

Projet Introduction à l'imagerie numérique: Transfert de Style

Thomas Eboli

January 17, 2017

Abstract

Ce projet porte sur la problématique de comprendre, implémenter et observer la technique de transfert de style d'une image (souvent un tableau) vers une photographie comme décrit dans [1]. Le code utilisé pour ces expériences est disponible à <https://github.com/TheFeanor/Transfert-style>.

La partie 1 portera sur l'écriture sous forme de problème d'optimisation du problème de transfert de style. La partie 2 portera sur l'intérêt des paramètres utilisés pour construire la fonction de perte à minimiser. La partie 3 portera sur les réponses du réseau de neurones à différents styles de peinture principalement. Enfin, la partie 4 portera sur l'étude des réponses de différentes couches du réseau.

1 Modélisation sous forme d'un problème d'optimisation

Dans cette partie, je vais décrire le fonctionnement de la procédure utilisée pour générer des images par cette technique et son écriture sous forme d'un problème d'optimisation quadratique mais non-convexe.

Le réseau

Pour réaliser le transfert de style, on ne peut pas faire simplement la différence des deux images ou n'importe quelle opération, même la plus farfelue qui soit car l'espace des images telles qu'ont les perçoit n'est tout simplement pas adapté pour ce type de procédé. En effet, les images vivent dans des espaces de dimension un million par exemple et une certaine famille d'image vit sur une variété de cet ensemble et la distance euclidienne ne signifie rien sur de tels espaces. Deux choix s'offrent à nous (i) travailler en très grande dimension ou (ii) réduire la dimension du problème en conservant l'information intrinsèque de chaque image. Il faut donc trouver une nouvelle représentation dans laquelle il est facile de quantifier des "grandeur" telles que le style, la régularité des structures, etc. C'est donc un changement de variable Φ qui sort naturellement de cette réflexion. Le changement de variable doit être invariant au groupe des déformations car on souhaite conserver des valeurs similaires pour des objets du même type (dans une image naturelle, deux humains doivent avoir des réponses proches, deux éléphants doivent avoir des réponses similaires, etc) comme on le verra dans la partie 4 et comme expliqué plus en détails dans [4]. Il se trouve que l'architecture des réseaux profonds vérifie une telle propriété. Dans le domaine des images, les réseaux de neurones convolutionnels (ou CNNs) atteignent des résultats ahurissants et dépassent toutes les méthodes développées jusqu'alors comme les SIFT ou les HOG. C'est donc tout naturellement que nous allons considérer un tel réseau pour construire la fonction Φ .

Le réseau utilisé est VGG-very-deep-19 dont l'architecture est décrite dans [5]. Dans [3], les auteurs expliquent comment utiliser un tel réseau pour inverser des images à partir de chacune des couches. Leurs résultats ont montré que les premières couches retransmettent très bien l'image de départ car l'effet des non-linéarités n'est pas important voire nul pour conv1-1 et conv1-2. Plus on avance dans le réseau et plus les images deviennent "floues". Le réseau détruit l'image pour notre perception mais il apprend à reconnaître un patron classique pour la catégorie figurant dans l'image ce qui est pratique pour faire de la classification faisant apparaître les idées de distance intra-classe et inter-classe.

En pratique, le VGG-19 est utilisé principalement avec des couches de max pooling mais dans notre cas, les auteurs de [1] préconisent d'utiliser des average pooling layers car les résultats sont meilleurs. Un comparatif des deux méthodes est détaillé dans la seconde partie.

La méthode

La méthode utilisée est un problème d'optimisation où il faut que l'on trouve une image \hat{x} solution du problème d'optimisation :

$$\min_x \alpha L_{contenu}(x, p) + \beta L_{style}(x, a) \quad (1)$$

La variable p désigne la photographie choisie et la variable a l'image dont on veut extraire le style. Il est à noter que la fonction objectif est non-convexe et que le résultat est donc unique pour chaque itération. On tombe donc un minimum local de la fonction mais ce n'est pas grave car le problème est celui de la génération d'image; ça nous arrange même d'avoir quelque chose de nouveau à chaque fois à vrai dire !

On optimise par descente de gradient cette fonction objectif pour trouver la solution \hat{x} .

