université de BORDEAUX

Colorisation interactive d'images

Mémoire de projet de fin d'études
Master 2 Informatique pour l'Image et le Son

Angelika COÏC,
Joséphine FOIX,
Chloé NEUVILLE

Clients :
Hernan Carrillo,
Michaël Clément,
Aurélie Bugeau

Chargé de TD :
Pascal Desbarats

Mars 2023

Table des matières

1 Contexte **3**
1.1 Introduction 3
1.2 Analyse de l'existant 4
1.2.1 Algorithmes de segmentation en superpixel 4
1.2.2 Architecture UNet 7
1.2.3 Colorisation d'images par apprentissage profond 8
1.2.4 Couches d'attention 9
1.3 État de l'art de la colorisation d'image 9

2 Analyse des besoins **11**
2.1 Version avec amélioration 11
2.1.1 Explications 11
2.1.2 User stories 11
2.1.3 Les Besoins 12
2.2 Version avec ajout de fonctionnalités 12
2.2.1 Explications 12
2.2.2 User stories 12
2.2.3 Les Besoins Fonctionnels 14
2.2.4 Les Besoins Non-Fonctionnels 14

3 Recherche et méthodologie **16**
3.1 Méthodologie de notre reconduction de couleurs 16
3.1.1 Déploiement général des opérations 17
3.1.2 Masques et superpixels 18
3.1.3 Carte d'attention 20
3.2 Déroulement et résultats des recherches 21

4 Architecture et implémentation **27**
4.1 Architecture 27
4.1.1 Version avec amélioration 27
4.1.2 Version avec ajout de fonctionnalités 28
4.2 Implémentation 28
4.2.1 Mise au propre 28
4.2.2 Affichage Interactif 29

5 Tests **29**
5.1 Tests de non régression 29
5.2 Test du Singe 30

6 Suivi de projet **30**
6.1 Méthodologie d'organisation 30
6.2 Liste des Tâches, Priorités & Complexités 31

7 Conclusion **33**

8 Bibliographie **35**

9 Annexe **37**
9.1 Manuel d'utilisation 37
9.1.1 Présentation de l'application 37
9.1.2 Exigences système 37
9.1.3 Lancement de l'application 38
9.1.4 Interface utilisateur et Colorisation 38
9.1.5 Importation d'images et exportation d'images 40
9.1.6 Problèmes 41
9.2 Images 42
9.3 Tâches 43
9.4 Diagramme 46

1 Contexte

1.1 Introduction

Le projet initial s'inscrit dans le cadre du projet ANR PostProdLEAP[1] qui s'intéresse à la restauration et la colorisation d'images et de vidéos d'archive. Dans le cadre de sa thèse, Hernan Carrillo a développé une méthode par apprentissage profond permettant de guider la colorisation d'une image en niveau de gris avec une image couleur fournie par un utilisateur [2]. Nous sommes maintenant intéressés par l'extension de ce travail en permettant à l'utilisateur d'autres types d'interaction.

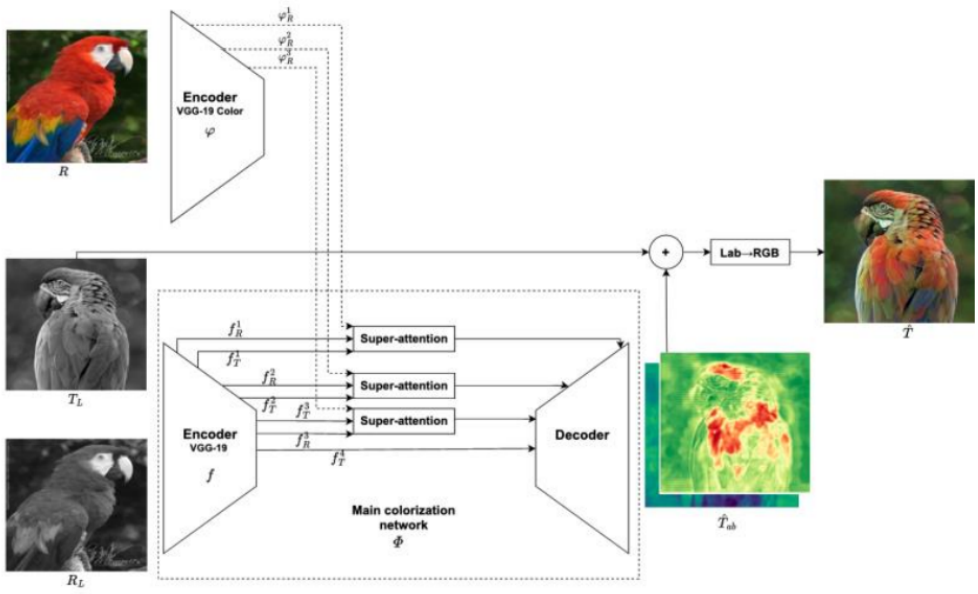


FIGURE 1 – Diagramme de la méthode de colorisation d'image.

L'objectif de ce Projet de Fin d'Etudes est d'ajouter en entrée du réseau une carte de segmentation permettant de guider plus précisément la colorisation. Par exemple si l'image exemple de la Figure 1 contenait un perroquet et un autre animal, seules les couleurs du perroquet devraient être utilisées pour guider la colorisation. Il s'agit donc de permettre aux couches d'atten-

tion d'intégrer la colorisation. Dans un premier temps, il s'agira de prendre en main, de comprendre et si nécessaire d'améliorer le code de la méthode de colorisation d'image par apprentissage profond. Par la suite, l'ajout de carte de segmentation sera étudié, ainsi que l'ajout de l'interactivité avec l'utilisateur.

La méthode de la Figure 1 est composée de deux parties principales :

— 1) l'extracteur de caractéristiques de couleur φ qui extrait des cartes de caractéristiques multi-niveaux à partir d'une image de référence en couleur,

— 2) le réseau principal de colorisation Φ, qui apprend à faire correspondre une image de luminance T_L à ses canaux de chrominance T_{ab} étant donné l'image de couleur de référence R.

Ce guidage de colorisation est effectué par des modules de super-attention, qui apprennent des cartes d'attention basées sur des superpixels à partir de la cible et des cartes de caractéristiques de référence à partir de niveaux distincts.

1.2 Analyse de l'existant

1.2.1 Algorithmes de segmentation en superpixel

SLIC (Simple Linear Iterative Clustering) [4] est un algorithme de segmentation d'image populaire utilisé pour diviser une image en segments ou en régions homogènes.

Les deux images de Figure 2 issues de l'article [11], représentes 3 différentes segmentations à l'aide de l'algorithme SLIC, avec des superpixels de tailles (approximatives) de 64, 256 et 1024 pixels. Les superpixels sont compacts, de taille uniforme et adhèrent à la frontière des régions.

SLIC peut être utilisé pour la détection d'objets et la reconnaissance de formes. En segmentant l'image en régions homogènes, il est possible de détecter des objets dans l'image en se concentrant sur les régions qui ont des caractéristiques similaires. Ensuite, en utilisant des techniques de reconnaissance de formes, il est possible d'identifier l'objet dans l'image.

L'algorithme SLIC, visible en pseudo code, Algorithme 1, utilise une méthode de clustering k-means pour regrouper les pixels en segments. Il commence par sélectionner un certain nombre de points de départ, appelés centres de cluster, qui sont distribués uniformément dans l'image. Ensuite, pour chaque pixel de l'image, l'algorithme calcule la distance entre le pixel

et tous les centres de cluster et associe ce pixel au centre de cluster le plus proche. Cela crée une partition initiale de l'image en segments.

FIGURE 2 – Illustration de l'algorithme SLIC

Ensuite, l'algorithme SLIC effectue une itération pour affiner cette partition en minimisant la distance entre chaque pixel et le centre de cluster auquel il est associé. Les centres de cluster sont déplacés vers la moyenne des pixels qui leur sont associés, et cette étape est répétée jusqu'à ce que la convergence soit atteinte.

L'algorithme SLIC est particulièrement adapté à la segmentation d'images en temps réel, grâce à sa rapidité de traitement. En effet, il peut être implémenté en parallèle, ce qui permet de segmenter des images à haute résolution en temps réel. De plus, SLIC est très efficace en termes de consommation de mémoire, ce qui permet de l'utiliser sur des plates-formes embarquées ou des appareils mobiles.

Algorithm 1 SLIC superpixel segmentation

/ Initialization */*
Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3×3 neighborhood.

Set label $l(i) = -1$ for each pixel i
Set distance $d(i) = \infty$ for each pixel i
repeat
  / Assignment */*
  for each cluster center C_k **do**
    for each pixel i in a $2S \times 2S$ region around C_k **do**
      Compute the distance D between C_k and i.
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  / Update */*
  Compute new cluster centers.
  Compute residual error E.
until $E \leq$ threshold

1.2.2 Architecture UNet

U-NET est l'un des réseaux de neurones les plus couramment utilisés pour la segmentation d'images. Ce modèle entièrement convolutif a été initialement développé en 2015 par Olaf Ronneberger, Phillip Fischer et Thomas Brox pour la segmentation d'images médicales.

L'architecture U-NET est composée de deux "chemins". Le premier, appelé encodeur ou chemin de contraction, est un assemblage de couches de convolution et de couches de "max pooling" qui permettent de capturer le contexte de l'image en créant une carte de caractéristiques et en réduisant sa taille pour diminuer le nombre de paramètres du réseau.

Le second chemin, appelé décodeur ou chemin d'expansion symétrique, permet une localisation précise grâce à la convolution transposée. Il permet de restaurer la résolution de l'image et de fusionner les informations de localisation avec les caractéristiques extraites dans l'encodeur.

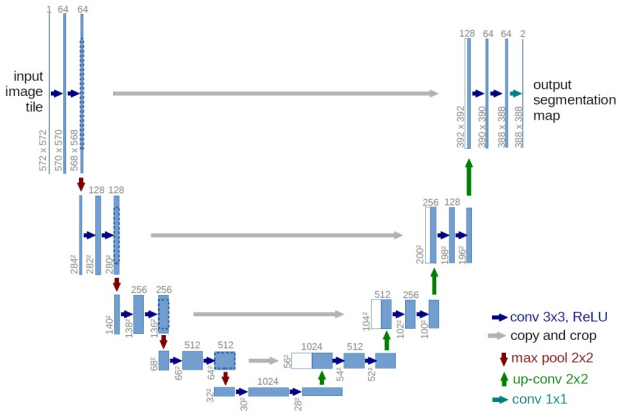


FIGURE 3 – Architecture U-net

En Deep Learning, de vastes ensembles de données sont nécessaires pour entraîner les modèles. Cependant, rassembler de telles quantités de données pour résoudre des problèmes de classification d'images peut s'avérer difficile en termes de temps, de budget et de ressources matérielles. De plus, l'étiquetage des données nécessite l'expertise de plusieurs développeurs et ingénieurs, ce qui est particulièrement vrai pour des domaines hautement spécialisés tels que les diagnostics médicaux.

