Projet Traitement Automatique d'Images
Colorisation d'images

Hao LIU, Robin DURAZ

7 janvier 2019

1 Introduction

Nous avons voulu nous pencher sur le travail réalisé dans l'article Colorful Image Colorization, réalisé par Richard Zhang, Phillip Isola, et Alexei A. Efros.

2 Résumé de l'article

2.1 Définition de problème

Commençons par définir le problème de colorisation en termes d'espace colorimétrique CIE Lab. Comme l'espace colorimétrique RGB, il s'agit d'un espace colorimétrique à 3 canaux, mais contrairement à l'espace colorimétrique RGB, les informations de couleur ne sont codées que dans les canaux a (composante vert-rouge) et b (composante bleu-jaune). Le canal L (luminosité) ne code que les informations d'intensité.

L'image en niveaux de gris que nous voulons colorer peut être pensée comme le canal L de l'image dans l'espace colorimétrique Lab et comme objectif de rechercher les composants a et b. L'image de laboratoire ainsi obtenue peut être transformée en espace colorimétrique RVB à l'aide de transformations d'espace colorimétrique standard.

Pour simplifier les calculs, l'espace ab de l'espace colorimétrique Lab est quantifié en 313 catégories, comme indiqué dans la Figure 1. Plutôt que de rechercher les valeurs a et b pour chaque pixel, il est simplement nécessaire, grâce à la quantification, de trouver un nombre compris entre 0 et 312 pour le canal ab. Une autre façon de penser au problème est que nous avons déjà le canal L qui prend des valeurs de 0 à 255 et nous devons trouver le canal ab qui prend des valeurs comprises entre 0 et 312. La tâche de prédiction de couleur est maintenant transformé en un problème de classification multinomiale où pour chaque pixel gris il y a 313 classes à choisir. Voir la figure 1

2.2 Architecture CNN pour la colorisation

L'architecture proposée par Zhang et al est un réseau de style VGG avec plusieurs blocs convolutifs. Chaque bloc comporte deux ou trois couches de convolution suivies d'une unité linéaire rectifiée (ReLU) et se terminant par une couche de normalisation par lots. Contrairement au réseau VGG, il n'y a pas de couches de pooling ou entièrement connectées. Voir la figure 2

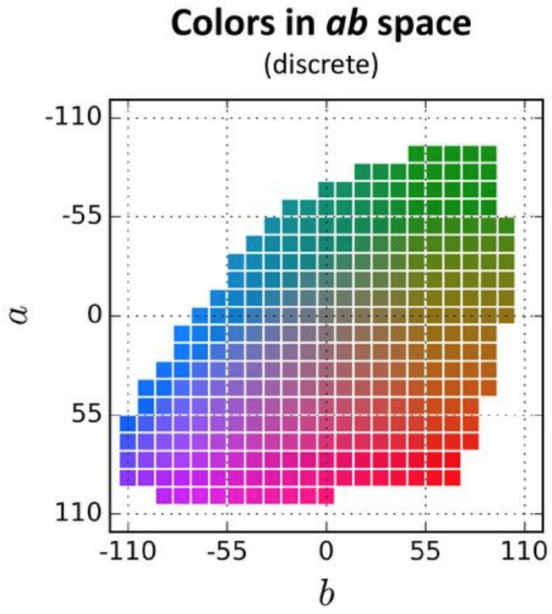


FIGURE 1 – 313 couleurs quantifiées dans l'espace ab.

