Étude comparative d'architecture de colorisation automatique

Souhaib Attaiki[1]*

Abstract
La colorisation est un fameux problème de traduction d'image à image. Dans cette étude, nous aborderons le problème de la colorisation d'une image en noir et blanc sans intervention humaine et de manière automatique. Pour ce faire, nous exploiterons la puissance des réseaux de neurones convolutifs (CNN). Nous explorons plusieurs architectures, fonctions objectives et formulations, en particulier, nous mettrons en œuvre et comparerons la formulation en tant que problème de régression et de classification. Les résultats obtenus seront présentés et comparés.

Keywords
Colorisation — CNN's — Deep learning — Computer vision

[1] *Ecole CentraleSupélec, Paris, France*
*souhaib.attaiki@student.ecp.fr

Contents

1 Introduction 1
1.1 Background . . . 1
1.2 Réseaux neuronaux convolutifs . . . 1
1.3 L*a*b* CIE color space . . . 2
1.4 Entrée/sortie du modèle . . . 2
2 Etat de l'art 2
2.1 Colorisation avec indice de coloration . . . 2
2.2 Colorisation entièrement automatisée . . . 2
3 Méthodologie 3
3.1 Fonction objective . . . 3
Modèle de régression • Modèle de classification
3.2 Pipeline général . . . 3
3.3 Codeur et décodeur . . . 4
3.4 Modèle de régression . . . 4
3.5 Modèle de classification . . . 4
4 Datasets 5
5 Résultats expérimentaux et discussion 6
5.1 Mise en place de l'expérience . . . 6
5.2 Réglage des l'hyperparamètres . . . 6
Modèle de régression • Modèle de classification
5.3 Métriques d'évaluation . . . 6
5.4 Résultats and discussion . . . 6
Conclusion 8
References 8

1. Introduction

1.1 Background

Le problème étudié est le suivant : à partir d'une image en noir et blanc, on veut halluciner une colorisation plausible sans qu'elle soit nécessairement similaire à la réalité de la vérité terrain, et qui peut potentiellement tromper un observateur humain.
Récemment, il y a eu beaucoup d'études concernant ce problème, qui n'est pas seulement intéressant d'un point de vue esthétique, mais qui a d'autres applications intéressantes qui peuvent aller de la restauration vidéo à l'amélioration des images pour une meilleure interprétation. La colorisation nous permet d'apprécier les vieilles images et nous aide à comprendre et à nous connecter avec notre histoire d'une manière jamais vue auparavant (nous pouvons citer pour cela le projet de National Geographic TV: APOCALYPSE WORLD WAR I [5]).
Colorer une image en noir et blanc est une tâche simple pour un humain, car il lui suffit de détecter la sémantique de l'image et des différents objets, et de se rappeler que le ciel est bleu et que l'arbre est vert. Cependant, ce n'est pas aussi simple que cela, car l'interprétation dépend de chaque personne, les objets peuvent prendre plusieurs couleurs (figure 1), et surtout, il faut beaucoup de temps qui peut durer jusqu'à un mois pour produire de bons résultats.
Comme notre but n'est pas de restaurer l'image originale, cette tâche devient plus réalisable, car il suffit d'adopter une approche qui permet de modéliser les dépendances statistiques entre la sémantique, la texture de l'image en noir et blanc et la version couleur afin de produire des résultats visuellement convaincants.

1.2 Réseaux neuronaux convolutifs

Puisque notre approche sera basée sur le mapping entre l'information sémantique de la scène, la détection des différents objets et de la vraie couleur, les réseaux convolutionnels neuronaux semblent être les premiers candidats pour cette tâche. En effet, au cours des dernières années, les CNN se sont distingués par leur capacité à résoudre les problèmes de classification des images en atteignant des taux d'erreur inférieurs

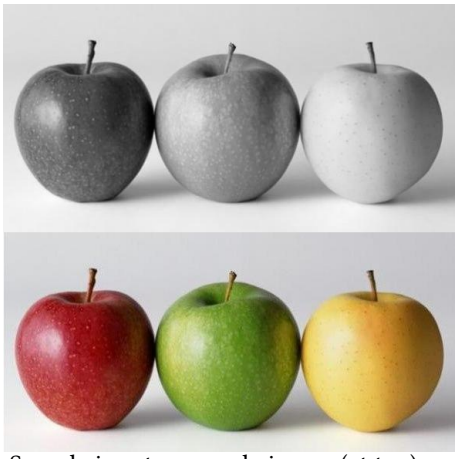

Figure 1. Sample input grayscale image (at top) and sample output colorized image (at bottom)

Figure 2. Image montrant les trois canaux d'une image codée dans l'espace couleur CIELAB, source: [9]

à 4% dans le challenge ImageNet [13].
Les CNN doivent une grande partie de leur succès à leur capacité d'apprendre et de discerner les couleurs, les motifs et les formes dans les images et de les associer à des classes d'objets.