U-NET permet de résoudre ces problèmes en étant efficace même avec un ensemble de données limité et en offrant une précision supérieure aux

modèles conventionnels.

Une architecture autoencodeur classique réduit la taille des données d'entrée à travers les couches suivantes. Ensuite, le décodage commence et la représentation de caractéristiques linéaires est apprise, augmentant progressivement la taille jusqu'à ce qu'elle atteigne celle de l'entrée. Bien que cette architecture soit idéale pour préserver la taille initiale, elle compresse l'input de manière linéaire, entraînant ainsi une perte de certaines caractéristiques.

C'est ici que l'architecture en U de U-NET [16] (cf. Figure 3) se démarque. La déconvolution est effectuée dans le décodeur, évitant ainsi le problème de goulot d'étranglement rencontré avec une architecture auto-encodeur et préservant toutes les caractéristiques.

1.2.3 Colorisation d'images par apprentissage profond

La colorisation par apprentissage profond est une technique qui utilise des architectures de réseaux de neurones pour coloriser automatiquement des images en noir et blanc. Cette technique est de plus en plus populaire dans le domaine de la vision par ordinateur et de l'imagerie numérique, car elle permet de donner vie à des images anciennes ou endommagées en leur ajoutant des couleurs.

Le processus de colorisation en apprentissage profond implique généralement deux étapes principales :

- Entraînement du modèle : dans cette première étape, un modèle de réseau de neurones est entraîné sur une base de données d'images en noir et blanc et en couleur. Le but est de faire apprendre au modèle à prédire les couleurs des pixels en se basant sur les couleurs des pixels environnants et d'autres caractéristiques de l'image, telles que les contours et les textures.
- Colorisation de l'image : une fois que le modèle est entraîné, il peut être utilisé pour coloriser des images en noir et blanc en prédisant les couleurs des pixels manquants. Pour cela, le modèle prend en entrée l'image en noir et blanc et produit une image en couleur correspondante.

Il existe différentes approches pour la colorisation en apprentissage profond, mais les plus populaires utilisent des réseaux de neurones convolutionnels (CNN). Les CNN sont des réseaux de neurones spécialement conçus pour la reconnaissance d'images et peuvent apprendre des caractéristiques à différentes échelles dans l'image, ce qui les rend particulièrement adaptés à la

colorisation.

L'article [5] est l'un des travaux fondateurs sur la colorisation en apprentissage profond. Les auteurs ont proposé une méthode pour la colorisation d'images en noir et blanc en utilisant un réseau de neurones convolutionnel (CNN) profond. Le réseau de neurones a été entraîné à prédire une distribution de probabilité sur les couleurs de chaque pixel de l'image en noir et blanc. La méthode de colorisation proposée a été appliquée à plusieurs bases de données d'images en noir et blanc. Plusieurs travaux existants seront mieux expliqués dans la partie 1.3.

1.2.4 Couches d'attention

Les couches d'attention sont une famille de couches de réseaux de neurones qui ont été introduites pour améliorer la performance des réseaux de neurones dans des tâches de traitement de séquences.

L'idée derrière les couches d'attention est de permettre au modèle de se concentrer sur les parties les plus importantes de la séquence d'entrée, plutôt que de traiter la séquence dans son ensemble de manière uniforme. Cela permet au modèle de mieux capturer les dépendances à long terme dans la séquence et d'améliorer la qualité des prédictions.

Plus précisément, les couches d'attention calculent des pondérations pour chaque élément de la séquence d'entrée, en fonction de l'importance de cet élément pour la tâche en cours. Ces pondérations sont ensuite utilisées pour pondérer l'importance de chaque élément lors de la propagation avant dans le réseau de neurones.

1.3 État de l'art de la colorisation d'image

Il existe de nombreuses techniques pour coloriser une image.

La méthode manuelle, la plus basique, avec des outils de dessin, peut-être effectuée. C'est une méthode difficile, chronophage et qui nécessite des compétences artistiques. Celle-ci sera très précise car le contrôle sera total sur la colorisation, bien que cette dernière sera subjective à l'artiste.

Une autre méthode [6], directement inspirée du coloriage, laisse un peu moins de contrôle à l'utilisateur, mais lui permet de gagner du temps et ne nécessite pas de compétence artistique. Elle implique que l'utilisateur dessine une ligne de couleur pour chaque zone de l'image. Ensuite, les couleurs sont

propagées jusqu'aux frontières des objets en utilisant les contours de l'image de luminance pour définir la force et l'orientation de diffusion des couleurs.

Donner la possibilité à l'utilisateur de fournir une image similaire en couleur en tant que modèle à l'algorithme est une méthode [7]. Le principe est de transférer les couleurs de cette image modèle dans l'image à coloriser. Tout d'abord, l'image modèle est convertie en niveaux de gris. Pour chaque petit carré de l'image à coloriser, appelé patch, on recherche le patch le plus similaire en termes de luminance et/ou de texture dans l'image modèle convertie en niveaux de gris. Les couleurs de ce patch sont ensuite copiées dans l'image à coloriser.

La méthode de colorisation [8] proposée utilise une architecture de réseaux de neurones convolutifs profonds avec des couches de convolutions, de max-pooling et de normalisation de lot. Il apprend ainsi à prédire la couleur des pixels en se basant sur les informations contextuelles de l'image en niveaux de gris. Pour cela, le réseau est entraîné sur un ensemble de données d'apprentissages comprenant des images en niveaux de gris et leurs correspondances en couleur.

Dans l'article [9], la méthode de colorisation proposée utilise un réseau de neurones convolutifs profonds avec une architecture "Classification via Retrieval" (CvR). Cette architecture permet d'apprendre une représentation sémantique de l'image ainsi qu'une correspondance entre les pixels en niveaux de gris et leur couleur correspondante. L'entraînement de ce réseau utilise le même type de donnée que le précédent. Le réseau est entraîné à minimiser une fonction de coût qui combine une perte de classification et une perte de récupération. La perte de classification mesure l'erreur de prédiction de la couleur de chaque pixel, tandis que la perte de récupération mesure la similarité entre les caractéristiques extraites de l'image en niveaux de gris et celles extraites de l'image en couleur correspondante.

La méthode de colorisation proposée [10], utilise un modèle de réseau de neurones appelé "transformeur", qui est entraîné sur des données contenant des images en niveaux de gris et leurs correspondances en couleur. Ce modèle est capable d'apprendre une représentation sémantique de l'image en niveaux de gris ainsi qu'une correspondance entre les pixels en niveaux de gris et leur couleur correspondante. Le modèle prend en entrée une image en niveaux de gris et utilise une série de couches de transformeur pour prédire une carte de probabilité pour la couleur de chaque pixel. Les mécanismes d'attention sont utilisés pour se concentrer sur les parties les plus importantes de l'image en niveaux de gris pour prédire la couleur des pixels correspondants.

2 Analyse des besoins

Durant la réalisation du projet, les user stories ont évolué en fonction des besoins de nos clients. Nous allons présenter les besoins finaux en fonction des deux versions demandées par les clients.

2.1 Version avec amélioration

2.1.1 Explications

Cette version du code que nos clients attendent correspond au code qu'ils nous ont fourni, **mis au propre**, avec **une sauvegarde des 4 matrices d'attentions** de chaque image colorisée par l'algorithme. Le projet étant un travail de recherche, nos clients avaient besoin d'un code publiable tout en restant en accord avec leur article. Nos modifications sont donc minimales dans cette version.

2.1.2 User stories

1. L'utilisateur veut entraîner le modèle de colorisation. Il met le mode `train`, lance l'entraînement.
2. L'utilisateur veut coloriser une image en niveau de gris, **sans utiliser de référence couleur, c'est à dire automatiquement.** Il met le mode `test`, met l'image à coloriser dans le dossier des cibles et met une image noire dans le dossier des références. L'utilisateur lance la colorisation et obtient son image colorisée automatiquement dans le dossier des résultats.
3. L'utilisateur veut coloriser une image en niveau de gris, **en utilisant une référence colorée.** Il met le mode `test`, met l'image à coloriser dans le dossier des cibles et met une image en couleur dans le dossier des références. L'utilisateur lance la colorisation et obtient son image colorisée à partir de l'image de référence dans le dossier des résultats.
4. L'utilisateur veut coloriser une image en niveau de gris, **en utilisant une référence colorée et enregistrer les cartes d'attention.** Il met le mode `test`, met l'image à coloriser dans le dossier des cibles et met une image en couleur dans le dossier des références. L'utilisateur lance la colorisation avec l'option `-save_attention_maps` pour obtenir les cartes d'attention et obtient son image colorisée à partir

de l'image de référence et les cartes d'attention de la colorisation dans le dossier des résultats .

2.1.3 Les Besoins

Les besoins principaux sont donc de mettre le code au propre. Il est donc question de lisibilité du code, notamment par les nomenclatures de variables, de classes ainsi que des dossiers si nécessaire. La mise en norme PEP8 est mentionnée. L'article mentionne également la démonstration de cartes d'attentions. Il s'agit d'ajouter au code existant une option ou un mode pour sauvegarder les cartes d'attention.

2.2 Version avec ajout de fonctionnalités

2.2.1 Explications

Nos clients souhaitaient une version orientée recherche. Premièrement, l'un des but est que les deux fonctionnalités principales de leur algorithme soient séparées en deux commandes distinctes. L'entraînement du modèle de colorisation doit pouvoir être lancé avec une commande différente de celle du test. Il est aussi question de mettre les paramètres importants dans les lignes de commande pour rendre l'algorithme plus adaptable à ce que veut l'utilisateur. Deuxièmement, ils souhaitaient pouvoir sélectionner des zones sur les images. Le but ici étant de pouvoir coloriser une zone de l'image cible et de pouvoir utiliser une zone de l'image de référence.

2.2.2 User stories

1. L'utilisateur veut coloriser une image en niveau de gris, **sans utiliser de référence couleur, c'est à dire automatiquement**. Il met l'image à coloriser dans le dossier des cibles et met une image quelconque dans le dossier des références. Ensuite, l'utilisateur lance le programme de colorisation. Une fenêtre s'ouvre avec l'image à coloriser, l'utilisateur clique sur le bouton "All", la fenêtre se ferme. Une nouvelle fenêtre s'ouvre, l'utilisateur clique sur le bouton "Next". L'utilisateur peut trouver l'image qui a été colorisé automatiquement dans le dossier des résultats.
2. L'utilisateur veut coloriser une image en niveau de gris, **en utilisant une référence colorée**. Il met l'image à coloriser dans le dossier des

cibles et met une image à utiliser comme référence dans le dossier des références. Ensuite, l'utilisateur lance le programme de colorisation. Une fenêtre s'ouvre avec l'image à coloriser, l'utilisateur clique sur le bouton "All", la fenêtre se ferme. Une nouvelle fenêtre s'ouvre avec l'image de référence, l'utilisateur clique sur le bouton "All". L'utilisateur peut trouver l'image qui a été colorisée à l'aide de la référence dans le dossier des résultats.