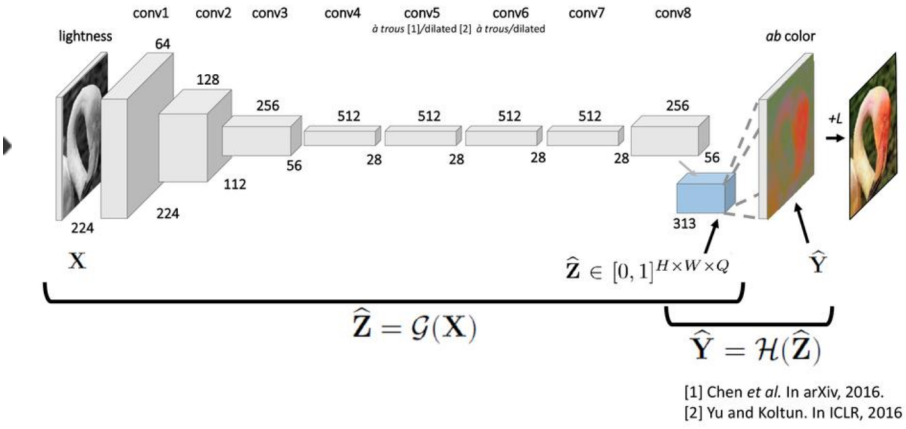


FIGURE 2 – Architecture CNN

2.3 Fonction de perte multinomiale avec rééquilibrage des couleurs

Tous les réseaux de neurones sont formés en définissant une fonction de perte. L'objectif de l'entraînement du réseau est de minimiser les pertes sur l'ensemble de d'entraînement. Dans le problème de la colorisation, les données d'apprentissage contiennent des milliers d'images en couleurs et leurs versions en niveaux de gris. La sortie du CNN est $\hat{Z}$ une image en couleur, étant donné une image d'entrée X noir et blanc. Nous devons transformer toutes les images couleur du jeu d'apprentissage en leurs valeurs Z correspondantes.
Mathématiquement, nous voulons simplement inverser le mapping H

$$Z = H^{-1}(Y)$$

Pour chaque pixel, $Y_{h,w}$, d'une image de sortie Y, nous pouvons simplement trouver la valeur quantifiée ab la plus proche et représenter $Z_{h,w}$ sous la forme d'un vecteur one-hot, dans lequel nous affectons 1 à la valeur de ab, et 0 aux 312 autres valeurs. Cependant, pour un meilleur résultat, les 5 voisins les plus proches sont pris en compte et une distribution gaussienne est utilisée pour calculer la distribution $Z_{h,w}$ en fonction de la distance par rapport à la ground-truth (couleur de l'image originale en couleur).

Dans la plupart des CNN réalisés pour le traitement d'images, la fonction de perte de cross-entropie est assez courante :

$$L(\hat{Z}, Z) = -\frac{1}{HW} \sum_{h,w} \sum_q Z_{h,w,q} log(\hat{Z}_h, w, q)$$

Cependant, ce type de fonction de perte produit souvent une colorisation assez "froide", ce qui est souvent dû aux données d'entraînement. En effet, les images tirées d'ImageNet ont souvent des couleurs assez centrées vers le gris.

De ce fait, une autre fonction de perte a été employée, pour obtenir des images plus colorées :

$$L(\hat{Z}, Z) = -\frac{1}{HW} \sum_{h,w} v(Z_{h,w}) \sum_q Z_{h,w,q} log(\hat{Z}_{h,w,q})$$

3 Méthodologie du travail réalisé

Nous avons, dans un premier temps voulu voir la performance du modèle tel qu'il est proposé par les auteurs de l'article. Nous avons donc téléchargé leur code ainsi que leurs modèles, qui sont réalisés avec Caffe.
Caffe demandant une installation conséquente, et étant assez compliqué à installer, nous avons réalisé un script capable de lire le modèle Caffe sans avoir installé la librairie, afin de pouvoir utiliser leur modèle.
Les résultats obtenus sont très réalistes, et certains vont être montrés dans la partie Démonstration.
Le script permettant de coloriser des images est bien sûr fourni dans notre code.

Nous avons ensuite voulu implémenter nos propres modèles avec des librairies que nous connaissions mieux, i.e. Keras et PyTorch, afin d'essayer de reproduire leurs résultats. Le but final que nous nous étions donné était de recréer les

(a) Original image (b) Output image

FIGURE 3 – Original and output images in the pytorch CNN

modèles, et de voir, si avec un plus petit entraînement, les résultats restaient satifaisants.

Cependant, au vu de la quantité d'entraînement necéssaire pour avoir une différence visible au niveau des résultats, nous avons décidé de commencer par essayer de répliquer leurs résultats en chargeant les poids que les auteurs de l'article fournissent avec leur modèle.

Nous nous sommes donc chacun lancé dans la création d'un modèle, une personne avec Keras, et l'autre avec PyTorch. Nous avons donc essayé de reproduire au mieux leur implémentation, mais nous sommes retrouvés bloqués par des différences entre les différentes librairies. En effet, et malheureusement pour nous, l'implémentation des couches de BatchNorm, effectuant une normalisation des données, était assez différente dans Caffe et dans Keras et PyTorch.

Ainsi, les résultats que nous avons obtenu avec nos réseaux, sont soit une image qui reste en nuances de gris, ou bien qui prennent une sorte de teinte cuivrée, mais donnant toujours l'impression d'avoir un seul channel de couleur.

4 Démonstration

4.1 PyTorch implementation

Nous disposons d'un script pytorch_imp.py qui permet d'exécuter le modèle que nous avons créé avec pytorch (à condition d'avoir pytorch installé).
Il suffit d'exécuter la commande suivante :
python pytorch_imp.py cheminImageAColoriser cheminOutput
Si le dernier argument n'est pas précisé, le résultat ira dans un fichier outfile.jpg dans le répertoire courant.

