HEC MONTRÉAL

Colorisation d'images à l'aide d'apprentissage profond

Travail présenté à Madame Mina Arzaghi

Dans le cadre du cours

Apprentissage automatique I (MATH 60629)

Par :

Dorothée BEAUDET — 11321160

Krystel LAUDON — 11322759

Colin LLACER — 11318002

Heddier Alberto SOLER — 11272957

Automne 2023

Le jeudi 07 décembre 2023

1. Description de la question

Dans le vaste domaine de la vision par ordinateur et du traitement d'image, l'une des questions les plus intéressantes et pourtant peu exploitée est la capacité de transformer des images en noir et blanc en images colorisées. La conversion d'images couleur en noir et blanc est un processus relativement simple et direct, car il implique la réduction de trois canaux de couleur en une seule dimension représentant des nuances de gris. Une formule simple existe pour des images utilisant l'espace de couleur RGB :

$$Y = 0.299\ R + 0.587\ G + 0.114\ B$$

Cependant, l'opération inverse, c'est-à-dire la colorisation, est un défi nettement plus complexe. La complexité réside dans le fait qu'un niveau de gris donné peut correspondre à des milliers de combinaisons de valeurs de couleur dans l'espace RGB. En d'autres termes, passer d'une unique dimension (niveaux de gris) à trois dimensions distinctes (rouge, vert, bleu) n'est pas une conversion directe, et nécessite une compréhension approfondie des patterns et du contexte afin de choisir la bonne couleur pour chaque pixel de l'image.

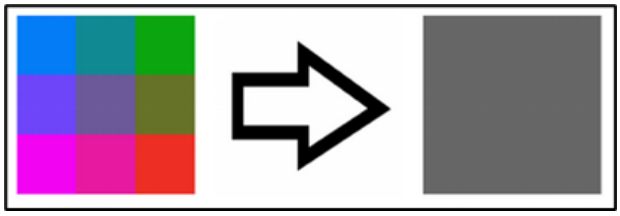

Figure 1 : Conversion RVB vers niveau de gris

Notre projet vise à explorer cette question en utilisant les avancées récentes en apprentissage profond. Plus précisément, nous nous concentrons sur la création d'un modèle de vision par ordinateur capable de coloriser avec précision des images en noir et blanc. Pour ce faire, notre modèle va détecter les patterns, textures et structures inhérentes à ces images. Ce défi requiert non seulement une compréhension technique des modèles de réseaux de neurones, mais aussi une appréciation artistique des nuances subtiles de la couleur et de la lumière.

L'objectif est de développer un système qui, en recevant une image en noir et blanc, peut prédire et appliquer une palette de couleurs réaliste, transformant ainsi une image monochrome en une image colorisée crédible. Un tel modèle pourrait avoir des applications dans la restauration de vieilles photographies ou dans le monde du cinéma pour coloriser des films anciens.

Notre première approche pour aborder cette question complexe implique l'utilisation d'un autoencodeur de type U-Net, un modèle d'apprentissage profond particulièrement adapté aux tâches de segmentation d'image et qui est habituellement relativement performant dans des tâches de traduction image à image. Par la suite, nous explorerons les réseaux antagonistes génératifs, afin d'améliorer la performance de notre U-net et d'obtenir une colorisation plus convaincante.

2. Revue de littérature

La revue de littérature pour ce travail concerne principalement des méthodes standards servant à la colorisation d'images. Ces méthodes sont celles qui seront utilisées dans la réalisation du projet. Il s'agit donc de la méthode U-net ainsi que de la méthode du GAN avec ses variations.

2.1 U-Net

Ronneberger, Fischer et Brox (2015) ont introduit le U-Net comme tentative de réponse à des défis rencontrés dans le domaine de traitement d'images biomédicales. Les méthodes précédemment utilisées permettent la segmentation d'images, mais le contexte biomédical entraine un besoin crucial de localisation des zones problématique. Ainsi, chaque pixel doit se faire attribuer une classe, ce qui n'était pas possible avec les méthodes courantes. Des tentatives avaient été d'ailleurs effectuées afin de répondre à cette problématique avant cet article. Ciresan, Gambardella et Giusti (2012), par exemple, ont tenté une méthode qui prédit la classe d'un pixel en utilisant en entrée un « patch » autour de ce pixel. Cependant, les méthodes précédentes le U-Net présentaient des limitations en termes de vitesse de traitement et de redondance dues à la superposition des « patchs » (Ronneberger, Fischer, & Brox, 2015). Ronneberger, Fischer et Brox (2015) ont donc introduit une modification de l'architecture de réseau convolutif, adoptant une approche en forme de « U ». Cette adaptation visait à surmonter les limitations tout en permettant une segmentation plus précise des images, même avec un ensemble de données limité, cette situation étant courante dans le domaine biomédical.