Perte liée au contenu

Le terme qui prend en compte le contenu de la fonction se calcule par la forme :

$$L_{contenu}(x, p) = \frac{1}{2} \sum_{i,j} (F_{i,j}^k - P_{i,j}^k)^2 \quad (2)$$

Où, si on note N_k le nombre de filtres pour la couche k et M_k le produit hauteur \times largeur d'un de ces filtres, alors $F_{i,j}^k$ est l'activation du i -ème filtre à la position j dans la forme vectorisée des filtres pour la couche k et pour l'image recherchée x . $P_{i,j}^k$ est la même activation mais pour la photo p .

Pour l'algorithme de rétro-propagation du gradient, on a besoin du gradient de cette perte. On obtient alors le gradient de cette perte par la formule :

$$\frac{\partial L_{contenu}}{\partial F_{i,j}^k} = (F^k - P^k)_{i,j} \times \text{ind}_{F_{i,j}^k > 0} \quad (3)$$

Perte liée au style

La perte liée au style est plus compliquée à calculer mais rien de bien horrible non plus. Contrairement à la perte liée au terme de structure et en accord avec les observations sur l'effet du VGG-19 à chaque couche, on prend l'information sur le style sur les couches conv1-1, conv2-1, conv3-1, conv4-1 et conv5-1. Chaque couche possède un niveau d'abstraction différent et donc une information différente sur le style de l'image. On considère d'abord les matrices de Gram qui mesurent les corrélations entre les divers canaux (on considère les versions vectorisées des activations !):

$$G_{i,j}^k = \sum_n F_{i,n}^k F_{j,n}^k \quad (4)$$

Si on note $G_{i,j}^k$ la matrice de corrélation associée aux activations des canaux i et j pour la couche k et l'image à chercher x et $A_{i,j}^k$ l'équivalent pour l'image dont on extrait le style, alors l'erreur pour la couche k s'écrit :

$$E^k = \frac{1}{4N_k^2 M_k^2} \sum_{i,j} (G_{i,j}^k - A_{i,j}^k)^2 \quad (5)$$

On pondère chacune de ces pertes avec les poids w_k . La perte totale s'écrit :

$$L_{style}(x, a) = \sum_k w_k E^k \quad (6)$$

Comme pour l'erreur sur le contenu, on cherche à rétro-propager l'erreur sur le style. Le gradient s'écrit:

$$\frac{\partial L_{style}}{\partial F_{i,j}^k} = \frac{1}{N_k^2 M_k^2} (F^k)^T (G^k - A^k)_{i,j} \times \text{ind}_{F_{i,j}^k > 0} \quad (7)$$

2 Jouons avec les divers paramètres

On a tous les outils pour maintenant effectuer l'opération de transfert de style. Dans toute la suite, si on ne précise pas un paramètre, c'est qu'il prend sa valeur par défaut qui sont les suivantes :

1. poids $w_k = 0.2 \forall k$
2. couche de contenu : conv4-2
3. nombre d'itération $N = 500$
4. $\alpha = 1, \beta = 10^4$
5. pooling : average

Pour les mesures, j'utilise les images suivantes:



Figure 1: Droite: p , Vue de Paris - Gauche: a , "Nuit étoilée" de Van Gogh

Faire varier les couches de style

Je décide ici de faire varier les poids associés aux différentes couches servant à la représentation en terme de style de l'image à trouver x et de l'image dont on extrait le style a .

Ne choisir qu'une couche J'ai essayé d'obtenir des images avec une seule couche de style mais la phase d'optimisation n'aboutit jamais car le gradient tombe très vite dans un minimum local. On ne peut donc tirer un quelconque résultat pertinent de cette partie.

Donner plus d'importance à certaines couches J'essaie avec des poids reflétant divers configurations où on donne plus de poids aux premières couches puis aux dernières couches. J'utilise tour à tour les poids :

1. $w_1 = 0.6, w_2 = 0.1, w_3 = 0.1, w_4 = 0.1, w_5 = 0.1$
2. $w_1 = 0.35, w_2 = 0.25, w_3 = 0.2, w_4 = 0.15, w_5 = 0.05$
3. $w_1 = 0.1, w_2 = 0.1, w_3 = 0.6, w_4 = 0.1, w_5 = 0.1$
4. $w_1 = 0.05, w_2 = 0.15, w_3 = 0.2, w_4 = 0.25, w_5 = 0.35$
5. $w_1 = 0.1, w_2 = 0.1, w_3 = 0.1, w_4 = 0.1, w_5 = 0.6$



Figure 2: Divers essais pour les configurations allant de 1 à 5 de gauche à droite

On remarque que globalement modifier les poids n'amène pas à de grands changements mais lorsqu'on regarde de plus près, on peut apercevoir des étoiles en plus ou en moins suivant les configurations (par exemple au sommet de la tour Eiffel). Les immeubles de la Défense ou en premier plan sont légèrement différents d'une image à l'autre aussi (il vaut mieux regarder ces images sur un écran plutôt que sur une version imprimée). Chaque couche possède donc sa spécialité en quelques sortes même si globalement, le résultat possède un rendu identique dans chaque cas.