1.3 L*a*b* CIE color space

L'espace chromatique L*a*b* CIE 1976, généralement nommé CIELAB, est un espace de couleur particulièrement utilisé pour la caractérisation des couleurs de surface. Trois grandeurs caractérisent les couleurs : la clarté L* dérive de la luminance de la surface ; les deux paramètres a* et b* expriment l'écart de la couleur par rapport à celle d'une surface grise de même clarté.
Les paramètres du système L*a*b* sont :

- la clarté L* qui prend des valeurs entre 0 (noir) à 100 (blanc de référence) ;
- le paramètre a* représente la valeur sur un axe vert ⟶ rouge ;
- le paramètre b* représente la valeur sur un axe bleu ⟶ jaune [7].

Comme le montre l'image 2, une image codée L*A*B* a une couche pour les niveaux de gris, et a compressé trois couches de couleur en deux, ce qui signifie que nous pouvons utiliser le canal L* en entrée de notre CNN. De ce fait, nous n'avons que deux canaux à prédire.

1.4 Entrée/sortie du modèle

L'approche utilisée consiste à entrainer un CNN avec des images en couleur, ce qui lui permettra d'apprendre la cartographie entre les caractéristiques et les couleurs sans aucune supervision humaine.
Plus spécifiquement, nous convertissons une image RGB vers l'espace couleur CIE Lab.Nous donnons en entrée à notre CNN le canal L*, et nous prédisons les canaux A* et B* (qui représentent les couleurs de l'image en entrée), puis nous concaténons l'entrée et la sortie pour générer l'image colorée.

2. Etat de l'art

La littérature du problème de la colorisation peut être divisée en deux catégories principales:

- Algorithmes classiques de vision par ordinateur qui nécessitent une intervention humaine pour spécifier les couleurs dans différentes régions de l'image (scribbling) ;
- Algorithmes entièrement automatisés basés sur les algorithmes de CNN.

2.1 Colorisation avec indice de coloration

La colorisation à base d'indice nécessite une supervision humaine pour effectuer la colorisation. Il existe deux méthodes populaires de colorisation basée sur des indications : la méthode basée sur le gribouillage (scribbling) et la méthode de transfert de couleur.

Les méthodes basée sur le gribouillage, introduites par Levin et al.[12], nécessitent la spécification manuelle des couleurs désirées de certaines régions. Ces couleurs de gribouillis sont propagées en supposant que les pixels adjacents avec une luminance similaire devraient avoir une couleur similaire, l'optimisation s'appuyant sur la méthode "Normalized cuts". Les utilisateurs peuvent affiner les résultats de manière interactive grâce à des gribouillis supplémentaires. Fig. 3 est un exemple tirée de l'article original.

Les méthodes basées sur le transfert reposent sur la disponibilité d'images de référence connexes, à partir desquelles la couleur est transférée à l'image en niveaux de gris cible. La cartographie entre la source et la cible est établie automatiquement, en utilisant les correspondances entre les descripteurs locaux [1], ou en combinaison avec une intervention manuelle [2]. Par exemple dans Fig. 4, les deux images ont des paysages naturels, alors le modèle apprendra à colorier l'image en niveaux de gris avec la couleur du ciel et de la montagne dans l'image référencée.