Plusieurs variations de la méthode U-net existent (Siddique, Sidike, Elkin, & Devabhaktuni, 2021). Toutefois, celles-ci sont toutes composées d'une structure de base. En effet, le U-net se base sur la structure d'un auto-encodeur. Ainsi, la structure de base d'une architecture U-net se compose de deux chemins. D'abord, il y a l'encodeur qui représente le chemin de contraction (Ronneberger, Fischer, & Brox, 2015). Il s'agit du processus conventionnel d'une série de convolutions (Siddique, Sidike, Elkin, & Devabhaktuni, 2021). Ensuite, il y a le chemin d'expansion, avec un décodeur composé de convolutions ascendantes et de

concaténations avec des caractéristiques du chemin de contraction. Cette expansion permet au réseau d'apprendre des informations de classification localisées. Ainsi, chaque étape agrandit la carte des caractéristiques en utilisant une up-convolution de 2 × 2. Ensuite, la carte des caractéristiques de la couche correspondante dans le chemin de contraction est rognée et concaténée sur la carte des caractéristiques agrandie, à l'aide de « skip connections ». Cela aboutit à une propagation des informations contextuelles le long du réseau, ce qui lui permet de segmenter des objets dans une zone en utilisant le contexte d'une zone plus grande et chevauchante. La figure 1 illustre l'architecture globale d'U- l'architecture globale d'U-net.

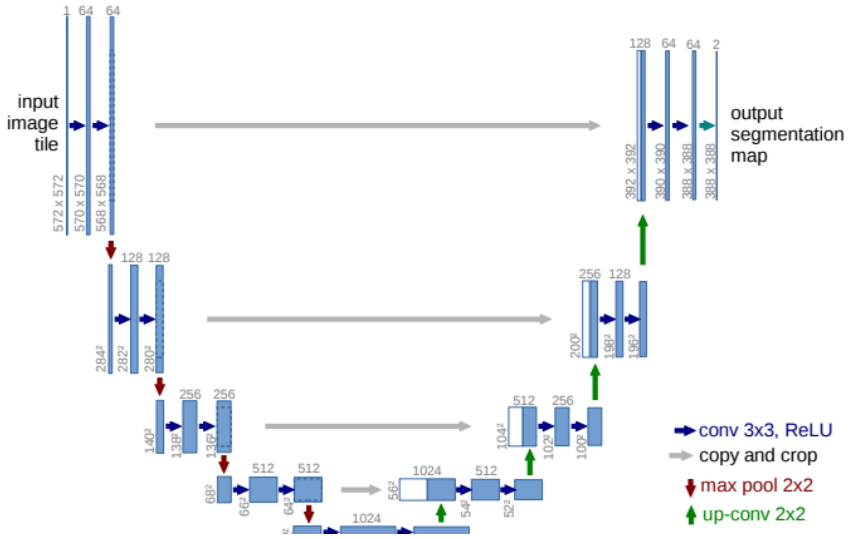

Figure 2. Structure du U-Net selon Roneberger et al. (2015)

2.2 GAN, Patch-GAN et WGAN-GP

Goodfellow *et al.* (2014) ont introduit la méthode GAN en réponse aux difficultés engendrées par les modèles génératifs profonds habituels. En effet, ceux-ci impliquent souvent des calculs probabilistes complexes qui peuvent être difficiles à résoudre analytiquement ou computationnellement. La méthode proposée par Goodfellow *et al.* permet de surmonter ces difficultés.

Les auteurs expliquent que le cadre proposé fonctionne sur le principe d'un jeu entre deux entités, un modèle génératif et un modèle discriminatif, qui s'engagent dans un jeu compétitif. D'une part, le modèle génératif s'efforce de créer des échantillons synthétiques qui imitent les données réelles. D'autre part, le discriminateur vise à distinguer entre les données réelles et générées. L'essence de cette configuration adversariale réside dans la compétition qui pousse les deux modèles à affiner continuellement leurs

méthodes. L'objectif est que le modèle génératif produise des échantillons si authentiques qu'ils deviennent indiscernables des données réelles, mettant à l'épreuve la capacité du modèle. La compétition entre ces deux modèles les pousse à s'améliorer continuellement, le modèle génératif produisant des faux de plus en plus convaincants et le modèle discriminatif devenant de plus en plus doué pour la détection. Le jeu est un jeu minimax parce que le générateur essaie de tromper le discriminateur en générant des données qui semblent aussi réelles que possible (maximisant la perte), tandis que le discriminateur essaie de classer correctement les données réelles et fausses (minimisant la perte) (Benet, 2021). L'équilibre de Nash est atteint lorsque le générateur produit des répliques parfaites des données réelles, et que le discriminateur ne peut pas distinguer entre les données réelles et fausses, produisant une probabilité de 0,5 pour chaque entrée (Benet, 2021).

La version de base du GAN introduite par Goodfellow *et al.* (2014) possède certaines limitations. En effet, les utilisations traditionnelles du GAN produisent des images qui sont souvent floues lors de l'étape de génération, car le modèle ne parvient pas à encourager la netteté à haute fréquence (Isola, Zhu, Zhou, & Efros, 2016). Par conséquent, comme l'explique Isola *et al.* (2016), plutôt que de développer un tout nouveau cadre pour assurer la précision des basses fréquences, l'approche est d'employer un discriminateur PatchGAN pour représenter la structure des hautes fréquences. Le discriminateur PatchGAN est élaboré dans le but de sanctionner la structure à l'échelle des patchs. Son rôle consiste à classifier si chaque patch N × N dans une image est authentique ou artificiel. Cette opération est réalisée à l'aide de convolutions sur l'ensemble de l'image, et les réponses obtenues sont mises en commun afin de générer la sortie finale. Le changement du discriminateur en un patchGAN permet donc d'améliorer la qualité d'image. En effet, cette version du GAN permet de détecter des structures de couleurs, des « patterns », au lieu de prédire des couleurs sur l'image globale.