Faire varier la couche de contenu

Je fais varier les couches de style en choisissant tour à tour conv1-2, conv2-2, conv3-2, conv4-2 et conv5-2. La figure 3 expose les résultats obtenus.



Figure 3: De gauche à droite, des images allant de conv1-2 à conv5-2 pour le choix de la couche de structure

Pour des choix de couches de style basses comme conv1-2 et conv2-2, la méthode de descente de gradient mise en place n'aboutit pas et on obtient seulement des débuts de procédure de transfert de style. Pour conv1-2 on peut quand même voir que l'image commence à tourner autour des points définis comme des emplacements de soleil et les rues symétriques commencent à se tordre. Pour conv2-2, l'algorithme du gradient va plus loin et on voit apparaître plus nettement l'effet du transfert mais on conserve encore très nettement le style de la photographie de Paris. Pour les essais sur les couches suivantes, les non-linéarités du réseau commencent à se faire sentir. On commence à franchement perdre la structure au profit du style. L'algorithme converge enfin. Plus on choisit une

couche élevée pour la la structure et plus l'image se met à tourner, plus on perd le détail des immeubles et même de la tour Eiffel. Il est amusant de noter que l'avenue menant au Champs de Mars est comprise comme un immeuble et la tour visible sur "Nuit Étoilée" est transposée sur cette avenue. Dans la dernière image, on ne conserve tellement plus d'information sur le contenu que la tour Eiffel est presque effacée !

Faire varier le nombre d'itérations

Je fais varier le nombre d'itérations N en choisissant tour à tour 5, 50, 100, 250 et 500. Les résultats se trouvent dans la figure 4.



Figure 4: De gauche à droite, des images allant de 5 à 500 pour le choix de N

Phénomène déjà visible dans la section précédent, le nombre d'itérations est primordial pour permettre un transfert efficace du style d'une image vers une autre. Ainsi Pour de petits nombres d'itérations, la photographie s'imprègne que très peu du style. On voit apparaître des formes psychédéliques dans le ciel puis petit à petit les étoiles et la couleur. Les bâtiments se tordent au fur et à mesure que les étapes se succèdent pour finalement produire un résultat probant autour de 500 itérations (convergence).

Faire varier α et β

Je touche ici au cœur du problème d'optimisation en modifiant aux poids α et β et plus particulièrement à β car optimiser $\alpha L_{contenu} + \beta L_{style}$ est équivalent à minimiser $L_{contenu} + \frac{\beta}{\alpha} L_{style}$. Donc en fixant $\alpha = 1$, je n'ai plus qu'à jouer sur β .

Je fais varier le ratio $\frac{\beta}{\alpha}$ en choisissant tour à tour $\beta = 1, 10, 100, 1000, 10000$ et 100000 . Les résultats sont visibles à la figure 5



Figure 5: De gauche à droite, des images avec un choix de β allant de 1 à 100000

On observe que pour des valeurs de β relativement faibles et comparables à α , le style n'est pas transféré ou très peu vers la photographie de Paris. Pour des valeurs plus fortes (à partir de 10000), l'opération se fait et pour la valeur maximale, on transfère tellement de style qu'on perd énormément de détails. Par exemple le quartier de la Défense est totalement noyé, les immeubles en premier plan aussi et la tour Eiffel disparaît à nouveau. Mettre beaucoup de style semble équivalent et ne pas considérer beaucoup de structure comme dans le cas du choix de conv5-2 pour la couche de structure. On retombe bien sur nos pieds.

Faire varier le type de pooling

Dans [1], les auteurs affirment qu'ils ont de meilleurs résultats avec un couche average pooling plutôt qu'avec les couches de max pooling classiques ou de stochastic pooling comme décrit dans [6]. Ils ne fournissent pas d'illustrations de ce propos donc je propose de fournir un exemple pour avoir le cœur net à la figure 6.