2.2 Colorisation entièrement automatisée

Malgré la qualité des résultats, les modèles basés sur les indices nécessitent encore beaucoup de travail humain pour générer des indices. Ainsi, des modèles de colorisation entièrement automatisés qui ne nécessitent que des niveaux de gris sont proposés. Ces algorithmes basés sur des CNNs peuvent être divisés en deux catégories principales : ceux qui

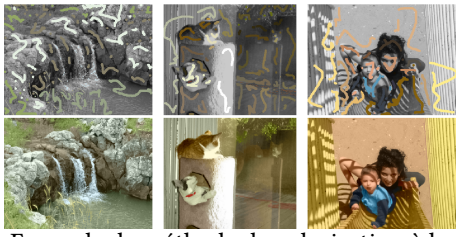

Figure 3. Exemple de méthode de colorisation à base de gribouillis

Figure 4. Exemple de méthode de colorisation par transfert

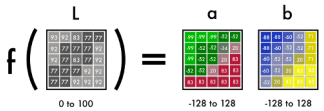

Figure 5. La fonction prédite par le modèle, source: [9]

considèrent le problème comme un problème de régression et ceux qui le considèrent comme un problème de classification.

L'architecture proposée par Iizuka et al. [10] est considéré comme l'état de l'art pour les problèmes de colorisation formulés comme problème de régression. Ils ont traité la colorisation comme un problème de régression avec une fonction de coût L2. Leur architecture se compose de quatre composants principaux, d'un réseau d'entités de bas niveau, d'un réseau d'entités de niveau intermédiaire, d'un réseau global d'entités et d'un réseau de colorisation. Les composants sont tous étroitement couplés et entraînés de bout en bout. La sortie du modèle est la chrominance de l'image (canaux A* et B*) qui est fusionnée avec la luminance pour former l'image de sortie.

D'autre part, le modèle "Colorful Image Colorization" proposé par Zhang et al. [16] est considéré comme le modèle le plus efficace pour les tâches de colorisation automatique. Ils ont considéré la colorisation des images en niveaux de gris comme un problème de classification en définissant un certain nombre de classes pour chaque pixel, et ont conçu une fonction objective appropriée qui gère l'incertitude multimodale du problème de colorisation et capture une grande diversité de couleurs.

Dans ce qui suit, nous nous intéresserons aux algorithmes de colorisation entièrement automatisés. En particulier, nous mettrons en œuvre et comparerons un modèle CNN basé sur la régression et un modèle CNN basé sur la classification.

3. Méthodologie

Dans ce projet, nous allons tester deux modèles de colorisation :

- Modèle de régression : pour ce modèle, nous avons implémenté le modèle fourni by Iizuka et al. dans [10];
- Modèle de classification : pour le modèle de classification, notre modèle s'est inspiré en partie de Zhang et al. dans [16] et R.Dah dans [3]

Les détails des deux modèles seront présentés ci-après.

3.1 Fonction objective

Nous considérons le problème de colorisation comme un problème d'apprentissage de la fonction $f : L \to Y$ (voir Fig. 5). Etant donné une image en noir et blanc $x \in L = [0,99]^{w \times h \times 1}$, la fonction f prédit la couleur $y \in Y$. (nous dénotons les prédictions avec un $\widehat{.}$ et la vérité terrain sans).

3.1.1 Modèle de régression

Dans le modèle de régression, nous avons utilisé la fonction de coût L2 entre les couleurs prédites et les couleurs de vérité terrain :

$$L_2(\widehat{y}, y) = \frac{1}{2}\Sigma_{h,w,c}\|y_{h,w,c} - \widehat{y_{h,w,c}}\|_2^2$$

(h indique la hauteur, w indique la largeur, c indique le canal) Dans ce cas, $Y = [0,255]^{h \times w \times 2}$. Cependant, cette fonction de coût est critiquée parce qu'elle n'est pas robuste face à l'ambiguïté héritée et à la nature multimodale du problème de colorisation. Si un objet peut prendre un ensemble de valeurs A*B* distinctes (comme les pommes dans Fig. 1), la solution optimale à la fonction de coût L2 sera la moyenne de l'ensemble. Dans la prédiction des couleurs, cet effet de moyennage favorise les résultats grisâtres et désaturés. [16]. C'est pour cette raison que la méthode de classification est introduite.