En outre, un problème important du GAN est isolé dans Arjovsky, Chintal et Bottou (2017). En effet, il est connu que l'entrainement des GAN est délicat et instable, en raison de l'entrainement simultané du générateur et du discriminateur (Bhagyashree, Kushwaha, & Nandi, 2020). Pour réduire ce problème, les auteurs ont introduit les WGAN, qui résolvent les problèmes majeurs rencontrés lors de l'entraînement des GAN. Plus précisément, l'entraînement des WGAN ne nécessite pas le maintien d'un équilibre délicat entre le formateur du discriminateur et celui du générateur, et il n'exige pas non plus une conception minutieuse de l'architecture du réseau. Le phénomène de « disparition de modes », où le générateur tend à produire qu'une variété limitée d'échantillons (Bhagyashree, Kushwaha, & Nandi, 2020) et les problèmes de non-convergence, couramment observés dans les GAN, sont également considérablement réduits.

L'apport nouveau des auteurs est l'utilisation de la distance de Wasserstein, en imposant que le discriminateur soit une fonction 1-Lipschit, (Gulrajani, Ahmed, Arjovsky, Dumoulin, & Courville, 2017), notamment en limitant les poids des paramètres du discriminateur à un certain ensemble (Massart, 2022).

Finalement, bien que les WGANs sont une amélioration importante du modèle traditionnel, ils possèdent toujours certaines limites. En effet, la limitation des poids peut entrainer des comportements indésirables et des difficultés d'optimisation (Gulrajani, Ahmed, Arjovsky, Dumoulin, & Courville, 2017). Le modèle WGAN-GP, introduit par Gulrajani *et al.* (2017) propose une alternative à la limitation des poids en introduisant la pénalité de gradient. Les auteurs affirment que l'utilisation de la pénalité de gradient permet des performances accrues ainsi qu'une meilleure stabilité durant l'entrainement.

D'autres articles de littératures seront abordés plus en détail dans la section empirique, afin de comparer nos résultats aux essais précédents.

3. Description de nos modèles

3.1 U-net

Le modèle U-Net est une architecture de réseau de neurones convolutifs profonds. Sa structure caractéristique en forme de « U » est en fait identique à celle d'un autoencodeur, divisé en deux parties :

- Un encodeur qui réduit la dimension spatiale tout en augmentant la profondeur des caractéristiques. L'encodeur va tenter de reconnaître les patterns dans l'image et d'encoder cette information de la manière la plus compacte possible.
- Un décodeur qui effectue l'opération inverse et tente d'utiliser la représentation compressée pour récréer l'image souhaitée (ici l'image en couleur).

Cette architecture en forme de « U » est particulièrement efficace pour certaines transformations, mais n'est pas suffisante pour conserver les fins détails de l'image. En effet, nos essais initiaux en utilisant un simple autoencodeur nous ont rapidement indiqué que l'autoencodeur basique n'est pas capable de maintenir la résolution de l'image, qui devient floue au moment de la reconstruction. Ainsi, notre choix s'est plutôt porté sur un U-Net, qui permet des sauts entre couches lorsque la dimension des images est la même. Ces sauts permettent de conserver les fins détails dans l'image, puisque toute l'information n'est

pas systématiquement compressée. Garder les détails de l'image est essentiel dans notre cas, puisque nous ne devrions pas perdre en qualité après la colorisation. L'encodeur du modèle U-Net, spécifiquement adapté pour notre projet de colorisation, prend en entrée une image en niveaux de gris avec une dimension de 128x128 pixels. Par la suite, chaque couche successive est composée de :

- Filtres de convolution 3x3, qui deviennent de plus en plus nombreux au fil des couches. Ces filtres ont un padding pour conserver la dimension et permettent de détecter les patterns importants dans l'image.
- Une couche de normalisation par batch et un dropout allant de 10 à 20%, pour éviter le surajustement.
- Une fonction d'activation ReLU, pour introduire de la non-linéarité.
- Un MaxPooling est également ajouté, afin de réduire la dimension après chaque couche. Le modèle va progressivement se concentrer sur des caractéristiques plus globales de l'image.

Ainsi, l'information va être concentrée jusqu'à atteindre une dimension de 8x8 au centre du modèle. Nous aurions pu continuer à réduire la dimension en ajoutant des couches, mais nous avons observé que la performance du modèle tendait à baisser si la compression était trop importante, car trop d'information était perdue. Cette information va ensuite être donnée au décodeur, qui va tenter de récréer l'image avec.

Le décodeur quant à lui prend en entrée la représentation compacte (8x8) résultant de l'encodeur. Chaque couche est composée de :

- Un UpSampling, qui permet au décodeur de doubler la résolution de l'image au fur et à mesure des couches, à partir de la représentation existante.
- Très nombreux filtres de convolution (Conv2d), qui deviennent de moins en moins nombreux lorsque la dimension augmente. Cette étape permet de détecter les patterns dans les images et donc la reconstruction de l'image en couleur.
- Une concaténation avec la couche symétrique de l'encodeur, dans laquelle l'image est à la même dimension. Cette concaténation est la spécificité des U-Net et nous permet de maintenir les détails fins de l'image en permettant au modèle de sauter des étapes.

Puisque chaque couche double la dimension de l'image, nous avons besoin de quatre couches afin de retrouver la résolution initiale de 128x128. La dernière couche de convolution correspondant à la sortie utilise une fonction d'activation sigmoïde afin de finaliser la colorisation. Ainsi, la sortie du décodeur est

l'image colorisée au format RGB (3 canaux), avec une dimension identique à celle de l'entrée. Nous utilisons une fonction de perte combinée comprenant une transformation de la métrique SSIM (mesure de similitude, 70%) et l'erreur absolue moyenne (MAE, 30%) pour l'entraînement de ce modèle, qui vont comparer l'image générée à l'image réelle.

