TP 8 : Du transfert learning au transfert de style

Jérôme Pasquet

Pré-traitement

1. Proposez une fonction permettant d'ouvrir une image en RGB d'une taille quelconque et de la transformer en une image de taille 224x224 BGR. Nous n'accordons pas d'importance à la méthode d'interpolation utilisée tant que celle-ci n'introduit pas de distorsion.
2. En utilisant la commande 'wget' téléchargez une image de style et une image de référence sur votre session Colab.
3. Construisez trois paquets contenant une et une seule image pour la référence, le style et une copie de la référence (notée R).
4. Vous avez fini le pré-traitement ! Appelez votre chargé de TD !

Transfert Learning

1. Nous allons utiliser le réseau VGG19 dont les poids sont téléchargeables à l'adresse suivante : https://media.githubusercontent.com/media/tensorlayer/pretrained-models/master/models/vgg19.npy
2. Analysez la classe Vgg19 qui vous a été envoyée. Remarquez que :
   (a) les poids sont chargés via la déclaration des poids avec une distribution constante !
   (b) le paramètre reuse permet de charger des poids avec le même nom (en les partageant).
   (c) vous pouvez accéder à une couche en utilisant la méthode 'get' et le nom de la couche.
3. Une fois le code correctement compris vous pouvez passer à la suite du TP.

Deep Learning

1. Construisez un réseau VGG19 en prenant en paramètre le paquet correspondant à l'image de style. Vous noterez que cette image n'est pas variable au cours du temps.
2. Construisez un réseau VGG19 en prenant en paramètre le paquet correspondant à l'image de référence. Vous noterez que cette image n'est pas variable au cours du temps.
3. Construisez un réseau VGG19 en prenant en paramètre le paquet correspondant à la copie de la référence (R). **Cette image est une variable**.
4. Définissez une fonction Euclidienne qui calcule la distance euclidenne entre R et la référence pour une ou plusieurs couches.
5. Définissez une fonction Gram qui calcule la distance des matrices de Gram entre R et le style pour une ou plusieurs couches. Dans un premier temps vous pouvez sauter cette étape afin de vous focaliser sur l'architecture globale. Vous utilisez alors une simple distance Euclidienne. Une fois le réseau construit revenez à cette étape.
6. Définissez la fonction de coût globale
7. Définissez votre optimiseur. Attention vous préciserez que celui-ci n'effectue des mises à jour que sur l'image R. Pour cela, renseignez-vous sur la documentation de la fonction 'minimize'.
8. Initialisez les poids **qui ne sont pas** déjà initialisés (à cause du chargement de Vgg19).
9. Lancez votre optimiseur avec un pas constant de 10^{-2} et affichez R toutes les 1000 itérations.

Devoir maison

Lisez le papier ci-contre. Proposez une implémentation (non optimisée) de la batch normalisation par contraste.
Instance Normalization: The Missing Ingredient for Fast Stylization - Dmitry Ulyanov, Andrea Vedaldi et Victor Lempitsky