Abdoulahat Leye, LEYA21309606

Philip Voinea, VOIP85020100

Prof. Mohammed Bouguessa

INF7370

TP3: Autoencodeur + SVM pour classification binaire d'images d'animaux

Montage de l'architecture et entrainement du modèle

Ensemble de données

L'ensemble de données d'entrainement fournies constitue 3600 images d'animaux (1800 pour chacune des deux classes d'animaux : dauphin et requin). On a réservé 20% de cet ensemble de côté pour la validation, c'est-à-dire 720 images (360 par classe). L'ensemble de données de test constitue 600 images d'animaux, soit 300 pour chacune des deux classes.

Nous n'avons effectué aucun prétraitement des données.

Paramètres et hyperparamètres

Nous avons utilisé l'optimisateur Adam avec les valeurs attribuées par défaut dans Keras ($\alpha = 0.001$, $\beta1 = 0.9$, $\beta2 = 0.999$, $\varepsilon = 10^{-7}$). Nous avons utilisé l'erreur quadratique (MSE) comme fonction de perte.

Pour l'entrainement, nous avons utilisé des lots de 32. Nous avons entrainé le modèle pendant 100 époques sans arrêt précoce; le modèle a continué de s'améliorer au cours des 100 époques.

Architecture

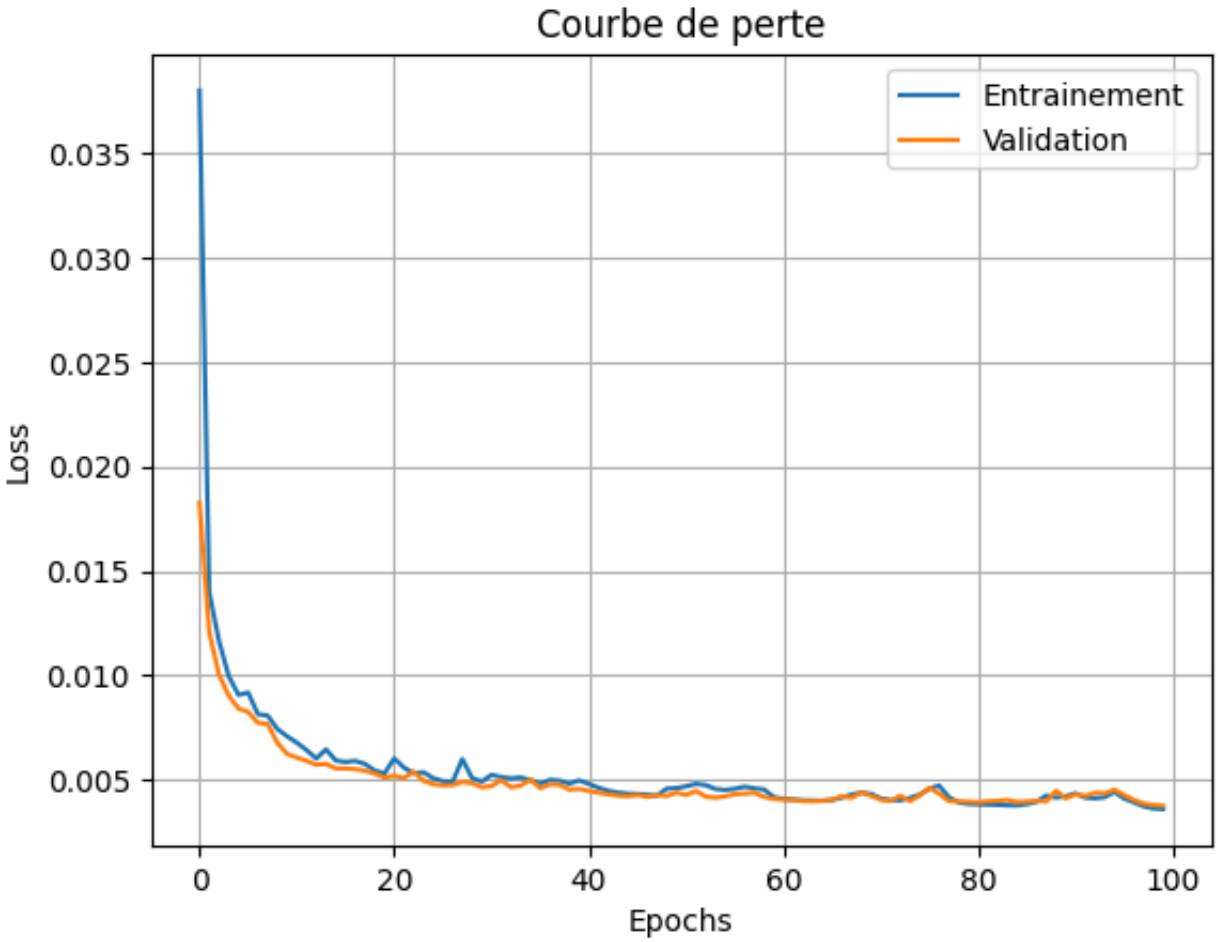
L’encodeur est composé de quatre couches de convolution avec filtre 3x3 et du zero-padding, chacune suivie d’une activation ReLU et d’une couche d’échantillonage avec filtre 2x2. La première couche de convolution a une profondeur de 128 filtres, la deuxième couche de convolution a une profondeur de 256 filtres, la troisième couche de convolution a une profondeur de 512 filtres et la quatrième couche de convolution a une profondeur de 1024 filtres.

