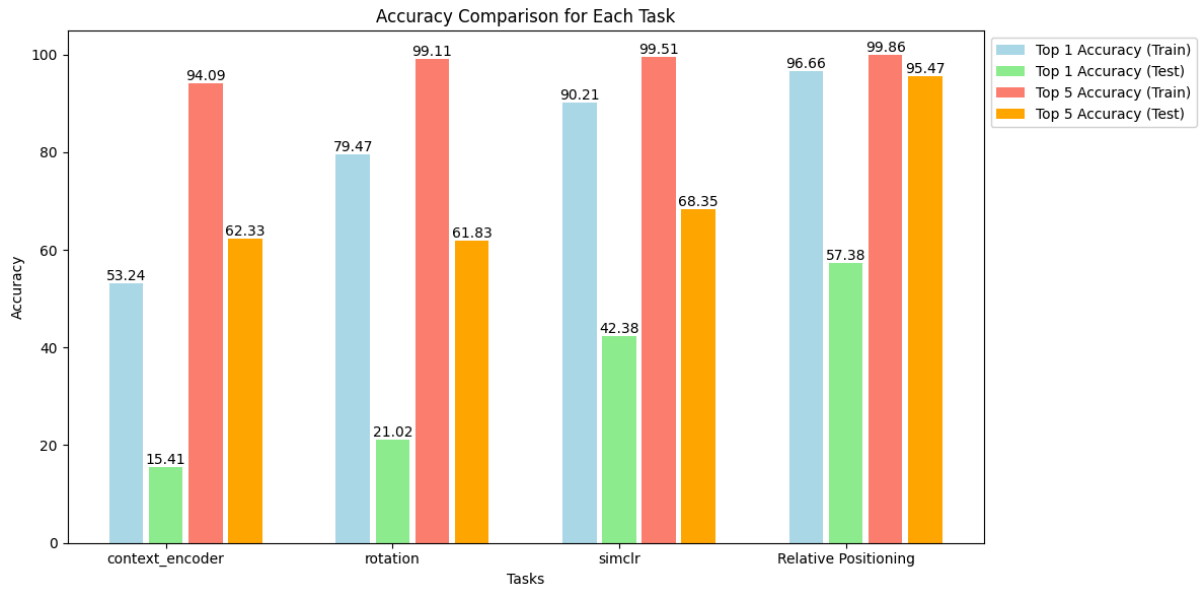
1. J'ai chargé à nouveau le dataset STL10, mais cette fois-ci avec les données étiquetées pour les ensembles d'entraînement et de test.
2. J'ai défini un modèle de classification, que j'appelle ClassifierModel, qui reprend l'encodeur du modèle prétexte. Ce modèle ajoute une couche linéaire pour classifier les images en 10 classes, correspondant aux catégories du STL10.
3. Ensuite, j'ai créé la fonction train_classifier pour entraîner le classifieur sur l'ensemble d'entraînement étiqueté.

J’ai par la suite évalué le classifieur sur le dataset CIFAR10.

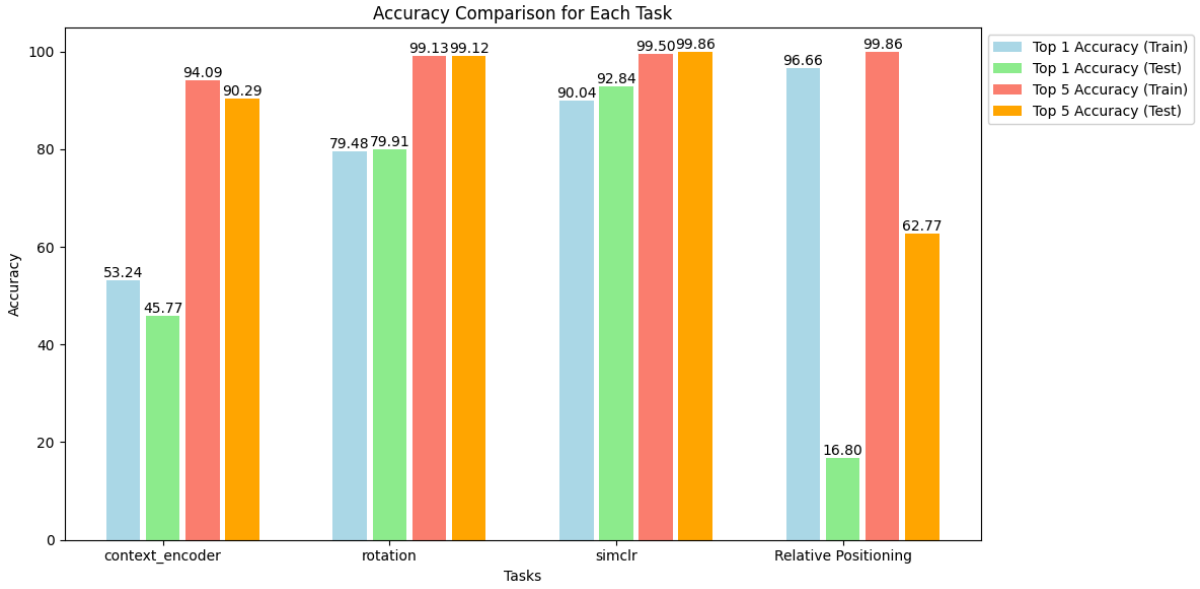
J’ai fixé un nombre d’epoch de 5 ce qui peut être trop peu, mais dans les délais j’ai préféré ne pas augmenter ce paramètre.

Récapitulatif de tous les résultats :

STL10



CIFAR10



Interprétation

Nos résultats indiquent que le modèle pré-entraîné sur la tâche de positionnement relatif des patchs d'image, appliqué au jeu de données STL10, montre une bonne capacité de généralisation. Notre classifieur utilisant ces caractéristiques est plutôt bon. Le classifieur obtient les scores de Top-1 accuracy de **57,38%** et de Top-5 accuracy de **95,47%**.

Cependant, lorsque nous évaluons le modèle sur CIFAR10, les performances diminuent de manière significative, avec des scores de Top-1 et Top-5 accuracy tombant respectivement à **16,80%** et **62,77%**. Cette diminution avait déjà été observée avec les autres classifieurs.

Bien que le modèle prétexte ait bien appris des caractéristiques utiles pour STL10, ces caractéristiques ne se traduisent pas aussi efficacement sur CIFAR10.

Il est aussi probable qu'avec davantage d'epochs pour l'entraînement, le modèle pourrait mieux ajuster ses paramètres et améliorer sa capacité de généralisation.

Conclusion

Dans l'ensemble, cette expérimentation montre que l'approche consistant à entraîner un modèle sur une tâche prétexte, puis à l'utiliser pour la classification sur des données étiquetées, fonctionne bien sur des ensembles de données similaires à celui de l'entraînement. Cependant, elle souligne également les limites du transfert de représentations lorsque le modèle est appliqué sur des données sensiblement différentes. Une meilleure adaptation des caractéristiques pourrait être explorée, notamment avec des techniques de fine-tuning adaptées aux nouveaux ensembles de données.