Cette approche avec un U-Net entièrement entrainé par nos soins s'est cependant avérée limitante. En effet, notre puissance de calcul et notre mémoire sont limitées, et notre échantillon n'est donc que d'un peu plus de 20 000 images. Ce nombre est insuffisant pour permettre à l'encodeur d'apprendre à reconnaître beaucoup de patterns pouvant être présents dans des photos de paysage. Dès lors, nous avons songé à utiliser un modèle pré-entrainé dans la partie encodeur de notre U-Net. Ainsi, notre modèle U-Net modifiée intègre le modèle pré-entraîné MobileNet en tant qu'encodeur. MobileNet est un modèle de réseau de neurones convolutif compact développé par Google. Il est connu pour sa performance élevée (classification) sur le dataset ImageNet, et ce malgré un nombre de paramètres restreint (afin de pouvoir être utilisé sur des appareils mobiles). En l'intégrant dans notre encodeur U-Net, nous bénéficions de certaines de ses couches convolutives pré-entraînées pour une extraction de caractéristiques plus efficace et robuste.

Cet ajout nous a cependant forcés à faire certaines modifications. Les images en niveaux de gris utilisées précédemment sont transformées pour s'adapter aux exigences d'entrée de MobileNet, qui demande trois dimensions. Chaque image en niveaux de gris est ainsi convertie en une image pseudo-RGB disposant de trois canaux identiques, grâce à la duplication des canaux de gris. Cette astuce nous permet d'utiliser efficacement MobileNet, qui est conçu pour des images RGB, tout en conservant nos images en niveaux de gris en entrée. Par ailleurs, nous avons conservé uniquement certaines couches du modèle MobileNet, puisque notre modèle doit rester symétrique et que notre décodeur reste inchangé.

3.2 GAN, Patch-GAN et WGAN-GP

Dans un but de perfectionnement de nos résultats, nous nous sommes ensuite intéressés aux réseaux antagonistes génératifs, les GAN. En effet, comme évoqué lors de la revue de littérature, les GAN sont désormais les modèles privilégiés pour ce qui est de la transformation image à image (avec les diffuseurs). La structure de notre GAN initial est relativement simple. Tout d'abord, nous réutilisons le U-Net avec MobileNet précédemment entrainé en tant que générateur. Cela permet d'éviter un départ à froid, et de diminuer le temps d'entraînement déjà particulièrement long. Bien que l'on puisse penser qu'un tel ajout

provoque un retard non rattrapable pour le discriminateur, ce n'est pas le cas en pratique puisque le discriminateur apprend assez rapidement. Notre inspiration concernant cette implémentation provient d'une publication sur un blog, ou l'ajout du modèle pré-entrainé avait grandement aidé à améliorer les résultats (Shariatnia, 2020). Puisque nous utilisons le GAN pour affiner la performance de ce générateur initial, son taux d'apprentissage est maintenu beaucoup plus faible que celui du discriminateur. Ce taux d'apprentissage plus faible nous permet également de dégeler certaines couches de l'encodeur, ce qui ruinait complètement la performance du U-Net précédemment.

Le discriminateur quant à lui possède une structure relativement classique. Nous avons plusieurs couches de convolutions (5, avec un nombre de convolutions qui double à chaque fois), toujours suivies d'une normalisation (dont nous reparlerons) puis d'une fonction d'activation LeakyReLU pour la non-linéarité. Le nombre de couches est relativement restreint, car nous ne voulons pas que le discriminateur soit trop performant par rapport au générateur. Nous n'avons pas besoin ici de pooling, puisque le modèle apprend lui-même comment réduire la dimension spatiale lors des convolutions (grâce au pas de 2). Nous n'utilisons pas non plus de dropout, puisque cela empire la performance du modèle et n'est pas nécessaire ici. Par ailleurs, la sortie de notre discriminateur n'est pas standard. En effet, nous avons besoin ici de préserver les détails fins de l'image, et la colorisation devrait être ajustée en fonction du morceau de l'image étudié plutôt que pour l'image dans sa globalité. Ainsi, un discriminateur jugeant des morceaux de l'image plutôt que l'image entière est préféré. Une telle implémentation est appelée PatchGAN, et est presque toujours utilisée lors des tâches de traduction image à image ou les détails sont plus importants que la structure globale.

Nos premiers essais avec les GAN n'ont cependant pas été concluants du tout. Nous n'avons en effet pas réussi à ce que le générateur apprenne, car il se faisait systématiquement dominer par le discriminateur. Une telle domination arrive lorsque presque toutes les images générées sont détectées comme fausses par le discriminateur. Dès lors, il ne reste que très peu d'images sur lesquelles le générateur peut s'entrainer, et il tend à se sur ajuster aux images qui ont trompé le discriminateur. Le générateur n'est alors pas capable d'apprendre efficacement, et sa performance tend à empirer au fil des epochs. Ce problème est extrêmement commun avec les GAN, y compris ceux qui ne servent pas à la colorisation. Leur entrainement est réputé très difficile, en plus d'être particulièrement lourd computationnellement. Ainsi, une part importante de la littérature qui leur est consacrée se concentre sur le fait de stabiliser l'entraînement. Les méthodes retenues dans notre cas ont été présentées dans la revue de littérature, mais beaucoup d'autres ont également été explorées avec plus ou moins de succès.