Le décodeur débute avec quatre couches de convolution avec filtre 3x3 et du zero-padding, chacune de ces couches suivie d’une activation ReLU et d’une couche de suréchantillonage avec filtre 2x2. La première couche de convolution a une profondeur de 1024 filtres, la deuxième couche de convolution a une profondeur de 512 filtres, la troisième couche de convolution a une profondeur de 256 filtres et la quatrième couche de convolution a une profondeur de 128 filtres. Ensuite, une couche de convolution 3x3 avec zero-padding est ajoutée avec une profondeur de 3 filtres, suivie d’une activation sigmoïde pour la sortie.

Aucun dropout ou technique de régularisation, sauf le goulot d’étranglement, n’a été utilisé pour l’autoencodeur.

Résultats d'entrainement

L'entrainement de l'autoencodeur a duré 21 minutes et 44 secondes sur un GPU L4 de Google Colab. L'erreur minimale fût de 0.0036 sur les données d'entrainement et de 0.00374 sur les données de validation.



Justification du choix de l'architecture

Nous nous sommes d'abord inspirés de l'architecture fournie par l'exemple MNIST.

Tout au long de notre expérimentation, nous avons gardé la fonction d'optimisation Adam, la fonction de perte MSE, la fonction ReLU comme fonction d'activation pour les couches cachées et la fonction sigmoïde comme fonction d'activation pour la couche de sortie. Toutes les couches de convolutions ont du zero-padding.

Nous avons commencé par tester un modèle identique au modèle du MNIST, mais avec un input de taille 64x64x3. Cela n'a pas achevé une bonne accuracy; nous avons réussi à beaucoup augmenter le résultat juste en augmentant le nombre de filtres pour les couches de convolution; avec 128 filtres pour la première couche de convolutions de l'encodeur et la dernière couche de convolutions du décodeur et 256 filtres pour la deuxième couche de convolutions de l'encodeur et la première couche de convolutions du décodeur, notre SVM a atteint une accuracy de 69%.

Ensuite, sans changer l'architecture, nous avons augmenté la taille de l'input à 128x128x3 afin d'augmenter la résolution des entrées; cela a amélioré la performance de l'autoencodeur en soi, mais a empiré la séparabilité des embeddings et donc la performance du SVM linéaire. J'ai constaté que cela pourrait être dû à la taille de l'embedding qui a augmentée avec la taille de l'input. En effet, un plus petit goulot d'étranglement oblige le modèle à apprendre les attributs les plus importants; plus les embeddings sont grands, plus ils se rapprochent, à la limite, aux données originales et moins l'autoencodeur est utile.

Pour revenir à la taille originale du goulot d'étranglement tout en augmentant la résolution des inputs à 128x128, nous avons ajoutés une troisième couche de convolutions 3x3 avec 512 filtres à la fin de l'encodeur et au début du décodeur, suivi d'une couche d'échantillonnage 2x2 dans l'encodeur et d'une couche de suréchantillonnage 2x2 dans le décodeur.

Ce modèle a produit des embeddings plus séparables que tous les autres modèles, atteignant une précision de 72.17% avec le SVM linéaire.

En augmentant la dimension des entrées à 256x256 et en préservant la taille du goulot en ajoutant une quatrième couche de convolutions 3x3 avec 1024 filtres suivie d'une couche d'échantillonnage 2x2 à la fin de l'encodeur et la même couche de convolutions suivie d'une couche de suréchantillonnage 2x2 au début du décodeur, nous avons atteint une accuracy en SVM de 72.67%.

Ensuite, nous avons expérimenté avec le drop-out dans les couches cachées de l'encodeur comme technique de régularisation, ce qui n'a que réduit la performance du modèle, même avec un taux d'extinction minime de 0.1. L'augmentation des données aussi n'a qu'empiré les modèles.