3.1.2 Modèle de classification

Dans cette approche, chacun des canaux A* et B* est divisé en 64 emplacements (bin) uniformément espacés, chaque emplacement représente une classe de sortie. La classe de chaque pixel est prédite. Lors de la phase d'apprentissage, nous formons le réseau à l'aide de la fonction d'activation softmax pour laquelle la distribution attendue est générée à partir de la vérité terrain. La fonction de coût est définie comme suit :

$$L_{cl}(\widehat{y}, y) = \Sigma_{h,w,c,q} y_{h,w,c,q} - \log \widehat{y_{h,w,c,q}}$$

Dans ce cas, $Y = [0,1]^{h \times w \times K \times 2}$. ($K$ indique le nombre de classes, ce qui est dans notre cas 64)

3.2 Pipeline général

Pendant la phase de formation, notre programme reçoit $h \times w$ images en couleurs au format RGB. celles-ci sont converties en espace CIE LAB. Le canal de luminance L* est donné en entrée au programme, tandis que les canaux A*B* sont transformés (normalisés pour le modèle de régression, et transformés en classes pour le modèle de classification) et donnés comme cible.

Pendant la phase de test, le modèle reçoit une image en noir et blanc $h \times w \times 1$, et il génère deux tableaux correspondant aux canaux A*B*. Ces trois canaux sont ensuite concaténés ensemble et transformés en espace couleur RGB pour former l'image colorisée.

3.3 Codeur et décodeur

Puisque pour la tâche de colorisation, la taille de l'image d'entrée et de sortie sont les mêmes, l'utilisation des CNN est délicate. Les couches max-pooling dans les réseaux de classification augmentent la densité d'information, mais déforment aussi l'image. Il ne valorise que l'information, mais pas la mise en page de cette dernière

Dans les réseaux de colorisation, nous utilisons plutôt une architecture appelée Encodeur/Décodeur. L'objectif du décodeur est d'extraire les informations nécessaires pour déterminer les couleurs, pour cela nous utilisons des strides de taille 2 pour diminuer la largeur et la hauteur de 50%, ce qui augmente aussi la densité d'information mais ne déforme pas l'image. L'encodeur vise à résumer l'information extraite par le décodeur pour prédire la couleur, et aussi à utiliser "conv2d transpose" pour suréchantillonner l'image et lui redonner sa forme originale.

3.4 Modèle de régression

Pour le modèle basé sur la régression, nous avons choisi d'implémenté le modèle proposé par Iizuka et al. [10]. La structure du réseau neuronal est montrée dans la Fig.6 (l'image est extraite du papier original).

Cette architecture utilise une approche novatrice pour fusionner les caractéristiques globales et locales. Les caractéristiques globales agissent comme une image avant les caractéristiques locales pour indiquer de quel type d'image est l'entrée. Par exemple, si les caractéristiques globales indiquent qu'il s'agit d'une image d'intérieur, les caractéristiques locales seront biaisées pour ne pas essayer d'ajouter des couleurs de ciel ou d'herbe à l'image, mais plutôt d'ajouter des couleurs convenant aux meubles [10]. Ce modèle entremêle à la fois un réseau global de fonctions d'image, similaire à ceux qui rivalisent dans les tâches de classification d'image (hors de la portée de notre étude), avec un réseau neuronal entièrement convolutif qui colore l'image. Afin d'améliorer l'efficacité du modèle, les deux réseaux utilisent un certain nombre de caractéristiques communes de bas niveau.

Activation Nous utilisons la fonction Relu comme la nonlinéarité qui suit chacun de nos couches convolutives. Mathématiquement, la fonction Relu est définie comme suit :

$$f(x) = max(0,x)$$

Nous utiliserons également la fonction de transfert Sigmoïde pour la couche de sortie couleur qui est définie comme suit :

$$f(x) = \frac{1}{1+e^{-x}}$$

Batch normalisation Ioffe et al. ont introduit la méthode de Batch normalisation comme moyen de réduire considérablement le temps de convergence de l'entraînement et d'améliorer la précision [14]. Pour notre modèle, nous avons placé une couche de batch normalisation après la sortie de chaque couche de non-linéarité, à l'exception de la dernière couche. Expérimentalement, nous avons constaté que cela accélère la convergence de l'algorithme.

Couche de fusion La couche de fusion vise à combiner les caractéristiques globales de l'image, un vecteur de 256 dimensions, avec les caractéristiques locales de l'image (de niveau intermédiaire), un volume h/8 × l/8 × l/8 × 256 dimensions. La sortie de la couche de fusion pour les coordonnées de niveau moyen (u,v) s'écrit comme suit :

$$y_{u,v}^{fusion} = \sigma\left(b + W\begin{bmatrix} y^{global} \\ y_{u,v}^{mid} \end{bmatrix}\right)$$

ou $y_{u,v}^{fusion} \in \mathbb{R}^{256}$ est la couche fusionnée à (u,v). σ est la fonction d'activation Relu. Les W et b sont des paramètres du réseau.