Une des méthodes les plus répandues pour stabiliser l'entraînement du GAN est de changer la fonction de perte des deux modèles. Ainsi, plutôt que de demander un choix binaire au discriminateur, nous pouvons le faire évaluer chaque image à l'aide d'un score. Ce score est élevé si le discriminateur pense l'image réelle, faible s'il pense l'image générée. Le discriminateur peut alors tenter de maximiser la distance entre le score moyen attribué aux images générées et aux vraies images. De son côté, le générateur tente de maximiser le score accordé à ses images par le discriminateur. Cette fonction de perte est appelée la perte Wasserstein, et transforme le GAN en un WGAN. Cependant, dans son implémentation initiale, cette fonction de perte doit nécessairement s'accompagner d'un écrêtage des gradients du discriminateur. Cet écrêtage est problématique, car il peut fortement limiter la capacité d'apprendre du discriminateur. Ainsi, une variante des WGAN consiste à ajouter une forme de régularisation aux gradients du discriminateur plutôt que de les écrêter. Cette pénalité pour les gradients permet au discriminateur d'avoir plus de liberté, tout en empêchant les gradients d'exploser. Le nouveau modèle est ainsi appelé WGAN-GP, pour « Gradient Penalty ».

L'utilisation d'un WGAN-GP ainsi que d'un PatchGAN pour le discriminateur nous a ainsi permis de commencer à stabiliser l'entraînement de nos modèles. Bien que de nombreux essais aient dû être effectués afin de trouver des hyperparamètres optimaux, nous étions d'ores et déjà capables d'entraîner un modèle pendant plusieurs epochs, sans que la performance du générateur empire. Cependant, nous avons également dû ajouter la fonction de perte combinée du U-Net à la perte Wasserstein du générateur, afin de nous assurer que le générateur n'apprenne pas à tromper le discriminateur (désormais plus faible) trop facilement. En effet, le générateur peut sinon profiter de certaines faiblesses du discriminateur pour produire des images avec une unique couleur qui trompe toujours le discriminateur. Forcer le générateur à maintenir une certaine mesure de similarité et à minimiser l'erreur absolue (le poids de ces métriques dans la perte est un hyperparamètre) permet d'éviter ces excès, qui rendent sinon le générateur inutile. Cette solution ne provient pas de la littérature, mais a très bien fonctionné dans notre cas.

Un dernier ajout, inspiré d'une autre publication de blog (Pasini, 2019), concerne le type de normalisation à appliquer au sein du discriminateur. En effet, la littérature semble indiquer qu'utiliser une normalisation spectrale (Miyato et al., 2018) plutôt que par batch tend à grandement améliorer la stabilité et la performance des GAN. La normalisation spectrale consiste à normaliser les poids d'une couche par leur plus grande valeur propre, limitant ainsi l'agressivité des mises à jour pendant l'apprentissage. L'ajout de cette normalisation nous a permis de stabiliser encore plus l'entrainement, et d'améliorer grandement nos résultats. Nous n'avons cependant pas pu implémenter cette technique dans le générateur, puisque nous

utilisons un modèle pré-entrainé dans l'encodeur. Finalement, notre modèle final est donc un mélange de divers modèles et techniques issus de la littérature ou de notre imagination. Toutes ces transformations visaient avant tout à stabiliser l'entrainement, l'aspect le plus difficile des réseaux antagonistes génératifs.

4. Section empirique

4.1 Entraînement

En ce qui concerne l'entraînement en tant que tel, le même jeu de données a été utilisé pour les deux modèles. Ce jeu de données (Kaggle, 2019) comprend plus de 20 000 images de paysages avec une résolution 128x128. Ce jeu de données n'est pas très conséquent ou dans une résolution élevée, mais correspond à ce que nous pouvions faire de mieux compte tenu de la puissance de nos ordinateurs. Afin de ne pas nuire à l'entraînement, nous avons filtré les images en noir et blanc présentes initialement dans le jeu de données. Nous avons ensuite normalisé puis transformé ces images en noir et blanc, afin d'obtenir deux jeux distincts, l'un en noir et blanc que nous donnons en entrée aux modèles, et l'un dans l'espace de couleur RGB que nous prenons comme cible. Nous faisons de même pour nos images tests, des paysages québécois trouvés sur Google et redimensionnés pour correspondre à l'entrée attendue par le modèle.

Énormément d'hyperparamètres rentrent en compte dans l'entrainement de nos modèles, et il ne semble pas particulièrement pertinent de les énumérer un par un (ils sont consultables au sein du code fourni en annexe). Cependant, nous pouvons noter que nous avons un taux d'apprentissage particulièrement faible pour notre générateur dans le GAN, bien moins élevé que celui du discriminateur. En effet, puisque nous utilisons le U-Net précédemment entrainé, il est nécessaire de ne pas modifier trop rapidement des poids qui sont déjà plutôt bons. Par ailleurs, alors que nous avions gelé les couches MobileNet lors de l'entrainement de notre U-Net pour la même raison, ce taux d'apprentissage très faible nous permet de les dégeler sans que le modèle s'effondre comme auparavant. Ainsi, notre GAN tente d'affiner la performance globale du U-Net initial qu'il prend comme base.