3. L'utilisateur veut coloriser une image en niveau de gris, **en utilisant les couleurs d'un objet sélectionné sur une référence.** Il met l'image à coloriser dans le dossier des cibles et met une image à utiliser comme référence dans le dossier des références. Ensuite, l'utilisateur lance le programme de colorisation. Une fenêtre s'ouvre avec l'image à coloriser puis il clique sur le bouton "All", la fenêtre se ferme. Une nouvelle fenêtre s'ouvre avec l'image de référence, l'utilisateur clique sur l'objet qui se retrouve détouré. Puis l'utilisateur clique sur le bouton "Next". L'utilisateur peut trouver l'image qui a été colorisée à l'aide de l'objet de la référence dans le dossier des résultats.
4. L'utilisateur veut coloriser **un objet sélectionné** sur une image en niveau de gris, **avec les couleurs d'une image de référence.** Il met l'image à coloriser dans le dossier des cibles et met une image à utiliser comme référence dans le dossier des références. Ensuite, l'utilisateur lance le programme de colorisation. Une fenêtre s'ouvre avec l'image à coloriser, l'utilisateur clique sur l'objet qui se retrouve détouré puis il clique sur le bouton "Next", la fenêtre se ferme. Une nouvelle fenêtre s'ouvre avec l'image de référence, l'utilisateur clique sur le bouton "All". L'utilisateur peut trouver l'image dont l'objet sélectionné a été colorisé à l'aide de la référence dans le dossier des résultats.
5. L'utilisateur veut coloriser **un objet qu'il va sélectionner** sur une image en niveau de gris, **avec les couleurs d'un objet qu'il va sélectionner sur une référence.** Il met l'image à coloriser dans le dossier des cibles et met une image à utiliser comme référence dans le dossier des références. Ensuite, l'utilisateur lance le programme de colorisation. Une fenêtre s'ouvre avec l'image à coloriser, l'utilisateur clique sur l'objet qui se retrouve détouré puis il clique sur le bouton "Next", la fenêtre se ferme. Une nouvelle fenêtre s'ouvre avec l'image de référence, l'utilisateur clique sur l'objet qui se retrouve détouré puis il clique sur le bouton "Next". L'utilisateur peut trouver l'image dont

l'objet sélectionné a été colorisé à l'aide de l'objet de la référence dans le dossier des résultats.

2.2.3 Les Besoins Fonctionnels

Segmenter en plusieurs commandes

1. Lancer l'entraînement avec une commande
2. Lancer la colorisation avec une commande
3. Choisir la sauvegarde des cartes d'attention avec un paramètre dans la commande pour la colorisation

Visualiser et sélectionner

— Pour Image cible et Image de référence :

1. Visualiser l'image
2. Sélectionner l'objet de l'image en cliquant sur l'image
3. Sélectionner le fond de l'image en cliquant sur l'image
4. Dé-sélectionner le fond ou l'objet en cliquant sur l'image
5. Visualiser l'objet lorsqu'il est sélectionné contouré par des lignes visibles
6. Visualiser le fond de l'image lorsqu'il est sélectionné par des contours
7. Visualiser lorsque le fond et l'objet sont sélectionnés par un cadre qui contour l'image entière
8. Valider la sélection en cliquant sur un bouton
9. Sélectionner l'image entièrement en cliquant sur un bouton

Coloriser

— Coloriser la sélection de l'image cible avec la sélection de l'image de référence
— Visualiser l'image colorisée
— Visualiser les cartes d'attention en image
— Visualiser les cartes d'attention en fichier binaire
— Effectuer la colorisation de plusieurs images à la suite

2.2.4 Les Besoins Non-Fonctionnels

— L'application doit être capable de coloriser les images avec une grande précision, en évitant les débordements de couleurs et en suivant les contours des objets.

— L'application doit être assez rapide pour permettre une utilisation fluide et une colorisation rapide des images.
— L'interface utilisateur doit être intuitive, permettant aux utilisateurs de coloriser facilement leurs images sans avoir besoin d'une formation ou d'une assistance supplémentaire.
— L'application doit être fiable et ne pas planter ou générer des erreurs lors de l'utilisation.

3 Recherche et méthodologie

3.1 Méthodologie de notre reconduction de couleurs

Le réseau de colorisation principal fourni par H. Carillo *et al.* est basé sur une architecture codeur-décodeur classique de type Unet [13], avec l'ajout de blocs de super-attention [14] qui permet de transférer les indices de couleur de l'image de référence au réseau principal de réseau de colorisation principal.

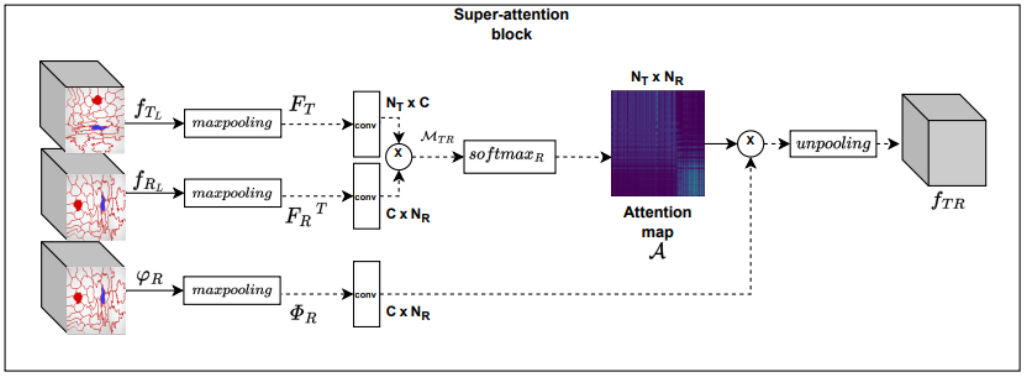


FIGURE 4 – Schéma du bloc de super-attention de la méthode de Hernan Carillo [2]. Cette couche prend en entrée une carte de luminance de référence f_R, une carte de couleur de référence φ_R et une carte de luminance cible f_T, en entrée, et apprend une carte d'attention au niveau du superpixel, le tout au moyen d'une correspondance robuste entre les cartes de caractéristiques encodées à haute résolution.

Cette partie de l'encodage rend possible des opérations telles que la corrélation entre les caractéristiques à haute résolution dans leur cadre de colorisation. La couche de correspondance des super-caractéristiques calcule la corrélation entre les caractéristiques d'apprentissage profond à haute résolution encodées. Cette couche s'inspire du mécanisme d'attention classique [15] sur les images pour obtenir une correspondance robuste entre les super-caractéristiques cibles et les super-caractéristiques de référence. Cependant, contrairement à [14], notre matrice de similarité (carte d'attention) est apprise par le modèle.

Nous reprenons donc le processus du code de H. Carillo *et al.* (cf Figure 4) afin d'y incorporer nos différentes recherches, pour mener à bien la

reconduction de couleur.

3.1.1 Déploiement général des opérations

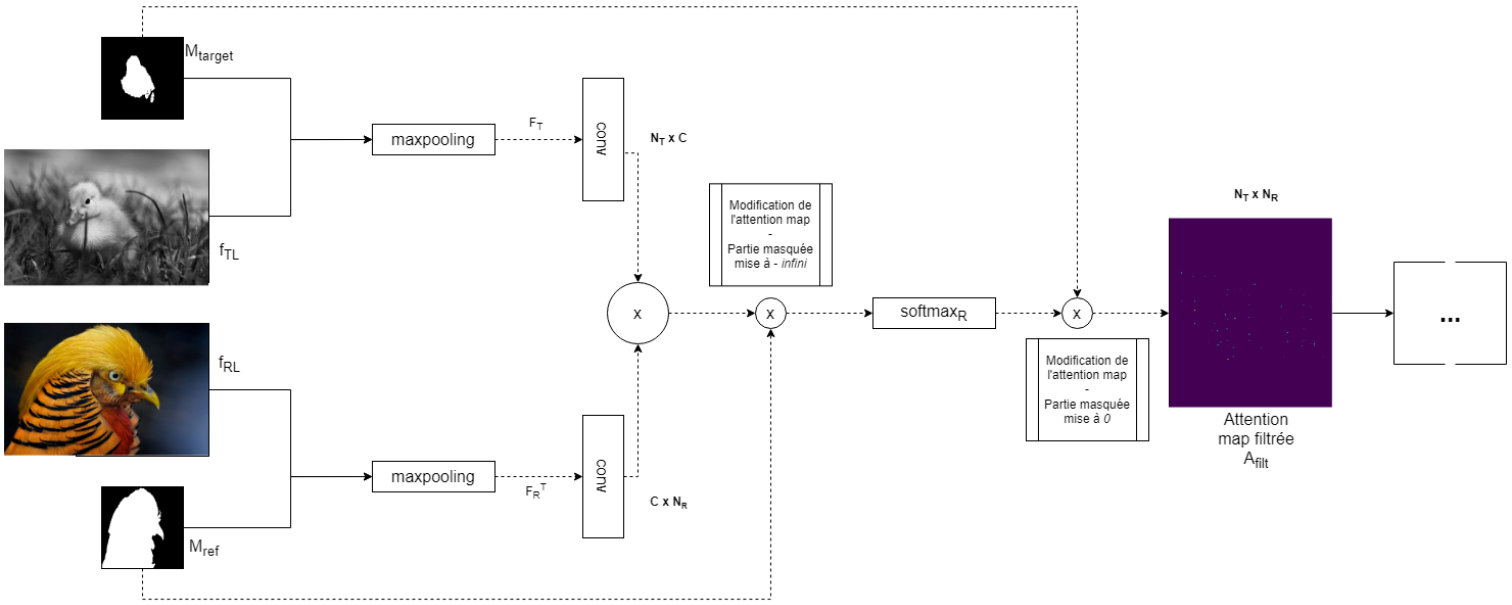


FIGURE 5 – Notre version modifée du bloc de super-attention de la méthode de Hernan Carrillo.

Nous présentons notre approche de la manière suivante :

— Contrairement à la méthode originale, une région précise d'une image peut être prise en compte dans le réseau, dont leur carte de luminance respective (cf. f_{TL} et f_{RL}, Figure 5), et ceci grâce à l'utilisation des masques. Les superpixels de l'algorithme SLIC continuent à apporter leur importance dans la reconnaissance de forme dans ces mêmes masques, et le tout sera précisé dans la partie 3.1.2.

— Comme à l'origine, ces cartes de luminance permettent le calcul de la carte d'attention à la couche l, cependant l'ajout des masques en entrée vont permettre la modification de cette carte d'attention. En effet, le masque M_{ref} apporte une modification avant l'effet du softmax, lorsque le masque M_{target} en apporte une juste après. Ces effets seront mieux expliqués dans la partie 3.1.3.