Optimization Le choix des taux d'apprentissage est un hyperparamètre très important pour la convergence de l'algorithme, ce taux d'apprentissage doit aussi être progressivement diminué au fur et à mesure que le modèle est enseigné. Puisqu'il n'y a pas de pratiques pour les réseaux de colorisation, nous avons choisi d'utiliser un optimiseur qui fixe de façon adaptative les taux d'apprentissage pour tous les paramètres du réseau sans nous obliger à fixer un taux d'apprentissage global. Adadelta est choisis pour ses bonnes propriétés de convergence [15].

3.5 Modèle de classification

Notre modèle basé sur la classification a été inspiré en partie par Zhang et al. dans [16] (en particulier, l'idée de créer des classes à partir de l'image originale) et R.Dah dans [3] (en particulier, l'idée d'utiliser le fine-tuning pour les tâches de colorisation). Notre architecture se compose de deux parties principales : le décodeur et l'encodeur.

Le codeur se compose des cinq premiers blocs de VGG16 pré-entraînés sur ImageNet Dataset (1,2 million d'images classées en 1000 catégories).

Le décodeur se compose de 6 couches avec un nombre décroissant de filtres. Chaque couche de décodage est composée d'une transposition conv2d pour le upsizing, et de fonctions d'activation ReLu, à l'exception de la dernière couche qui a une activation softmax. La structure du modèle est illustrée dans la Fig.7.

VGG est un CNN développé par un groupe d'universitaires d'Oxford, la partie convolutive n'utilise que des filtres de taille réduite (3*3) ainsi que des poolings de faible dimension (2*2). Cela le rend particulièrement efficace pour le traitement des images, en réduisant le nombre de paramètres pour chaque champ récepteur. L'autre caractéristique majeure de ce type de réseau est sa grande profondeur (16 couches pour

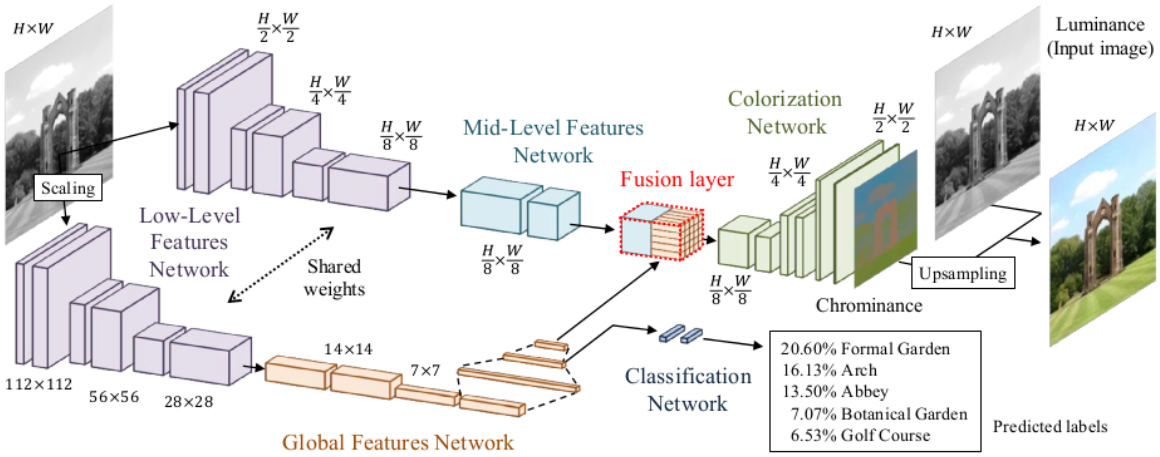

Figure 6. Vue d'ensemble du modèle proposé par Iizuka et al.