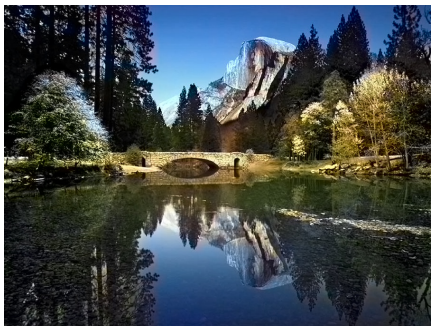
Dans l'état actuel du réseau, nous ne voyons pas de différence entre l'image originale et l'image de sortie du réseau, ce qui est assez étonnant compte tenu de toute les opérations effectuées à l'intérieur de celui-ci. Les deux images sont visibles dans la figure 3.

(a) Original image

(b) Output image

FIGURE 4 – Original and output images in the Keras CNN

(a) Original image



(b) Output image

FIGURE 5 – Original and output images in the caffe CNN

4.2 Keras implementation

Nous disposons d'un script Keras.py qui permet d'exécuter le modèle que nous avons créé avec Keras (à condition d'avoir Keras installé).
Il suffit d'exécuter la commande suivante :
python Keras.py cheminImageAColoriser cheminOutput
Si le dernier argument n'est pas précisé, le résultat ira dans un fichier outfile.jpg dans le répertoire courant.

Dans l'état actuel du réseau, nous voyons une petite différence entre l'image originale et l'image de sortie du réseau. Nous pouvons constater que notre résultat n'est pas satisfaisant. Le niveau de colorization sur l'image ne nous parait pas naturel et fiable pour que cette colorization soit acceptable dans la figure 4.

4.3 Reproduction par OpenCV

Nous disposons d'un autre script OpenCV.py qui permet d'exécuter le modèle que les auteurs de l'article ont créé en Caffe (Sans condition d'avoir Caffe installé).
Il suffit d'exécuter la commande suivante :
python OpenCV.py cheminImageAColoriser cheminOutput
Si le dernier argument n'est pas précisé, le résultat ira dans un fichier outfile.jpg dans le répertoire courant.

La même image que nous avons choisie en passant par le modèle orignal dans la figure 5.

5 Résultats

Nous avons globalement été assez déçus par les résultats que nous avons obtenu en essayant de reproduire leur réseau, et les tentatives que nous avons fait ne nous ont pas permis d'arriver au résultat que nous espérions. Nous avons cependant beaucoup appris, aussi bien en terme d'outils utilisés dans le Deep Learning pour le traitement d'image, qu'en terme de méthodes généralement employées.
Nous pensons tout de même, au vu de ce nous avons finalement obtenu, que ces méthodes de Deep Learning, bien que pouvant être très efficaces, sont très peu interprétables et difficiles à reproduire compte tenu de toutes leurs spécificités.
Un exemple de colorisation sur plusieures images noires et blanches par le réseau décrit dans l'article est visible dans la figure 6.

6 Discussion / Critique

Nous avons mis beaucoup de temps, que ce soit en Keras, ou en PyTorch, pour obtenir la même architecture de CNN telle que décrite dans l'article. Une fois le réseau en place, nous avons essayé de l'utiliser et sommes tombés sur de nombreux problèmes. Après en avoir résolus quelques-uns, nous avons fini par avoir le plus gros problème, que nous n'avons pas réussi à résoudre : l'implémentation Caffe de la BatchNorm ayant pour but de normaliser les données est aseez différente de celle de Keras et PyTorch. Nous avons essayé différentes

solutions, mais ne sommes finalement pas parvenus à résoudre ce problème.
Il faudrait donc que nous entraînions nous même nos modèles depuis le tout début, pour voir à quel point nous sommes en mesures de reproduire leurs résultats. Malheureusement, des contraintes matérielles ainsi que temporelles nous ont empêché d'y parvenir. En effet, ce type de d'énorme CNN necéssite des entraînements très longs sur d'énormes jeux de données, ainsi que des serveurs de GPUs dédiés, ressources que nous n'avons pas.

Références

[1] Richard Zhang, Phillip Isola, Alexei A. Efros. *Colorful Image Colorization.* University of California, Berkeley.
https ://arxiv.org/pdf/1603.08511.pdf

(a) Original image

(b) Output image

(c) Original image

(d) Output image

(e) Original image

(f) Output image

(g) Original image

(h) Output image

FIGURE 6 – Expériences en Caffe sur des images du TP