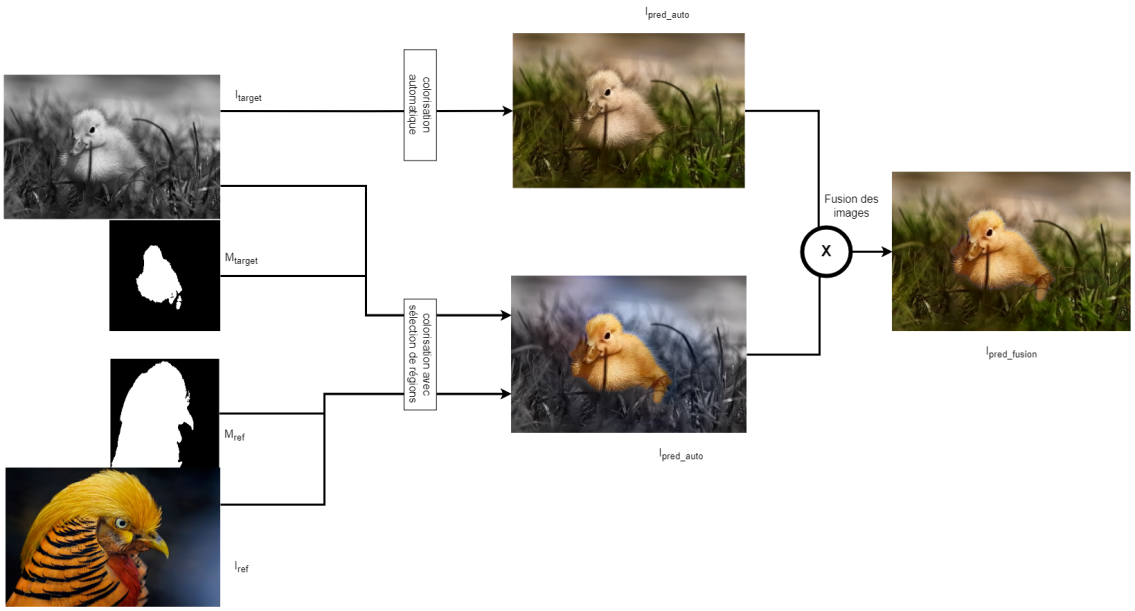


FIGURE 6 – Approche de la méthode avec reconduction de la couleur.

Plus globalement, on peut observer le déroulement général de ce travail sur la Figure 6. Effectivement, si une coloration par sélection a lieu grâce à l'utilisateur (I_{pred_select}), une colorisation automatique a lieu en parallèle (I_{pred_auto}).

Ainsi, en fonction de la région voulue à coloriser par l'utilisateur, nous avons, par fusion, une colorisation générale I_{pred_fusion} de l'image cible, avec reconduction de couleur par l'interactivité homme/machine. Le déroulement de nos recherches et résultats correspondant à cette approche sont développés dans la partie 3.2

3.1.2 Masques et superpixels

Le masque est représenté comme une matrice à 2 dimensions de la taille de l'image et constitué exclusivement de 0 et de 1. Les différentes librairies graphiques vont interpréter les pixels originaux situés au niveau des 1 du masque comme étant les pixels à conserver, les autres seront ignorés.

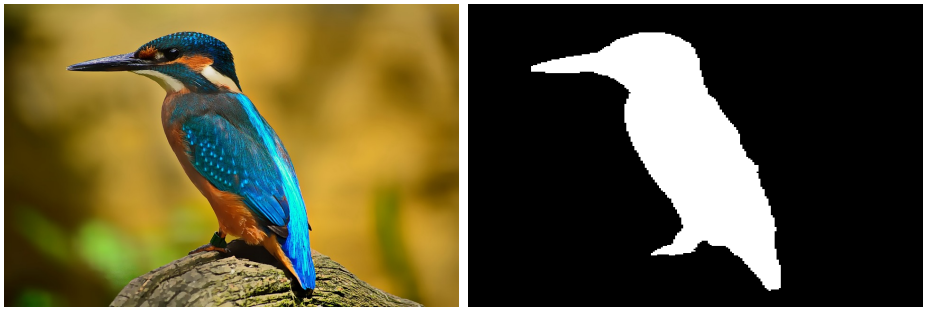


FIGURE 7 – Exemple de masque représentant un oiseau

Outre un intérêt graphique, les masques binaires sont surtout utilisés pour identifier une partie d'une image et l'associer à une classe d'objet, c'est ce que l'on appelle la segmentation. Ici dans l'exemple démontré dans la Figure 7, le blanc représente la valeur 1 à garder dans l'image, et le noir la valeur 0 à ignorer. Ainsi en multipliant les 2 images on se retrouverait avec une image d'oiseau avec un fond noir. Cette méthode aurait pour intérêt de garder uniquement une certaine partie d'une image référence pour une meilleure reconduction de couleurs.

Afin d'obtenir les superpixels inclus dans le masque, deux méthodes sont possibles : l'utilisation de SLIC et des barycentres ou bien la méthode maskSLIC.
Nous avons appliqué la première manière qui consiste à obtenir la carte des superpixels de toute l'image avec l'algorithme SLIC présenté au début dans la section 1.2.1. Puis en utilisant le barycentre de chaque superpixel, nous les avons filtrés en ne gardant que ceux dont celui-ci est inclus à l'intérieur du masque. (cf. Fig. 8)
Pour ce qui est de la deuxième méthode, MaskSLIC restreint l'algorithme SLIC à s'appliquer qu'à l'intérieur du masque en lui donnant le masque en paramètre.

FIGURE 8 – Filtrage des superpixels

3.1.3 Carte d'attention

Les cartes d'attention nous montrent à quel point chaque superpixel de l'image à coloriser (la target) corrèle avec ceux de la référence.
Nous avons modifié ces cartes afin de limiter celles-ci aux régions du masque. En effet, nous voulons restreindre le calcul d'attention que sur les superpixels sélectionnés afin que ceux hors du masque ne fausse pas le calcul en y contribuant trop.
Pour cela, nous avons filtré les colonnes de la matrice d'attention avant de faire le softmax, puis filtréé les lignes après le softmax. Le softmax permet de renormaliser les valeurs d'une ligne afin que leur somme fasse 1 pour chaque ligne.
Les colonnes de la carte d'attention correspondent aux superpixels de l'image de référence et les lignes correspondent à ceux de l'image à coloriser. Afin de filtrer les colonnes, nous avons mis toutes celles correspondantes aux superpixels qui sont en dehors du masque de l'image de référence à -∞ (-100000 dans notre code). En faisant le softmax le -∞ va se traduire en 0 en résultat de softmax ce qui va donc permettre d'ignorer ces colonnes.
Après avoir fait le softmax, nous avons filtré les lignes en mettant à 0 celles qui correspondent aux superpixels non sélectionnés sur l'image à coloriser. (cf. Fig. 9)
Nous obtenons donc une carte d'attention qui met plus de poids à l'intérieur du masque par rapport à lorsqu'elle était inchangée.

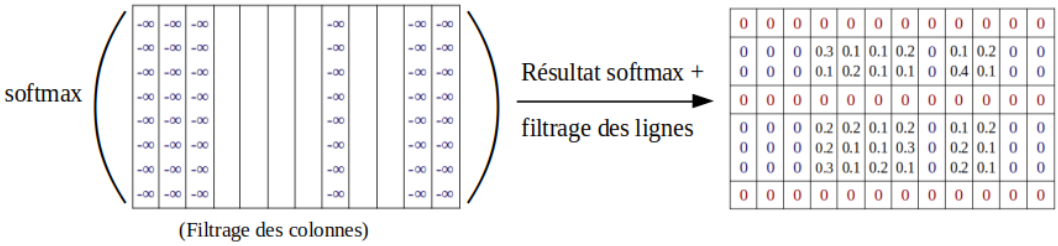
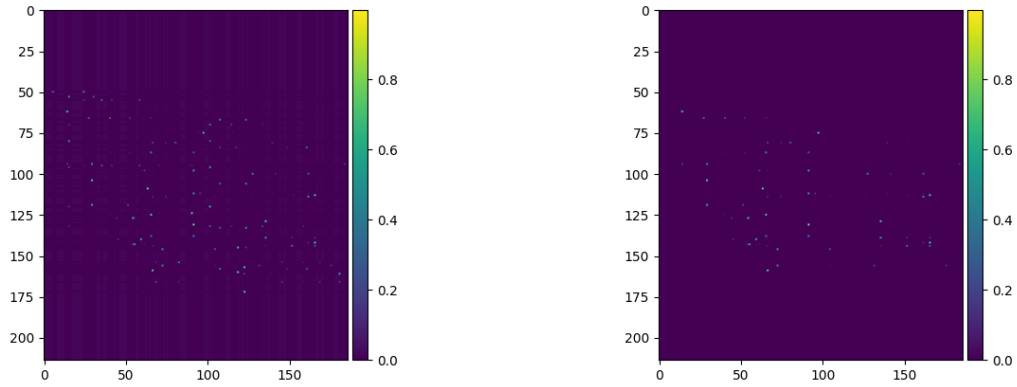


FIGURE 9 – Modification matrice d'attention

Comme nous pouvons le remarquer sur la figure 10, le nombre de pixels différents de zéro diminue sur la carte d'attention modifiée.



Avant Après

FIGURE 10 – Cartes d'attention avant et après la prise en compte du masque pour le 1er niveau de super attention

3.2 Déroulement et résultats des recherches

Avant que nous ne commencions à modifier les travaux de nos clients, (à part une variable qui à permis d'amplifier la colorisation,) les résultats, présents dans la Figure 11 de colorisation étaient, soit obtenus après une colorisation automatique (donc uniquement grâce au réseau déjà entraîné), soit obtenus grâce à une colorisation guidée à l'aide une référence colorée et du même réseau entraîné.

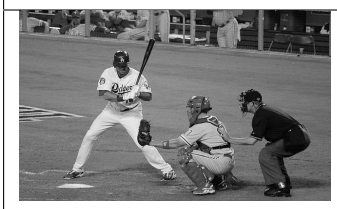
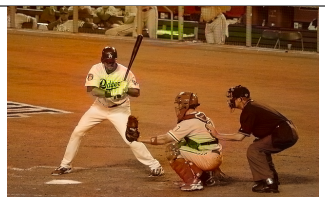
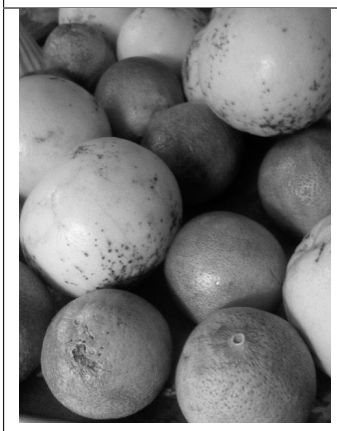
Image à coloriser	Colorisation Automatique	Colorisation avec les référence de la Figure 19
		
		
		

FIGURE 11 – Exemples d'image colorisée par le programme de base.