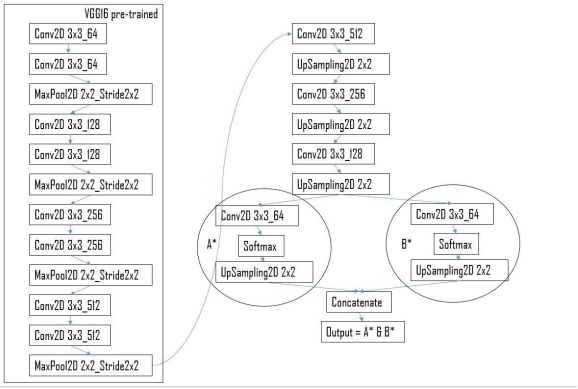

Figure 7. Architecture de notre modèle de classification

Figure 8. Exemples d'images de l'ensemble de données MIRFLICKR25K

Table 1. Nombre d'images d'apprentissage, de validation et de test dans les datasets utilisées.

Dataset	Training	Validation	Test
MIT CVCL open country	260	65	82
MIT CVCL street	127	38	39
MIRFLICKR	16000	4000	5000

le VGG16). Les nombreuses non-linéarités induites renforcent le caractère discriminatoire du réseau qui fait de lui un bon candidat pour notre encodeur.

Il est à noter que pendant la phase de formation, les paramètres de la VGG16 sont gelés, nous nous concentrerons donc uniquement sur l'apprentissage du décodeur.

4. Datasets

Datasets utilisées Les datasets utilisées pour tester nos modèles sont fournis dans le tableau no.1.

MIT CVCL dataset contient des images appartenant à 10 classes [4]. Nous avons utilisé cet ensemble de données pour prototyper nos modèles, tester les différents paramètres ainsi que la capacité des modèles à généraliser sur des images appartenant à la même classe (nous avons utilisé les classes "open country" ≈ 400 images et "street" ≈ 200 images).

L'ensemble de données MIRFLICKR comprend 25000 images Creative Commons téléchargées à partir du site communautaire de partage de photos Flickr [6]. Les images couvrent une vaste gamme de catégories, de styles artistiques et de sujets. Fig.8 montre des exemples de cet ensemble de données.

Prétraitement Nous effectuons un prétraitement pour chaque image avant de la donner au réseau.

Pour le modèle de régression, nous convertissons chaque image dans l'espace CIE L*A*B*, puis nous extrayons le canal L* qui servira d'entrée au modèle. A la sortie du modèle, nous concaténons les canaux A*B* prévus avec les canaux L* pour former l'image colorée prévue.

Pour le modèle de classification, puisqu'il est basé sur le réseau VGG16, nous redimensionnons chaque image à $224 \times 224 \times 3$. Et comme VGG16 a besoin de trois canaux d'entrée, nous dupliquons le canal L* trois fois. le reste est le même que pour la régression.

Augmentation des données Les techniques d'augmentation des données permettent d'enrichir l'ensemble des données disponibles, en particulier dans les cas où la quantité de données disponibles n'est pas suffisante pour former de grands réseaux.

Pour nos modèles, nous effectuons une transformation

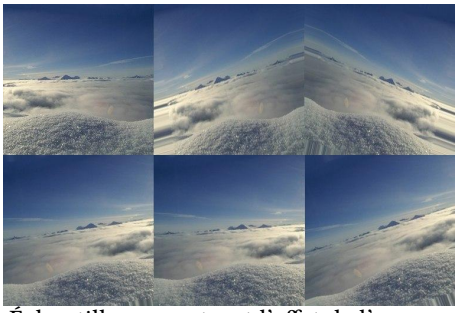

Figure 9. Échantillons montrant l'effet de l'augmentation des données

aléatoire des données avant chaque epoch. Il en résulte que nos modèles ne verront jamais les mêmes données. Des exemples de transformations sont montrés dans la figure Fig.9.

5. Résultats expérimentaux et discussion

5.1 Mise en place de l'expérience

Tous nos modèles sont implémentés sous python3 en utilisant la bibliothèque Keras (en utilisant le backend TensorFlow). Les manipulations et transformations d'images sont effectuées à l'aide des bibliothèques d'images OpenCV et Scikit.

Le code source des différents modèles est hébergé sur notre repository Github [8]. Il est à noter que nous avons implémenté tous les modèles à partir de zéro, soit parce qu'il n'y a pas de mise en œuvre disponible ("modèle de régression"), soit parce qu'il est disponible dans un autre language (Caffe2 pour le modèle de classification).