L'entraînement du U-Net aura nécessité quelques heures sur un GPU, tandis que l'entraînement du GAN aura lui pris près d'une centaine d'heures. Ce long temps d'entraînement est notamment dû au très grand nombre de variations que nous avons dû explorer avant d'obtenir un résultat satisfaisant. Un critère d'arrêt a d'ailleurs été défini pour arrêter l'entraînement si la performance sur l'échantillon de validation cesse de s'améliorer, pour éviter le surajustement et d'allonger inutilement le temps d'entraînement.

4.2 Résultats

La présente section expose les résultats obtenus par les deux modèles que nous avons examinés dans le cadre de notre étude. Dans notre évaluation, une approche plus subjective a été privilégiée, mettant ainsi en lumière la qualité visuelle des résultats plutôt que de se concentrer exclusivement sur des métriques objectives. En effet, l'utilisation de métriques n'est pas jugée très pertinente dans le cas de la colorisation, notamment car chacune de ces métriques comporte de nombreux défauts. Puisque l'appréciation du résultat est hautement subjective, il est normal de ne pas avoir de véritables métriques pour quantifier de manière précise la performance d'un modèle.

Parce que le U-Net a été entrainé avec le SSIM et l'erreur absolue moyenne en tant que fonction de perte, il performe très bien sur ces mêmes métriques au moment de l'évaluation sur l'échantillon de validation. Nous obtenons ainsi respectivement 0.93 (SSIM) et 0.04 (MAE).

Pour ce qui est du GAN, la fonction de perte Wasserstein est utilisée en plus du SSIM et de la MAE. Nous pouvons voir sur les graphiques ci-dessous que le générateur se stabilise rapidement. En effet, le générateur va rapidement réduire la fonction de perte adversariale (observable avec **Δ**perte totale – **Δ**(SSIM+MAE)), mais atteint un plateau en quelques epochs. Nous pensons que cela est principalement lié à la petite taille de notre échantillon, qui a rapidement épuisé son pouvoir prédictif. Notre modèle manque aussi peut-être de capacité, mais celle-ci est limitée par la puissance de nos ordinateurs. Les petits gains graduels obtenus par la suite pour la perte du générateur sont donc plutôt liés à la maximisation de la similitude et la minimisation de l'erreur absolue moyenne. Cependant, optimiser ce morceau de la fonction de perte tend à empirer la performance, puisque le modèle reproduit les mêmes artefacts que le U-Net initial, entraîné de cette manière. Nous avons inclus cette perte combinée pour éviter que le générateur apprenne à toujours tromper le discriminateur avec une couleur unique, mais nous ne voulons pas hériter de ses défauts. Concernant la perte Wasserstein du discriminateur, nous pouvons voir qu'elle est très rapidement limitée par les pénalités de gradients implantées pour stabiliser l'entraînement. Cela permet qu'il ne domine pas le générateur, même lorsque celui-ci a épuisé le pouvoir prédictif des données.

Ainsi, après comparaison des résultats du GAN à différentes epochs, nous avons trouvé que le GAN performe le mieux après seulement quatre epochs. Ce faible nombre n'est pas si surprenant, puisque notre générateur était pré-entrainé. Après ce point, le GAN tend à se rapprocher de plus en plus de la performance du U-Net initial et perd les gains obtenus grâce à l'affrontement avec le discriminateur. La performance reste cependant supérieure au U-Net. Ces résultats démontrent par ailleurs la non-

pertinence des métriques pour évaluer la performance des modèles, puisque le GAN obtient des scores moins bons que le U-Net initial malgré le fait que les résultats soient visuellement plus convaincants.