Notre but à ce moment là était de coloriser une zone de l'image en niveau de gris avec l'aide d'une autre zone de l'image de référence. Nous avons décidé de donner le contrôle à l'utilisateur, celui-ci pouvant directement sélectionner les super-pixels voulus sur les deux images. Nous avons donc réussi à coloriser les régions voulues avec les zones sélectionnées de la référence comme nous pouvons l'observer avec les exemples de la figure 12. Nous pouvons aussi remarquer certains problèmes de colorisation sur le troisième exemple, où du bleu est présent sur la partie qui n'est pas à coloriser.

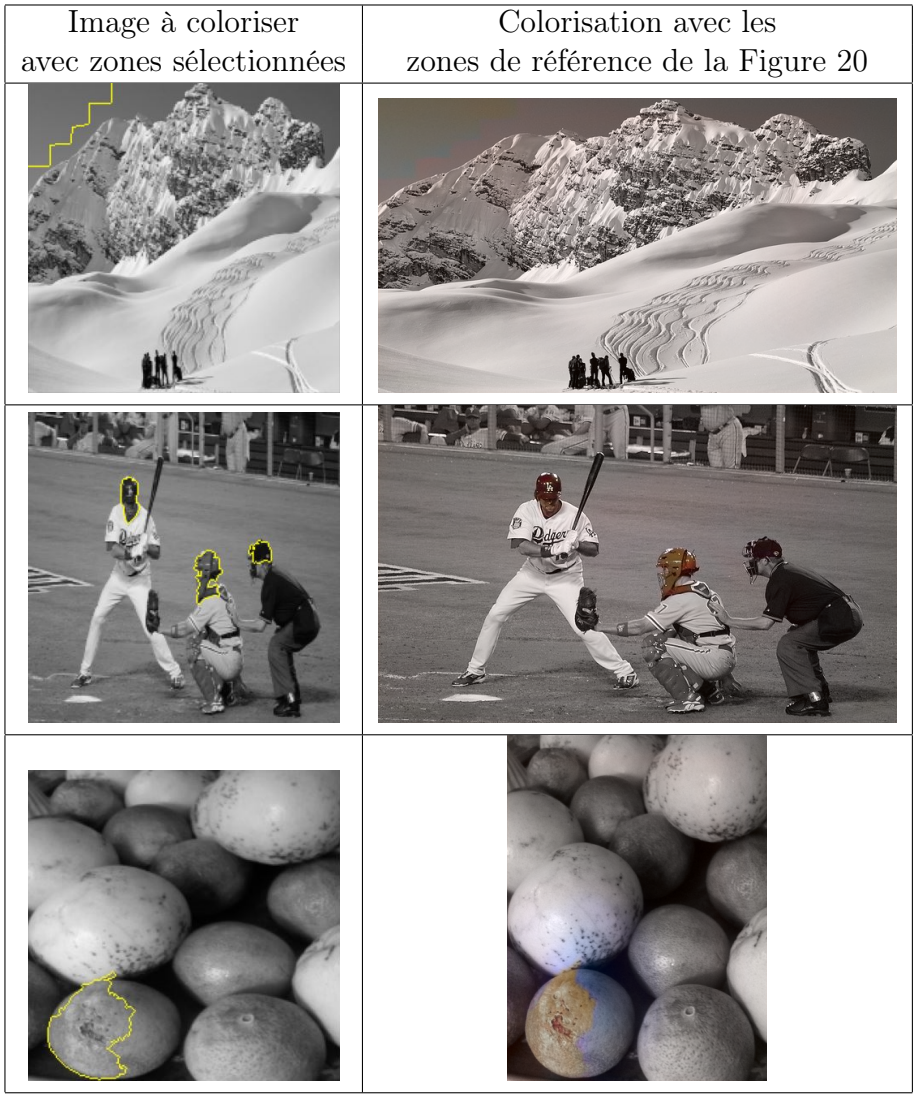


FIGURE 12 – Exemples d'image colorisée par sélection de super-pixels.

Ensuite, les besoins de nos clients se sont affinés. Nous avions désormais besoin de sélectionner un objet directement sur chaque image. Ce qui permettait un gain de temps conséquent, car au lieu de devoir cliquer plusieurs fois sur l'objet, il suffisait d'un seul clic pour le sélectionner. Pour correspondre à ce nouveau besoin, il nous a donc fallu de nouvelles images et des masques correspondants à celles-ci. Des bases de données déjà existantes nous ont été proposées. Cependant, étant trop lourdes à télécharger, complexes à utiliser,

et le temps nous étant compté, nous avons opté pour la constitution de nos propres images. Ces dernières sont toutes libres de droit et issues du site [12], ainsi que la création de nos propres masques. Pour la plupart de nos images, nous utilisons leur version colorée comme référence, mais nous avons quelques exemples comme celui de la deuxième image de la Figure 13, où la référence couleur est tout autre.

Dans ce tableau nous pouvons comparer, notre colorisation, (la colorisation de l'objet seul qui a été colorisé a l'aide des couleurs de l'objet sur la référence,) avec la colorisation automatique. Nous pouvons observer que notre colorisation est plus efficace concernant les objets sélectionnés mais que le réseau colorise aussi l'extérieur de ses objets. Nous pouvons clairement l'observer autour du caneton du deuxième exemple, une colorisation bleutée. Ainsi qu'à gauche du chien du troisième exemple, une colorisation rouge.

Après avoir obtenu une colorisation d'un objet précis de l'image cible, nos clients avaient la volonté que le reste de l'image soit également colorisé de manière automatique. Nous avons donc choisi d'effectuer deux colorisations. L'une d'entre elles est la colorisation automatique pour la partie qui n'a pas été sélectionné par l'utilisateur et l'autre est la colorisation ciblée de la zone de l'image à colorisée par ce qu'il veut de l'image de référence.

Dans le tableau de la Figure 14 nous pouvons comparer notre colorisation (colonne 3) à la colorisation automatique (colonne 1) mais aussi à la colorisation de base de nos clients qui est la colorisation par référence (colonne 2). Notre colorisation n'est pas parfaite sur la plus part des exemples, les problèmes viennent de la segmentation SLIC mais aussi de notre fusion d'image qui dépend d'un mask que l'on redimensionne, c'est pourquoi il y a un liseré non coloré qui entoure le caneton de l'exemple 2, colonne 3. Le positif de notre colorisation est que pour certaine image, elle est mieux, même si c'est subjectif, nous pouvons affirmer que les couleurs, de la langue et les oreilles du chien de l'exemple 3 colonne 3, sont plus vibrantes que dans les deux autres colorisations.

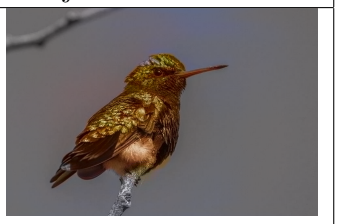
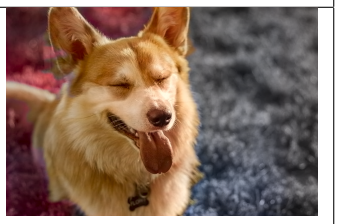
Image à coloriser	Colorisation automatique du réseau	Notre colorisation de l'objet guidée avec l'objet de référence 21
		
		
		
		

FIGURE 13 – Exemples d'images colorisées par sélection d'objet.

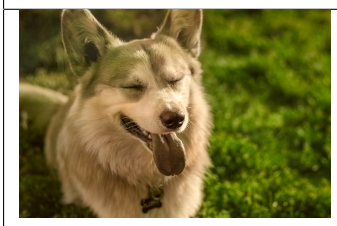
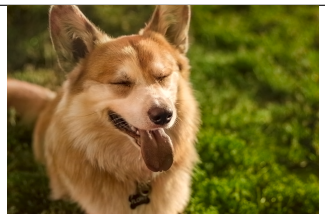
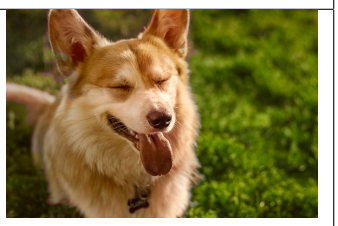
Colorisation Automatique du réseau	Colorisation entière guidée par la référence entière 21	Notre Colorisation par fusion
		
		
		
		

FIGURE 14 – Exemples d'images colorisées entièrement.

4 Architecture et implémentation

4.1 Architecture

4.1.1 Version avec amélioration

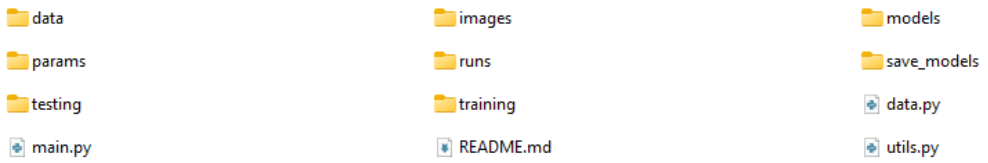


FIGURE 15 – Architecture de la version originale

Nous avons globalement gardé l'architecture du code fourni car les modifications devaient être minimales.

Ainsi nous allons développer la liste des répertoires et leur contenu (cf. Figure 15) :

- `data/` : ce répertoire comporte les données nécessaires à l'entraînement, avec les 2 matrices de similarités entre les images cibles et les images réferences, ainsi que l'échantillon de ces images.
- `images/` : comporte les images nécessaires pour la création du README, visible sur le dépôt Git.
- `models/` : comporte les modèles `model_perceptual.py` et `super_encoder_unet.py`, servant à entraîner les données.
- `params/` : correspond à des fichiers comportant des données de paramètres généraux, accessible pour les différents types de lancement.
- `runs/` et `save_models/` : contient les événements de TensorBoard pour le suivi de l'entraînement et de la validation des modèles, ainsi que le modèle enregistré.
- `testing/` et `training/` : on y retrouve les deux programmes correspondant aux processus de lancement respectifs de la colorisation et de l'entraînement à la colorisation.
- `main.py` : script de lancement des différents processus, à savoir 'train' ou 'test'.
- `data.py` et `utils.py` : fichiers contenant des fonctions utiles au bon fonctionnement du programme à part entière.
- `README.md` : Readme du dépot Git.

4.1.2 Version avec ajout de fonctionnalités

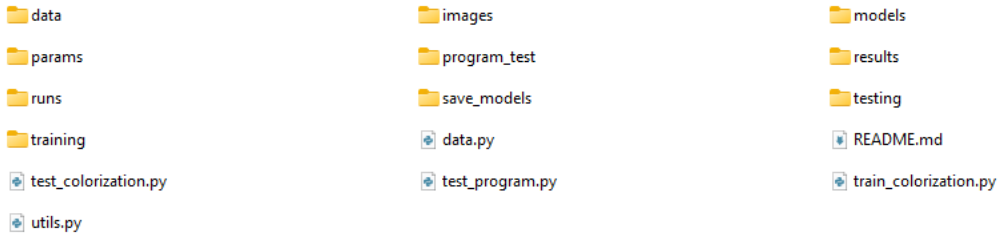


FIGURE 16 – Architecture de la Version avec ajout de fonctionnalités

Le principal changement dans l'arborescence de fichiers se joue dans la séparation des scripts de lancement, comme on peut le voir sur la Figure 16, avec les scripts `test_colorization.py` et `train_colorization.py`.