5.2 Réglage des l'hyperparamètres

Afin de faire converger nos modèles, nous avons effectué plusieurs tests pour déterminer les hyperparamètres.

5.2.1 Modèle de régression

Nous avons d'abord utilisé un optimiseur Adagrad avec les valeurs recommandées (rho = 0,95 et epsilon = 1e-7) et avec un taux d'apprentissage égal à 1. Nous avons remarqué que notre algorithme atteint un niveau qui ne peut pas être franchi. Pour résoudre ce problème, nous avons varié la taille du batch et le taux d'apprentissage. Une taille de batch de 4 et un taux d'apprentissage = 0,1 aide à franchir ce pallier. L'ajout d'un réducteur de taux d'apprentissage (le taux d'apprentissage est réduit après un certain nombre de batch si le résultat ne s'est pas amélioré) a encore optimisé la fonction de coût.

5.2.2 Modèle de classification

Nous avons éprouvé des difficultés à faire converger le modèle de classification, en particulier le fait qu'il faut beaucoup de temps de calcul par epoch par rapport au modèle de régression a limité le nombre d'expériences que nous avons effectuées. Au début, nous avons utilisé un optimiseur SGD avec un taux d'apprentissage de 1e-3 puisque nous utilisons le fine tuning, et il est conseillé de faire ainsi. Après plusieurs tentatives, nous avons constaté qu'une taille de batch de 8 et un taux d'apprentissage de 1 nous permettait de converger plus rapidement. Cependant, notre algorithme atteint un palier qu'il ne peut plus franchir (similaire au modèle de régression). Pour adresser cela, nous avons rendu les poids du dernier bloc du VGG16 variables, et donc l'algorithme les apprendra. Couplé à un optimiseur Adam (taux d'apprentissage = 0,001) a considérablement amélioré le résultat. Nous avons essayé d'utiliser des couches de batch normalisation mais cela n'a pas donné de résultats significatifs.

5.3 Métriques d'évaluation

Afin d'évaluer les performances de nos modèles, nous avons utilisé les métriques suivantes :

Précision de la classification cette métrique nous permet de comparer les résultats des deux modèles. nous mesurons la proximité de l'image générée par rapport à l'image réelle par le pourcentage de valeurs de pixels dont la classe a été bien prédite (nous avons utilisé le même nombre de classe que dans le modèle de classification) ;

Mathématiquement, cela est donné par :

$$Accuracy = \frac{1}{N}\Sigma_N \mathbb{1}_{[bin(predicted)=bin(actual)]}$$

Test de colorisation Turing Le test ultime de la colorisation est de savoir à quel point les images sont fascinantes pour l'observateur humain. L'objectif est de savoir si un humain peut distinguer entre une image automatiquement colorée et la vérité terrain. Malheureusement, nous n'avons pas été en mesure de déployer ce test à grande échelle.

5.4 Résultats and discussion

Modèle de régression Les résultats obtenus par notre modèle de régression sont présentés dans la Fig.10. Comme prévu, les résultats obtenus par le modèle de régression sont désaturés et les couleurs ne sont pas vives, ce qui n'est pas attrayant pour un observateur humain. Cependant, le modèle a été capable de détecter les contours des scènes (on peut voir par exemple que seule l'herbe est colorée en vert), ce qui donne une colorisation propre.

Modèle de classification Les résultats obtenus par notre modèle de régression sont présentés dans la Fig.11. Contrairement au modèle de régression, les couleurs obtenus par le modèle de classification sont plus vifs et agréables et, en général, plus attrayantes que le modèle de régression.

Cependant, ce modèle ne parvient pas à bien détecter les contours (on remarque par exemple qu'une partie de la montagne est colorée en vert). Cela est probablement dû à la discrétisation de l'espace pendant l'apprentissage. Il est également possible que cela soit dû au fait que le modèle fait la classification en prenant en compte chaque pixel séparément, et donc, le modèle ne parvient pas à prendre en

Figure 10. Exemples de résultats obtenus par modèle de régression : image de vérité terrain (colonne de gauche), image d'entrée du réseau (colonne centrale) et sortie du réseau de régression (colonne de droite).

Figure 11. Exemples de résultats obtenus par modèle de classification : image de vérité terrain (colonne de gauche), image d'entrée du réseau (colonne centrale) et sortie du réseau de régression (colonne de droite).