Nous avons ensuite tenté de simplement doubler le nombre de filtres de nos modèles mais cela a diminué à nouveau la performance du SVM. Encore une fois, c'est probablement à cause du goulot d'étranglement élargi. Nous avons donc fait l'inverse en étranglant le goulot en diminuant de moitié le nombre de filtres dans la dernière couche de convolutions de l'encodeur (ainsi que dans la première couche de convolutions du décodeur); cela a encore réduit la performance du SVM. D'après nous, la réduction de la dernière couche de convolutions aurait mené à une perte importante d'informations sur l'image. Nous avons interprété cela comme étant un signe que nous avions le nombre adéquats de filtres pour le problème.

Pourtant, une autre méthode d'étranglement du goulot a bonifié la performance : en ajoutant une couche de convolutions 3x3 avec le double des filtres de la couche de convolution précédente suivie d'une couche d'échantillonnage 2x2 à l'encodeur (et l'inverse au décodeur), nous avons réussi à améliorer nos modèles. Nous avons d'abord testé ce concepte sur notre petit modèle aux inputs de taille 64x64, auquel nous avons ajouté une troisième couche de convolutions 3x3 avec 512 filtres et une couche d'échantillonnage 2x2. Finalement, au modèle aux inputs de taille

128x128x3, nous avons ajouté une quatrième couche de convolutions 3x3 avec 1024 filtres et une couche d'échantillonnage pour créer notre modèle final. Nous n'avions malheureusement pas pu adapter cette solution pour des inputs de taille 256x256 pour créer un modèle à 5 couches de convolutions et d'échantillonnage à cause de limites computationnelles (de toute manière, l'augmentation de la dimension des inputs de 128x128 à 256x256 la première fois n'a pas beaucoup augmenter la accuracy du SVM). Nous attribuons le succès de cette démarche au goulot d'étranglement plus étroit ainsi qu'à l'addition d'une nouvelle couche de convolutions qui permet l'apprentissage de caractéristiques plus abstraites. Étant donné que la dernière méthode d'étranglement de goulot a empiré la performance du modèle, nous pensons que le niveau additionnel d'abstraction procuré par la nouvelle couche de convolutions joue un rôle important dans la capacité du modèle à stocker plus d'information dans un espace latent plus petit; en théorie, c'est l'abstraction qui permet la bonne compression de données en général

Ajouter encore une autre couche de convolutions et d'échantillonnage a empiré le modèle; nous théorisons que la taille du goulot est trop petit après tant de couches d'échantillonnage.

Finalement, nous avons expérimenté avec l'implémentation de couches denses vers le goulot d'étranglement; pourtant, ces modèles ont mal performé parce que nos ressources computationnelles ont trop limité la taille du goulot étant donné la complexité explosive des couches entièrement connectées.