Un dossier `program_test/` a été rajouté, comportant nos différents tests du projet.

Enfin, le dossier `results/` intègre les résultats de notre approche de colorisation, avec comme sous dossiers :

- `attention_maps/` : les cartes d'attentions correspondantes aux prédictions d'images.
- `color_treatment_results_auto/` : les prédictions d'images liées à la sélection de région de l'utilisateur (reconduction de couleurs)
- `color_treatment_fusion/` : les fusions des images pour une colorisation totale.

4.2 Implémentation

4.2.1 Mise au propre

Nous avons renommé des variables, fonctions, fichiers avec des noms plus adaptés. Nous avons aussi restructuré le code pour le mettre à la norme PEP8. Nous devions également supprimer les bouts de code, ou fonctions non utilisées. La seule partie fonctionnelle que nous avons apporté à cette partie est la sauvegarde des matrices d'attentions à l'aide d'une option dans la ligne de commande.

4.2.2 Affichage Interactif

La fonction `select_and_plot_mask()`, permet de sélectionner une partie d'une image en utilisant une carte de segmentation de superpixels et en utilisant la souris pour sélectionner les superpixels.

Plus précisément, la fonction prend en entrée une image, une carte de segmentation des superpixels de l'image, un masque (qui correspond à l'objet présent sur l'image) et un paramètre booléen optionnel `ref`.

La fonction crée un plot avec l'image de base, puis écoute les clics de souris sur le plot. À chaque clic de souris, la fonction utilise la carte de segmentation pour identifier le superpixel sélectionné et ajoute ce superpixel au masque de sélection. Elle met ensuite à jour l'image en appliquant le masque de sélection et en affichant les limites des superpixels sélectionnés.

Deux boutons sont également présents sur le plot : le bouton "Next" permet de sauvegarder l'image avec la partie sélectionnée, tandis que le bouton "All" permet de sauvegarder l'image de référence en entier. L'un des fonctionnements possibles est illutré dans le diagramme de séquence de la Figure 29

5 Tests

5.1 Tests de non régression

1. Pour tester la fontion `select_and_plot_mask` :
   - Tester la fonction avec une image en niveaux de gris pour vérifier si la conversion en image RGB se déroule correctement.
   - Tester la fonction avec une image en couleur pour vérifier que l'affichage est correct.
2. Pour tester la fonction `add_id_to_list` :
   - Vérifier que la fonction ajoute bien les superpixels de la zone à colorer dans la liste correspondante.
   - Vérifier que si save_masked est faux, la fonction ajoute tous les superpixels de l'image à la liste correspondante.
3. Pour tester la fonction `apply_mask_to_image` :
   - Vérifier que la fonction applique correctement le masque à l'image en gardant seulement la partie de l'image à colorer si `save_masked` est vrai.

— Vérifier que si `save_masked` est faux, la fonction ne modifie pas l'image.

4. Pour tester la fonction `save_image_and_mask` :
   — Vérifier que la fonction sauvegarde correctement l'image et le masque en PNG pour la référence et la target.
5. Pour tester la fonction `select_superpixels_in_mask` :
   — Vérifier que la fonction sélectionne correctement les superpixels dans le masque en fonction de l'objet ou du fond.
   — Vérifier que si le barycentre d'un superpixel n'est pas dans le masque de l'objet ou du fond, le superpixel est retiré du masque.
6. Pour tester la fonction `toggle_border_highlight` :
   — Vérifier que la fonction ajoute bien un cadre jaune autour de l'image si tous les superpixels sont sélectionnés dans le masque.
   — Vérifier que la fonction enlève le cadre jaune s'il était présent et que tous les superpixels ne sont pas sélectionnés dans le masque.
7. Pour tester la fonction `save_attention_maps`
   — Vérifier que la fonction crée le nombre attendu de fichiers PNG et NPY.
   — Vérifier que les fichiers créés ont les noms attendus.

5.2 Test du Singe

Durant notre projet, il semblait parfois plus compliqué et long de tester notre interface avec un test automatique. Nous avons donc effectué chacune de notre côté de nombreux tests manuels, où l'on cliquait partout, pour tester les limites de notre interface. Nous répétions à chaque modification importante pouvant impliquer à la fois de nouveaux résultats mais aussi de nouvelles erreurs.

6 Suivi de projet

6.1 Méthodologie d'organisation

Nous avons choisi de mettre en place des sprints de 7 jours pour notre projet. Ce choix s'est avéré très efficace car il nous a permis de travailler de manière concentrée sur des objectifs précis et de suivre régulièrement notre

avancement. Chaque sprint était rythmé par un rendez-vous hebdomadaire avec nos clients, qui nous permettait de faire le point sur les avancées, les difficultés rencontrées et les attentes des clients.

Pour gérer l'attribution et la priorisation des tâches, nous avons utilisé un Kanban sur l'application Trello. Nous avons créé des colonnes pour chaque étape de notre processus et chaque tâche était assignée à un membre de l'équipe, avec, parfois, une description détaillée pour une meilleure compréhension des tâches les plus complexes.

Nous avons utilisé Git pour versionner notre code et avoir différentes branches en fonction des deux versions souhaitées par nos clients. Cette approche nous a permis de travailler en parallèle sur différentes fonctionnalités du projet, sans risque de perdre des données.

Pour communiquer avec nos clients, nous avons utilisé Discord, un outil de communication efficace. Nous avons créé un canal dédié pour partager des mises à jour importantes sur l'avancement du projet et pour échanger avec nos clients sur leurs attentes et leurs retours. Nous avons également organisé des appels avec nos clients pour remplacer certains rendez-vous .

Notre méthodologie d'organisation a été un élément clé de la réussite de notre projet. En utilisant des sprints réguliers, un Kanban pour l'attribution et la priorisation des tâches, Git pour versionner notre code et Discord pour communiquer avec nos clients, nous avons pu travailler de manière efficace.

6.2 Liste des Tâches, Priorités & Complexités

Avant de commencer à programmer, il est essentiel de se renseigner sur l'état actuel de la technologie pour éviter de travailler à l'aveugle. Tout au long du projet, de nouvelles tâches de veille technologique (Figure 22) ont été nécessaires en fonction des besoins croissants.

Des tâches de gestion de projet ont également été mises en place pour une meilleure organisation (Figure 23). Pour éviter de se perdre dans nos recherches, des tâches de rédaction régulières ont également été effectuées (Figure 25).

Pour améliorer le code, des tâches de mise au propre ont été effectuées en plusieurs exemplaires (Figure 26), pour chaque fichier, dossier et version.

Pour ajouter des nouvelles fonctionnalité, nous avons également effectué des tâches de segmentation pour les cartes de segmentation des images cibles et de référence (Figure 28), à la fois pour les images à coloriser et pour les

images de référence. Enfin, nous avons eu des tâches de segmentation du code (Figure 24) et de création de cartes d'attention (Figure 27).

7 Conclusion

Parmi toutes les problématiques qui nous étaient proposées, nous avons choisi le projet qui a éveillé le plus d'intérêt et de créativité chez nous trois. La colorisation d'image était déjà un sujet de discussion récurrent durant nos études, et pouvoir enfin travailler dessus et potentiellement le faire innover a été un moteur principal pour le choix de notre projet de fin d'études.

Notre objectif initial était d'améliorer une méthode de colorisation d'image par apprentissage profond en y ajoutant une carte de segmentation, afin de permettre une colorisation plus précise et plus guidée par l'utilisateur. Pour ce faire, nous avons développé une méthode de fusion de deux colorisations qui a permis d'obtenir des résultats satisfaisants, malgré certaines limites liées à la nécessité d'avoir un masque pour la zone colorisée, ainsi qu'à la qualité du découpage fourni par la méthode de colorisation SLIC.

Notre principale contribution consiste en cette méthode de fusion, ainsi qu'en l'identification de ses limites, ce qui ouvre la voie à de futures recherches et améliorations. Notre projet a ainsi participé à la recherche en traitement d'image et pourrait être utile dans diverses applications, comme la restauration d'images et la création de contenu visuel pour l'industrie créative.

Bien que notre méthode ne soit pas destinée à une utilisation grand public, nous sommes convaincus qu'elle pourra être améliorée et étendue dans le futur. Nous avons également apporté des contributions importantes au projet en renommant les variables et les fonctions, en restructurant le code et en ajoutant une fonction de sélection d'objet à l'aide de la souris.

Si nous avions disposé de davantage de temps, nous aurions pu améliorer l'interface en intégrant les sélections d'objet pour l'image à colorier et l'image de référence couleur dans une même fenêtre. Nous aurions également pu résoudre les problèmes techniques de l'interface ainsi que les problèmes de colorisation. Nous aurions peut-être pu contourner les problèmes des superpixels qui ne détouraient pas l'objet comme prévu, et nous aurions pu améliorer la fusion des deux colorisations en évitant le liseré gris qui demeure et qui entrave la colorisation. Enfin, nous aurions pu ajouter plus d'options dans les commandes pour utiliser l'application.

Notre méthodologie d'organisation a été un élément clé de la réussite de notre projet. En utilisant des sprints réguliers, un Kanban pour l'attribution et la priorisation des tâches, Git pour versionner notre code et Discord pour communiquer avec nos clients, nous avons pu travailler de manière efficace.

Malgré quelques difficultés rencontrées en cours de route, nous sommes

fiers du travail accompli et nous pensons que notre projet a apporté une contribution au domaine de la colorisation d'image par apprentissage profond. Nous espérons que nos travaux pourront inspirer de futurs étudiants et chercheurs à poursuivre les développements dans ce domaine passionnant et en constante évolution.

8 Bibliographie

[1] Aurélie Bugeau, Michaël Clément, Vinh-Thong Ta, Vincent Lepetit (2020) , "ANR postprodLEAP". https://www.labri.fr/perso/bugeau/PostProdLEAP/. (Visited on 2023-02-01).

[10] Colorization Transformer,Manoj Kumar, Dirk Weissenborn, Nal Kalchbrenner. International Conference on Learning Representations, 2021.

[11] SLIC Superpixels, Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. EPFL Technical Report 149300, 2010.

[12] Photos libres de droits, pixabay. https://pixabay.com/fr/photos/,2023.

[13] Ronneberger, o., fischer, p., brox, t. : U-net : Convolutional networks for biomedical image segmentation. in : International conference on medical image computing and computer assisted intervention,2015.

[14] Hernan Carrillo, Michaël Clément, Aurélie Bugeau : Non-local matching of superpixel-based deep features for color transfer. international conference on computer vision theory and applications,2022.

[15] Zhang, b., he, m., liao, j., sander, p.v., yuan, l., bermak, a., chen, d. : Deep exemplar-based video colorization. in : Conference on computer vision and pattern recognition. (2019).