Table 2. Résultats du test classification accuracy pour MIT CVCL dataset

Modèle	Accuracy %
Modèle de régression	13.2
Modèle de classification	9.7

compte le contexte de l'image. Enfin, cela peut aussi être dû au fait qu'un timbre particulier contenant une forme ou un motif a de nombreuses correspondances de couleurs possibles dans le monde réel et que le système n'a pas la capacité de sélectionner une couleur particulière de manière cohérente.

Comparaison Pour comparer nos deux modèles, nous avons utilisé la métrique que nous avons spécifiée précédemment. les résultats sont résumés dans le tableau n° 2.

Contrairement à ce qui était prévu, nous constatons que le modèle de régression était plus efficace que le modèle de classification. Ceci peut être justifié par le fait que nous avons utilisé une discrétisation trop petite (64 bins), mais aussi par le fait que nous avons une capacité de calcul plutôt limitée, ce qui n'a pas permis au modèle de classification d'optimiser ses paramètres.

Test de colorisation Turing Nous avons effectué un test de colorisation de Turing avec trois élèves de CentraleSupélec où nous leur avons montré des images aléatoires et leur avons demandé de prédire si ces images étaient originales ou créées par notre modèle. Ils ont deux secondes pour voir l'image et trois secondes pour répondre. Pour évaluer ce test, nous calculons le pourcentage du nombre de fois que les élèves ont été dupés. Nous avons atteint un ratio de 7% (l'état de l'art est de 33%). Ceci tout à fait normal, étant donné que notre modèle est appris sur la base d'un petit ensemble de données et que notre capacité de calcul est limitée ce qui n'a pas permis de bien optimiser les paramètres.

Conclusion et pistes d'améliorations

Ce projet a été pour nous l'occasion de nous confronter à la réalité de l'état de l'art en ce qui concerne l'utilisation des CNN pour effectuer des tâches de colorisation automatique.

Grâce à nos expériences, nous avons démontré l'efficacité et le potentiel de l'utilisation des réseaux neuronaux convolutionnels profonds pour coloriser des images en noir et blanc.

Afin d'améliorer nos modèles, nous avons identifié plusieurs pistes. Pour adresser le problème de l'inconsistance de la col-

oration pour le modèle de classification, on peut envisager une modélisation de classe qui prend en compte l'interaction des deux canaux A* et B*. Par exemple, au lieu de discrétiser chaque canal séparément, on peut envisager de discrétiser le plan formé par le produit des deux canaux. Les prédictions de couleurs peuvent être améliorées en considérant une fonction de passage des classes prédites aux canaux A*B* qui prend en compte la distribution des couleurs dans l'espace. qui est non uniforme comme le montre [16]. Un dernier axe d'amélioration est redéfinir le système autour d'un réseau adversarial GAN, car au lieu de se concentrer sur la minimisation de la cross-entropy, le système apprendrait à générer des images qui se comparent bien avec les images du monde réel.

References

[1] Hofmann M.-Sch¨olkopf-B. Charpiat, G. Automatic image colorization via multimodal predictions. 2008.

[2] Zhuo S.-Gupta-R.K.-Tai-Y.W.-Cho-S.Y.-Tan P. Lin S. Chia, A.Y.S. Semantic colorization with internet images. 2011.

[3] R. Dah. Automatic colorization. 2016.

[4] http://cvcl.mit.edu/database.htm.

[5] http://natgeotv.com/asia/apocalypse-world-war i/about.

[6] http://press.liacs.nl/mirflickr/.

[7] https://fr.wikipedia.org/wiki/LabCIE1976.

[8] https://github.com/pvnieo/Deep Levi.

[9] https://medium.freecodecamp.org/colorize-b-w-photos-with-a-100-line-neural-network 53d9b4449f8d.

[10] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4), 2016.

[11] You Zhou Jeff Hwang. Image colorization with deep convolutional neural networks. 2016.

[12] Lischinski D.-Weiss-Y. Levin, A. Colorization using optimization. 2004.

[13] H. Su-J. Krause-S. Satheesh-S. Ma Z. Huang A. Karpathy A. Khosla M. Bernstein et al. O. Russakovsky, J. Deng. Imagenet large scale visual recognition challenge. 2015.

[14] Christian Szegedy Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[15] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. 2012.

[16] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. 2016.