Résultats

Accuracy du SVM sur embeddings: 74.17%

Accuracy du SVM sur les images originales : 59%

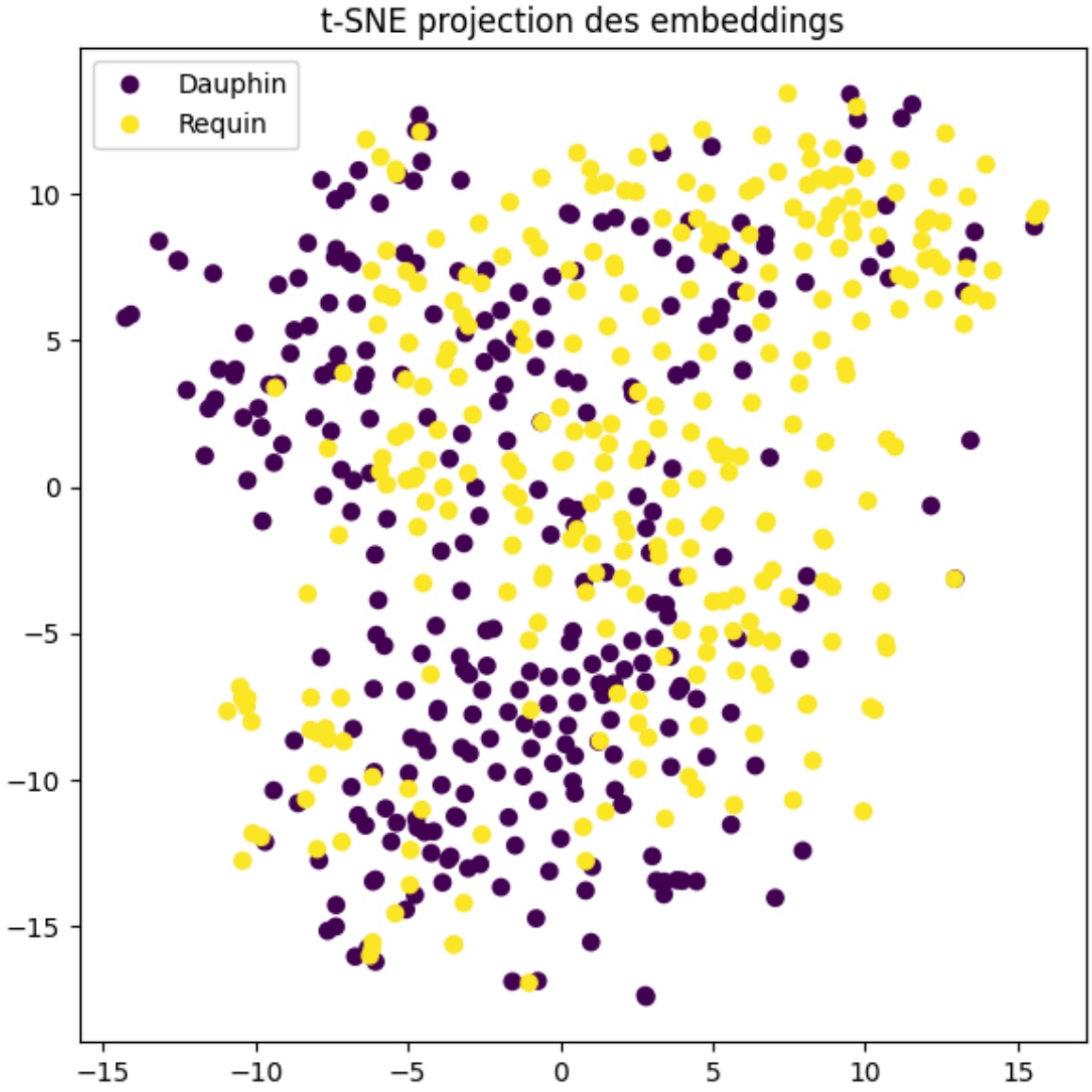
Reconstruction des images

Dauphin (originale)

Requin (originale)

Dauphin (reconstruite)

Requin (reconstruite)



Analyze des résultats

Les reconstructions des images sont devenues de moins en moins précises plus on a étranglé le goulot (un effet attendu de quelconque compression des données) pourtant, ce ne sont pas que les détails qui disparaissent et les formes qui perdent leur définition comme dans un

simple filtre flou. On observe, par exemple, que le ventre du requin reconstruit dans l'exemple est ondulé comparé à l'image originale.

Les projections t-SNE des embeddings ne sont pas linéairement séparables; elles paraissent même très entremêlées. Ceci est très différent des projections t-SNE des embeddings MNIST de l'exemple, qui sont comparativement très proprement séparables; pourtant, étant donné que notre accuracy est seulement de 74.17%, ce qui ne dépasse pas énormément le hasard dans un problème binaire, ce n'est guère surprenant que nos groupes ne présentent qu'une faible différence entre eux dans notre graphique t-SNE.

Conclusion

Nous avons d'abord adapté l'architecture de l'exemple MNIST pour nos données en augmentant la taille des inputs à 64x64x3; de là, nous avons d'abord amélioré la performance du modèle en augmentant le nombre de filtres par convolution à 128 pour la première couche de convolutions et à 256 pour la deuxième couche de convolutions.

D'ici, l'augmentation de la résolution des entrées à 128x128 et puis à 256x256 a amélioré l'accuracy du SVM tant que des couches de convolution et d'échantillonnage étaient ajoutées au fur et à mesure afin de préserver la taille de la couche de codages — c'est où nous avons relevé le premier défi de se rendre compte que l'augmentation de la résolution des entrées réduit l'accuracy du SVM parce que cela mène à une couche de codage plus grande et moins discriminatoire.

Ensuite, nous avons expérimenté avec des méthodes de régularisation tels le dropout et l'augmentation des données; pourtant, ces méthodes n'ont que nuit à l'apprentissage.

Doubler le nombre de filtres dans chaque convolution n'a qu'empiré les résultats du SVM et nous théorisons que c'est à cause de l'élargissement du goulot; d'autre part, l'étranglement du goulot par la

réduction du nombre de filtres de la dernière convolution a aussi empiré l'accuracy du SVM et nous théorisons que c'est à cause de la perte d'informations importantes dans la dernière convolution.

Par contre, l'étranglement du goulot par l'ajout d'une couche additionnelle de convolutions avec une couche d'échantillonnage a amélioré la séparabilité des embeddings; nous théorisons que l'étape additionnelle d'abstraction procurée par la nouvelle couche de convolutions permet un stockage de caractéristiques plus représentatives dans un espace latent plus restreint, ce qui distinguerait cette technique d'étranglement du goulot de la dernière.

Finalement, nous avons tenté d'étrangler le goulot par le biais de couches denses aplaties; cependant, nos ressources computationnelles ont trop restreint la taille du goulot à cause de la complexité des couches denses. Cette approche n'était donc pas praticable et nous a donné parmi les pires accuracy au SVM.