[16] UNet — Line by Line Explanation. https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5. (Visited on 2023-03-15).

[2] Super-attention for exemplar-based image colorization, Hernan Carrillo, Michaël Clément, Aurélie Bugeau. 16th Asian Conference on Computer Vision (ACCV), 2022.

[4] SLIC Superpixels Compared to State-of-the-Art Superpixel Methods, Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, Sabine Süsstrunk. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012.

[5] Colorful Image Colorization, Richard Zhang and Phillip Isola and Alexei A. Efros. European Conference on Computer Vision, 2016.

[6] Colorization using Optimization, Anat Levin, Dani Lischinski, Yair Weiss. ACM Transactions on Graphics, 2002.

[7] Variational Exemplar-Based Image Colorization, Aurélie Bugeau, Vinh-Thong Ta, Nicolas Papadakis. IEEE Transactions on Image Processing, 2014.

[8] Deep Colorization, Zezhou Cheng, Qingxiong Yang, Bin Sheng. International Conference on Computer Vision, 2015.

[9] Learning Representations for Automatic Colorization, Gustav Larsson, Michael Maire, Gregory Shakhnarovich. European Conference on Computer Vision, 2016.

9 Annexe

9.1 Manuel d'utilisation

9.1.1 Présentation de l'application

Notre application a pour but, de donner la possibilité à l'utilisateur de choisir quelle partie de l'image en niveau de gris, et avec quelle partie de l'image de référence il souhaite effectuer la colorisation. Ces parties peuvent être un objet sur l'image, le fond de l'image ou l'image entière. Mais l'utilisateur peut choisir de ne rien coloriser en ne sélectionnant rien pour l'image en niveau de gris. Il peut aussi choisir une colorisation automatique en ne sélectionnant rien pour l'image de référence.

Notre application n'est pas adaptée au grand publique. Le publique visé est constitué de chercheurs en informatique et intelligence artificielle. L'utilisateur va devoir constituer un ensemble de données particulier que nous définissons dans la partie "Importation et exportation d'image". Il va d'autre part devoir placer les données ainsi constituées dans des dossiers bien précis. Lors de son lancement notre application met, (en fonction des systèmes,) entre 40 secondes et 2 minutes à se lancer. Lorsque notre application se lance, c'est pour coloriser toutes les images du dossier dans lequel elles se trouvent, et l'application se termine naturellement après que toutes les images soient passées par le processus de colorisation. Après de la colorisation, l'image colorisée ainsi que les cartes d'attention si elles étaient voulue par l'utilisateur sont récupérables dans un dossier précis.

9.1.2 Exigences système

Pour que notre application fonctionne, les principales dépendances nécessaires à son fonctionnement sont les bibliothèques suivantes :

— torch
— torch-scatter
— torchvision
— einops
— ftfy
— kornia
— lpips
— matplotlib
— numpy

— Pillow
— scikit-image
— scipy
— tensorboard

9.1.3 Lancement de l'application

Pour lancer l'application pour une colorisation, dans le terminal il faut rentrer la commande suivante :

```
python test_colorization.py
```

Pour lancer l'application pour une colorisation avec la sauvegarde des cartes d'attentions, il faut ajouter ce paramètre à la commande :

```
--save_attention_maps
```

Pour lancer l'application pour entraîner le modèle de colorisation, dans le terminal il faut rentrer la commande suivante :

```
python train_colorization.py
```

9.1.4 Interface utilisateur et Colorisation

Pour chaque colorisation, notre interface s'ouvre deux fois de suite. La première fois pour l'image à coloriser et la deuxième pour l'image de référence couleur. Dans les deux cas le fonctionnement est le même.

Tout d'abord, l'image à coloriser est affichée, ainsi que deux bouton "Next" et "All". Si le bouton "Next est directement cliqué, sans qu'il n'y ait aucune sélection, il n'y aura pas de colorisation, peut importe ce qui sera sélectionné sur l'image de référence le résultat sera l'image en niveau de gris du début.

Si le bouton "All" est cliqué, toute l'image est sélectionnée pour être colorisé, même si une sélection avait été effectué sur l'image.

S'il y a un clique sur l'image, la zone correspondante sera entouré en jaune comme sur la première image de la Figure 17. Ici, l'oiseau présent sur l'image a été sélectionné, mais si le fond avait été sélectionné, la représentation aurait été la même.

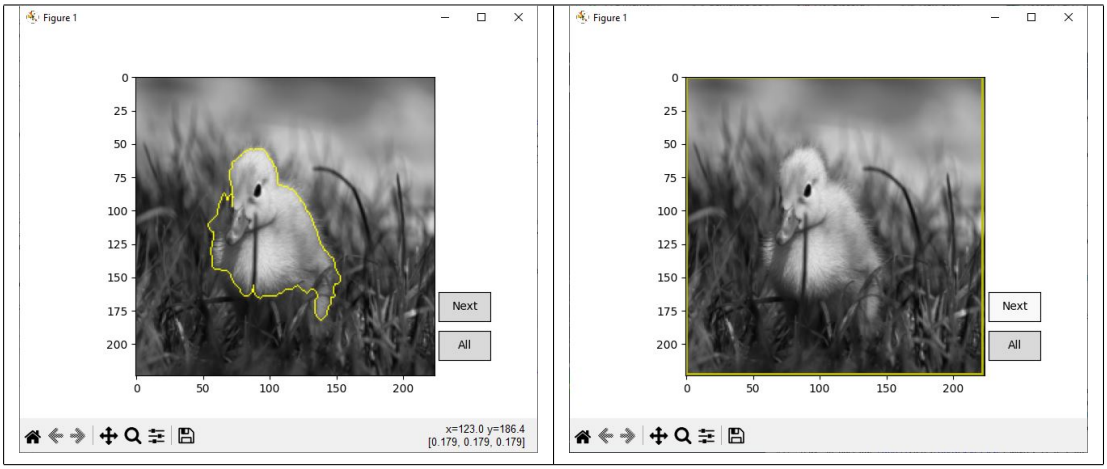


FIGURE 17 – Interaction interface image à coloriser.

Si les deux zones de l'image sont sélectionnées, l'objet et le fond alors un cadre jaune entoure l'image comme sur la deuxième image de la Figure 17.

Ensuite, si le bouton "Next" est cliqué, la zone sélectionnée sera la zone qui sera colorisée par notre méthode.

Ensuite, la fenêtre de l'interface se ferme et une nouvelle s'ouvre, l'image de référence est affichée, ainsi que deux bouton "Next" et "All" comme pour l'image à coloriser.

Si le bouton "Next" est directement cliqué, sans qu'il n'y ait aucune sélection, la colorisation sera faite automatiquement sans prendre en compte les couleurs de cette image de référence.

Si le bouton "All" est cliqué, toute l'image est sélectionnée pour référence couleur, même si une sélection avait été effectué sur l'image.

Tout comme pour l'image à coloriser, la zone cliquée sur l'image donnera les même effet de contourage que l'on peut voir dans la Figure 18. Et si le bouton "Next" est cliqué après une sélection, la zone sélectionné servira pour la colorisation de notre méthode.

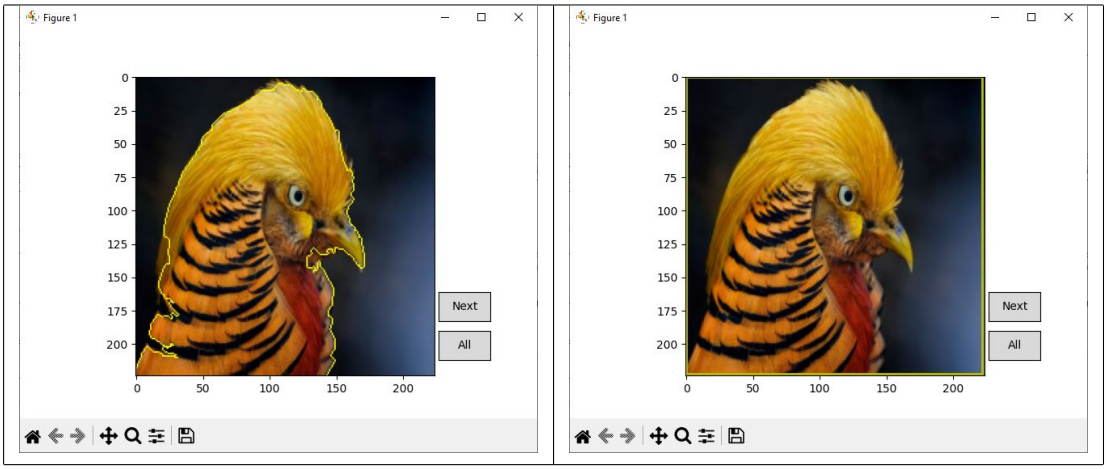


FIGURE 18 – Interaction interface image référence.

9.1.5 Importation d'images et exportation d'images

Pour chaque colorisation, il y a un lot d'image, elles ont un identifiant qui commence à 0. Dans le dossier `data\target` il y a les images en niveau de gris. Elles sont nommées `[id]_target.png` et leurs masques, qui représentent l'objet présent sur l'image, sont dans le dossier `data\mask_target` et sont nommés `[id]_mask.png`. Dans le dossier `data\ref` il y a les images de référence qui sont nommées `[id]_ref.png`. Leurs masques qui représentent l'objet présent sur l'image, est dans le dossier `data\mask_ref` et sont nommés `[id]_mask.png`.

Pour voir le résultat de la colorisation de la sélection l'utilisateur va dans le dossier `results\color_treatment_results`

Pour voir le résultat de la colorisation automatique l'utilisateur va dans le dossier `results\color_treatment_results_auto`

Pour voir le résultat de la colorisation par fusion l'utilisateur va dans le dossier `results\color_treatment_results_fusion`

Les images colorisées seront respectivement nommées `[id]_ pred.png`

Pour voir les cartes d'attention de la colorisation l'utilisateur va dans le dossier `results\attention_maps\[id]\` Les cartes d'attention sous format image png et seront nommées `superattention_1.png`, `superattention_2.png`, `superattention_3.png`, `superattention_4.png`. Les cartes d'attention sous format de fichier binaire seront nommées `superattention_1.npy`, `superattention_2.npy`, `superattention_3.npy`,

`superattention_4.npy`

9.1.6 Problèmes

Si l'utilisateur ferme l'une des fenêtres en cliquant sur la croix, le programme va crash simplement parce qu'aucune sélection n'a été effectué. Nous avons essayé de faire une condition où l'image est colorisée automatiquement lorsque la personne ferme la fenêtre mais cette condition se déclenche aussi lorsque la fenêtre est fermée pendant la sélection des masques.

Nous ne garantissons pas le fonctionnement de la partie entraînement du modèle de colorisation car elle ne fait pas partie des besoins de nos clients, nous n'y avons pas participé.

Si l'image à coloriser est déjà en couleur, ce n'est pas pris en charge.