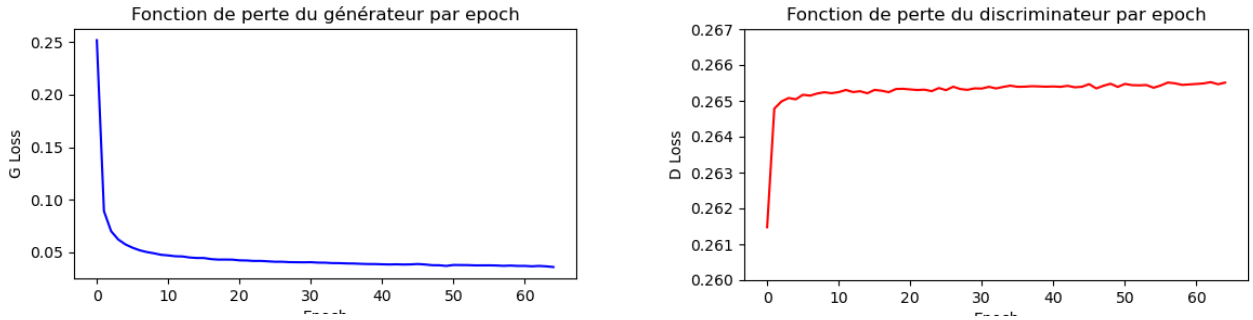

Figure 3 : Fonction de perte du générateur et du discriminateur du GAN

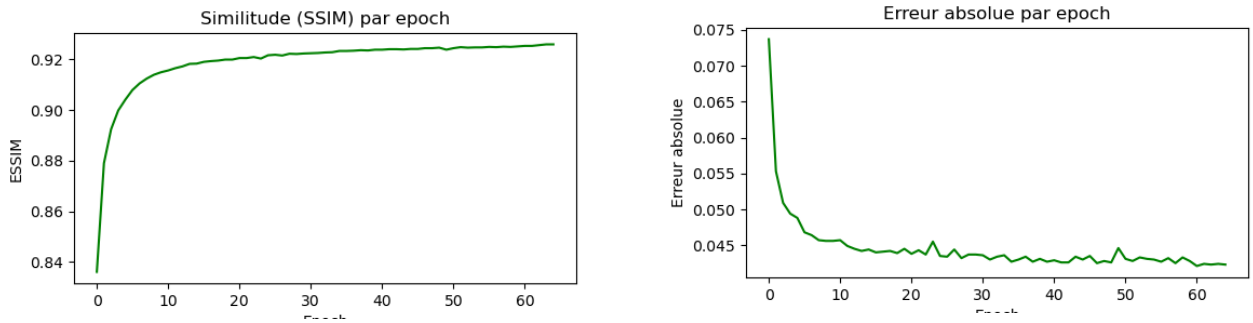

Figure 4 : Indice de Similarité Structurale (SSIM) et erreur absolue moyenne (MAE) du GAN

Par ailleurs, lorsque nous regardons la littérature, nos résultats semblent coïncider avec ceux des travaux connexes. En effet, les conclusions d'une étude sur la colorisation d'images comparant les auto-encodeurs et le GAN sur différents espaces de couleurs démontrent que le GAN performe le mieux (Arshiya Sarmai et Apratim Sadhu, 2022). Cette même étude montre également l'avantage dont disposent les U-Net par rapport aux autoencodeurs traditionnels, puisque les images des autoencodeurs sont floues. De même, une autre étude de colorisation comparant cette fois-ci le GAN au U-net permet également de démontrer une meilleure performance du GAN par rapport au U-net (Kamyar Nazeri *et al.*, 2018). Cette convergence entre les constatations antérieures dans le domaine et nos résultats renforce la validité de notre approche.

Concernant les résultats, l'évaluation de nos modèles se basera sur une inspection visuelle. Pour chaque figure, on retrouve **à gauche la vraie image, au milieu l'image du U-net initial et à droite le résultat du générateur du GAN**. Il est important de noter que la comparaison est cependant injuste, puisque le GAN utilise le U-Net pré-entrainé comme base et tente de l'améliorer. Les résultats du GAN démontrent ainsi plutôt l'ampleur de cette amélioration.

Figure 5

Sur cette image, nous pouvons voir que le GAN est bien meilleur que le U-net initial, notamment au niveau des cailloux. Là où le U-net n'arrive absolument pas à les distinguer mais considère plutôt cela comme de l'herbe, le GAN arrive à les différencier bien qu'il reste tout de même un peu de vert au pied du sapin. Quant au ciel, comme on le verra dans les autres images, celui du GAN a généralement une couleur un peu plus bleue et plus cohérente.

Figure 6

Pour l'image suivante, les deux modèles sont mauvais. Le GAN ne semble pas réussir à faire globalement mieux que le U-net, notamment au niveau de la couleur de l'herbe. Néanmoins, lorsque nous regardons le ciel ou la colorisation du lac par le GAN, celles-ci sont quand même plus réalistes que celles du U-net qui tend à apposer des couleurs trop rougeâtres. Ce résultat n'est cependant pas très satisfaisant dans les deux cas.

Figure 7

Pour la figure 7, la différence entre les deux modèles est importante. En effet, nous retrouvons un patch orange dans le U-net que le GAN l'enlève. De même, l'artéfact bleu au niveau du bras s'atténue dans le GAN comparativement au U-net. Par ailleurs, nous pouvons voir que les deux modèles associent la roche à de l'herbe. La couleur du GAN est toutefois un peu plus correcte que celle du U-Net, puisque beaucoup moins verte et plus proche de la couleur d'une roche.

Figure 8

Sur cette image, nous pouvons voir que le GAN génère une meilleure colorisation que le U-net. En effet, nous observons de gros patchs bleus au niveau des arbres dans le U-net, que nous retrouvons moins dans celle du GAN. Les arbres, bien qu'ils n'aient plus de feuilles, sont coloriés dans un vert vif qui ne correspond pas à l'image initiale. Cela reste compréhensible, car au vu de la faible quantité de pixels, le modèle ne sait pas exactement faire la différence entre un arbre avec ses feuilles et un arbre qui a seulement ses branches.

Figure 9

Pour cette image, les résultats du GAN sont également meilleurs que ceux du U-net. Nous pouvons voir que le U-net, notamment en bas de l'image, a tendance à tirer vers de l'orange-jaune tandis que le GAN détecte correctement les cailloux. La verdure des arbres semble également plus réaliste dans la version du GAN, tout comme le ciel.

Figure 10

Enfin, pour cette image, les deux modèles ne sont vraiment pas bons mais le GAN est légèrement meilleur. Dans le U-Net, la couleur du ciel est vraiment étrange et cela est accentué par les patchs marron au niveau des branches. Toutefois, ce qui n'a pas de sens, c'est surtout le sable, colorisé en vert. Pour le GAN, bien que les résultats ne soient pas parfaits car trop pâles, cette incohérence est réparée et le sable est correctement séparé de l'herbe à côté. La montagne a une couleur trop pâle mais pas irréaliste dès lors que nous ne comparons pas cela à l'image originale. L'image du GAN n'est pas choquante, et pourrait correspondre à une photographie prise en hiver par exemple.

