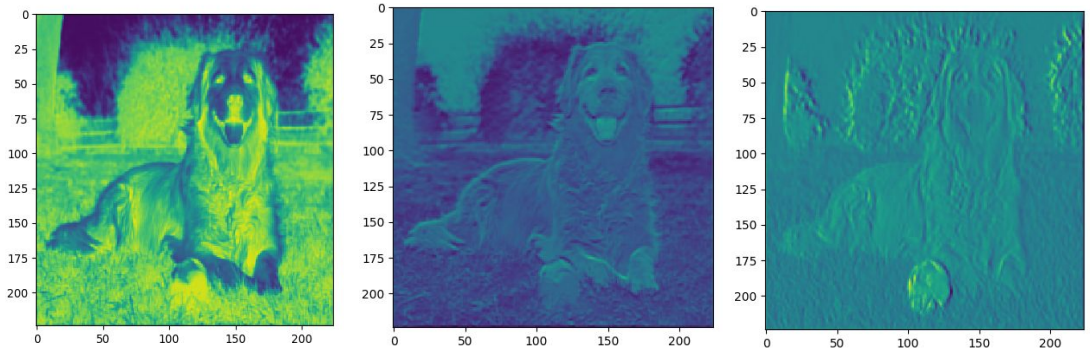
Keyvan Beroukhim
Arij Riabi

TP 8 - Classification d'images par CNN

1) Le nombre de poids d'une couche fully connected est le produit de la dimension d'entrée et de la dimension de sortie (en négligeant les biais). Cela nous donne 7*7*4096 + 4096*4096 + 4096*1000 = **124M de paramètres** environ dans le réseau VGG16, cela représente 90% des poids du réseau (total=138 M)
2) La sortie du réseau est un vecteur de taille 1000, cela correspond à la **probabilité d'appartenance à chacune des 1000 classes** pour lesquelles ce réseau a été entraîné.
3) Le réseau fonctionne très bien si l'image en entrée correspond à une des classes apprises, mais il n'a **aucune chance de réussir pour une autre classe**. Le réseau se trompe parfois, surtout quand on lui présente des images de style différent.
4) On visualise l'**activation obtenue par l'application de différents filtres sur l'image**. La première couche est comparable aux filtres sobels par exemple au sens où elle donne des informations de bas niveau.



5) Le dataset 15Scene est **beaucoup trop petit pour permettre d'apprendre un réseau contenant des millions de paramètres comme VGG**.
6) On essaye ici de se servir de la partie feature extraction de VGG et de seulement réapprendre la partie finale. L'extraction de caractéristiques générales d'une image est la même quelle que soit la tâche à effectuer. On peut donc **se servir des très bonnes performances du réseau VGG**.
7) Le problème avec cette méthode est que VGG a été entraîné avec un type particulier d'image, si on lui fournit des images avec des **caractéristiques nouvelles**, par exemple des images en noir et blanc, elles ne pourront **pas être détectées**. Si les images possèdent une **autre dynamique** (par exemple si elles sont plus sombres), le réseau aura plus de mal à détecter les caractéristiques.
8) Plus on s'éloigne de l'entrée et plus les feature maps sont de haut niveau, les premières convolutions détectent par exemple des contours, et les dernières des parties de chien ou de chat.
9) Les images de 15Scene sont en noir et blanc, pour pouvoir les faire passer par VGG, on duplique 3 fois leur channel.

10) En utilisant un SVM apprenant à classifier les images à partir d la feature map, on obtient un **taux d'erreur de 11%**. Dans le TP précédent, en utilisant un SVM sur les BoW, on avait obtenu un taux de 30%. Les résultats obtenus avec VGG sont 3 fois meilleurs, cela montre que les caractéristiques extraites par VGG sont de meilleure qualité.
11) **Utiliser un SVM bloque le fine-tuning**, on pourrait le remplacer simplement par une autre couche linéaire à la fin du réseau.

12) Remplacer le SVM par une simple couche neuronale permet la backpropagation des gradients dans VGG. Pour des raisons de performances, nous ne souhaitons pas fine-tuner les poids de VGG, cela ne peut donc pas nous apporter un gain de performances et nous fait perdre l'utilisation de l'hyperparamètre C.

    Le paramètre C est très important pour les SVM, chercher une bonne valeur pour ce paramètre est donc pertinent. en essayant des valeurs de 0.001, 0.01, 0.1, .5, 1 et 10 le meilleur score en test est C=1 (comme précédemment)

    Pour extraire les features de VGG, on utilise actuellement la partie 'feature' et le début de la partie 'classification'. D'après le nom des parties du réseau de neurones, il est légitime de penser que pour classifier de nouvelles classes d'objets, il ne faut pas se servir de la partie 'classification' de VGG. On réalise donc une nouvelle fois les expériences en utilisant **seulement la partie 'feature'**. Les vecteurs de caractéristique extraits sont plus gros que précédemment et sont extraits plus rapidement étant donné qu'on applique moins de couches de VGG. Les performances obtenues sont **nettement meilleures** : on atteint un **taux d'erreur de 1%** seulement, c'est à dire une différence d'un ordre de grandeur. **Utiliser la partie 'feature' uniquement est donc primordial.**