Que ce soit le fond ou l'objet la représentation de ce qui a été cliqué est la même, il aurait fallu faire une modification pour que la partie du fond sur les bords de l'image soit en jaune aussi mais nous n'avons pas eu le temps.

Arrêter le programme en cours de route et proprement n'est pas possible, il faut mener à bien les colorisations de toutes les images jusqu'au bout sinon il faut fermer les fenêtres et faire crash le programme dans ce cas la colorisation en cours n'est pas effectué mais toutes les précédentes le sont.

9.2 Images

FIGURE 19 – Références couleurs de la colorisation de la figure 11.

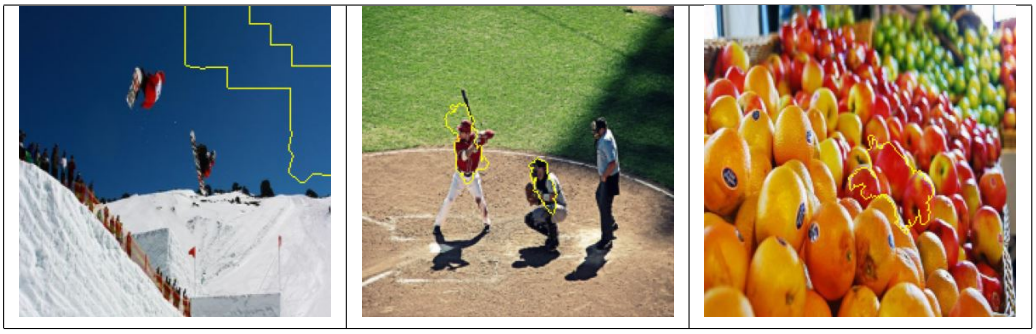


FIGURE 20 – Sélections des Références de la colorisation de la figure 12.

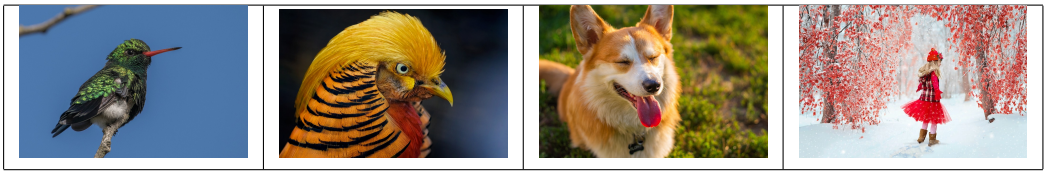


FIGURE 21 – Sélections des Références de la colorisation des figures 13 et 14.

9.3 Tâches

Tâches	Priorité	Complexité
S'informer sur les super-pixels	Haute	Faible
S'informer sur la segmentation SLIC	Haute	Faible
S'informer sur Unet	Moyenne	Faible
S'informer sur les methodes d'attention	Moyenne	Moyenne
S'informer sur les methodes de cross-attention	Moyenne	Moyenne
Lire l'article en rapport avec le projet	Haute	Moyenne
S'informer sur les bases de données d'image ayant des maques	Haute	Moyenne

FIGURE 22 – Tâche de Veilles technologiques

Tâches	Priorité	Complexité
Créer un git	Haute	Faible
Configurer un environnement de travail par machine	Haute	Moyenne
Créer deux branches git différentes, une par version	Haute	Faible
Récupérer des images libres de droit pour des exemples	Faible	Faible

FIGURE 23 – Tâche de Gestion

Tâches	Priorité	Complexité
Segmenter le code du train et du test	Haute	Faible
Adaptation des fonctions crées	Haute	Haute
Création de fonctions nécessaires	Haute	Moyenne

FIGURE 24 – Tâche de Segmentation du code

Tâches	Priorité	Complexité
Rédiger le README	Haute	Faible
Rédiger le rapport du projet	Haute	Haute
Rédiger les questions à poser aux clients	Haute	Faible
Effectuer les tests des nouvelles fonctionnalités	Haute	Haute

FIGURE 25 – Tâche de Rédactions

Tâches	Priorités	Complexités
Enlever chaque partie inutilisée dans le code	Moyenne	Faible
Redéfinir la nomenclature	Moyenne	Moyenne
Redéfinir l'architecture	Moyenne	Moyenne
Mettre le code à la norme PEP8	Moyenne	Faible

FIGURE 26 – Tâche de Mise au propre du code

Tâches	Priorités	Complexités
Comprendre le fonctionnement des cartes	Haute	Haute
Trouver les carte dans le réseau	Haute	Haute
Redimensionner les masques pour chaque niveau de super attention	Haute	Faible
Récupérer les listes des superpixels sélectionnés pour chaque niveau	Haute	Faible
Modifier les cartes en filtrant lignes et colonnes	Haute	Haute
Sauvegarder les cartes en png et en npy	Haute	Faible

FIGURE 27 – Tâche pour les Cartes d'attention

Tâches	**Priorités**	**Complexités**
Pouvoir utiliser toutes les images du répertoire target	Moyenne	Moyenne
Afficher les images une à une	Haute	Faible
Afficher les super-pixels sur les images	Faible	Moyenne
Utiliser le clique sur les super-pixels	Haute	Haute
Visualiser le super pixels qui a été sélectionné	Haute	Faible
Sauvegarder la zone sélectionnée	Haute	Moyenne
Sauvegarder le masque obtenu	Haute	Moyenne
Ajouter un bouton "Save" pour sauvegarder le masque	Moyenne	Faible
Appliquer le masque obtenu à une image	Moyenne	Faible
Sauvegarder l'image masquée avec le bouton "Save"	Moyenne	Moyenne
Ajouter un bouton "Next", supprimer le bouton "Save"	Moyenne	Faible
Passer à la sélection suivante avec le bouton "Next"	Haute	Moyenne
Ajouter les fonctionnalités de "Save" à celle de "Next"	Moyenne	Faible
Ajouter un bouton "All"	Moyenne	Faible
Sélectionner entièrement les images avec le bouton "All"	Moyenne	Faible
Mettre les images au bon format pour le réseau	Haute	Haute
Visualiser tous super pixels qui ont été sélectionnés	Haute	Haute
Permettre la dé-sélection d'un super-pixel	Faible	Moyenne
Régler l'affichage pour la sélection de la zone en couleur	Moyenne	Haute
Créer deux dossiers de masques pour les objets d'image	Moyenne	Faible
Inclure les masques des images	Haute	Faible
Utiliser les nouveaux masques pour sélectionner un objet	Haute	Moyenne
Permettre la sélection d'un objet en un clic	Haute	Haute
Réparer les trous dans la sélection de l'image à coloriser	Haute	Haute
Créer une liste d'identifiant de super-pixel sélectionnés pour chacunes des 4 segmentations slic	Haute	Faible
Permettre la sélection du fond de l'image	Faible	Moyenne
Contour jaune lors de la sélection complète	Faible	Faible
Permettre la colorisation automatique à chaque fois	Haute	Moyenne
Faire la fusion des deux colorisations	Haute	Haute

FIGURE 28 – Tâche pour la Carte de segmentation pour les images cibles et références

9.4 Diagramme

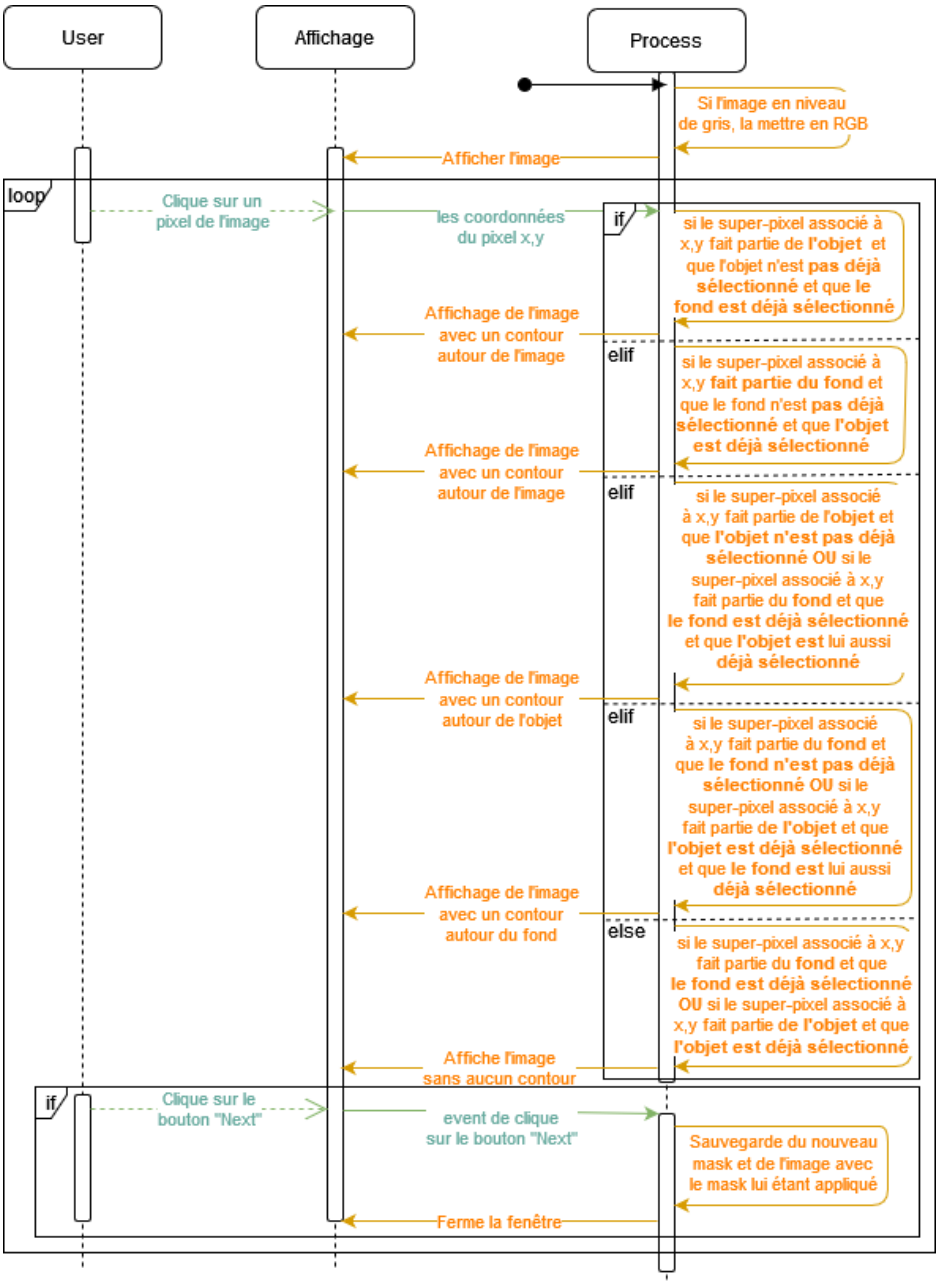


FIGURE 29 – Diagramme de séquence