Ainsi, le point de comparaison le plus juste serait de regarder l'image et de se poser la question si la couleur générée a un sens plutôt que de comparer nécessairement à l'image de départ. Ce qu'il est important de souligner, c'est que ce sont des images que le modèle n'a jamais vues. De ce fait, il ne pourra jamais deviner parfaitement la couleur parmi les milliers de possibilités. Cependant, s'il parvient à générer une image où les couleurs sont cohérentes et réalistes, alors cela représente déjà une réussite. C'est d'ailleurs cette raison qui permet de justifier l'affirmation selon laquelle le modèle GAN est meilleur que le simple U-net. Bien qu'il soit imparfait, le GAN donne dans la plupart des cas des résultats satisfaisants et cohérents.

Conclusion

En conclusion de cette étude, il ressort que le GAN a largement démontré son efficacité dans l'amélioration de la colorisation des images, surpassant dans la plupart des cas les résultats du U-net. Ces résultats soulignent la puissance du modèle GAN dans le domaine de la colorisation, offrant des perspectives prometteuses pour des applications futures dans la restauration de vieilles photographies.

Toutefois, l'entraînement des GAN demeure un défi considérable, comme en témoigne la vaste littérature cherchant à stabiliser ce processus complexe. La recherche d'une combinaison efficace s'est avérée chronophage, en raison du grand nombre d'hyperparamètres à ajuster, notamment lorsque les essais sont restreints par la puissance de notre ordinateur.

Notre étude présente ainsi certaines limitations importantes. Nos modèles ont été entraînés uniquement sur un ensemble de 20 000 images de paysages au format 128x128, en raison des contraintes imposées par la capacité de mémoire et la puissance de calcul de nos ordinateurs. La faible résolution des images a rendu délicate la détection de certains petits patterns, permettant habituellement une colorisation fine. De plus, il est important de noter que le modèle est entraîné seulement à coloriser des paysages, limitant ainsi sa généralisation à d'autres contextes.

Concernant les pistes d’amélioration, il serait judicieux d'utiliser des images à résolution plus élevée pour une colorisation plus précise des détails de l’image. De même, un échantillon beaucoup plus conséquent et varié serait préférable, sous réserve que les ressources computationnelles le permettent. Enfin, l'adoption d'un espace colorimétrique mieux adapté à la tâche, tel que LAB, serait pertinente. Pour ce faire, le portage du modèle de TensorFlow à PyTorch serait nécessaire, puisque cet espace de couleur n’est pas supporté par TensorFlow.

Bibliographie

Arjovsky, M., Chintala, S., & Bottou, L. (2017). *Wasserstein GAN.* https://doi.org/10.48550/arXiv.1701.07875.

Arshiya Sarmai, A. S. (2022). Image Colorization using Convolutional Autoencoders and Generative Adversarial Networks. *International Advanced Research Journal in Science, Engineering and Technology, 9*(1).

Bansal, P. (2019). *Intel Image Classification*. Récupéré sur Kaggle: https://www.kaggle.com/datasets/puneet6060/intel-image-classification

Benet, L. T. (2021). *GAN-based Image Colourisation with Feature Reconstruction Loss.* Escola Tècnica d'Enginyeria de Telecomunicacio de Barcelona Universitat Politècnica de Catalunya.

Bhagyashree, Kushwaha, V., & Nandi, G. C. (2020). Study of Prevention of Mode Collapse in Generative Adversarial Network (GAN). *IEEE 4th Conference on Information & Communication Technology (CICT)* (pp. p.1-6.). Chennai, India: doi: 10.1109/CICT51604.2020.9312049.

Ciresan, D., Gambardella, L., & Giusti, A. S. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. *In: NIPS*, p.2852–2860.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative Adversarial Nets. https://doi.org/10.48550/arXiv.1406.2661.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). *Improved Training of Wasserstein GANs.* https://doi.org/10.48550/arXiv.1704.00028.

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2016). *Image-to-Image Translation with Conditional Adversarial Networks.* https://doi.org/10.48550/arXiv.1611.07004.

Massart, E. (2022). Improving weight clipping in Wasserstein GANs. *2022 26th International Conference on Pattern Recognition (ICPR),*, (pp. p.2286-2292, doi: 10.1109/ICPR56361.2022.9956056.). Montreal, QC, Canada.

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral Normalization for Generative Adversarial Networks. https://doi.org/10.48550/arXiv.1802.05957.

Nazeri, K., Ng, E., & Ebrahimi, M. (2018). Image colorization using generative adversarial networks. *Lecture Notes in Computer Science*, p.85-94. https://doi.org/10.1007/978-3-319-94544-6_9.

Pasini, M. (2019). *10 Lessons I Learned Training GANs for one Year*. Récupéré sur Towards Data Science: https://towardsdatascience.com/10-lessons-i-learned-training-generative-adversarial-networks-gans-for-a-year-c9071159628

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *https://doi.org/10.48550/arXiv.1505.04597*.

Sarmai, A., & Sadhu, A. (2022). Image Colorization using Convolutional Autoencoders and Generative Adversarial Networks. *International Advanced Research Journal in Science, Engineering and Technology, 9(1).*

Shariatnia, M. (2020). *Colorizing black & white images with U-Net and conditional GAN — A Tutorial*. Récupéré sur Towards Data Science: https://towardsdatascience.com/colorizing-black-white-images-with-u-net-and-conditional-gan-a-tutorial-81b2df111cd8

Siddique, N., Sidike, P., Elkin, C. P., & Devabhaktuni, V. (2021). U-net and its variants for medical image segmentation: A review of theory and applications . *IEEE Access, 9*, p.82031-82057. http://doi.org/10.1109/.