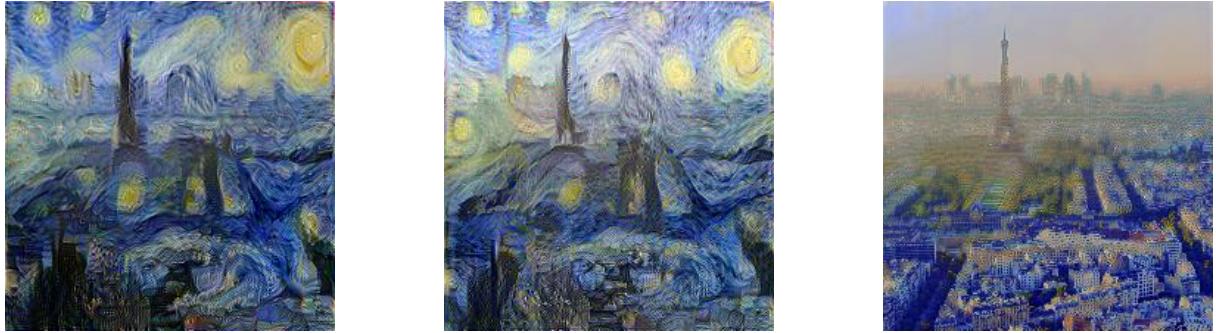


Figure 6: A gauche, le average pooling, au milieu le max pooling et à droite le stochastic pooling

Les auteurs de [1] choisissent le average pooling car il permet de mieux rétro-propager le gradient et ça se voit ! Les deux autres techniques détruisent beaucoup d'informations à chaque étape en ne gardant que certains pixels de chaque patch alors que la méthode par moyenne conserve un peu d'information de chacun de pixels. Avec de petits filtres comme ceux du VGG-19, on peut même espérer en tombant sur des zones uniformes de reconstruire l'information très fidèlement car la moyenne de la zone uniforme est très proche de la valeur de chacun des pixels de cette zone.

3 Jouons avec les styles artistiques

On a vu que la méthode de transfert de style fonctionnait bien avec les bons paramètres pour "Nuit Étoilée" de Van Gogh mais que se passe-t-il maintenant si on change le style du tableau ? Chaque mouvement en peinture se définit par le style que partagent les artistes d'une même périodes et certains sont plus fidèles à la réalité que d'autres et certaines œuvres font plus apparaître des lignes et des structures que d'autres (ce qui n'est pas le cas des œuvres de Van Gogh !). Un cas d'école est d'opposer Ingres et Delacroix, c'est la ligne contre la couleur. Je propose donc d'exhiber ces différences en le faisant non pas sur un paysage mais une photographie de moi-même pour des raisons de droit à l'apparition et pour ne pas ressortir la sempiternelle image de Lenna, on l'a assez vue. Un visage possède plusieurs zones uniformes et pleins de détails donc on peut tester les capacités de la méthode sur plusieurs configurations de pixels. Pour ce qui est des tableaux, je pioche dans les paysages et les portraits pour voir si la structure du tableau influe un peu, beaucoup ou pas du tout dans la qualité du transfert. La photographie de départ est la figure 7 et les résultats sont les figures 8, 9, 10, 11, 12 et 13.

Nuit Étoilée - fig 8 Les structures sont respectées et là où il y a du jaune dans l'image initiale il y a des étoiles. Le visage étant une zone uniforme, un même ton est utilisé (celui de l'espèce de nuage semble-t-il). Le second visage possède une couleur plus proche de la chair montrant que le réseau considère le visage du badaud et le mien comme deux choses différents. La présence d'un arbre puis d'un fond est elle aussi respectée car une étoile est estompée par le tronc. L'aspect des maisons se retrouve dans la modélisation de la foule au fond ainsi que dans les yeux et les dents. qui sont des zones particulières qui tranchent avec l'uniformité de la peau.

Impression Soleil Levant - fig 8 Comme précédemment, ce qui frappe, c'est que l'algorithme cherche à associer des zones de même couleurs. La chair est proche de la couleur du ciel du tableau donc la peau ressemble au ciel avec quelques zones d'eau. Le rouge est modélisé par des ersatz de soleils et les cheveux prennent la couleur du port de Londres dans la brume. Le tableau ne possédant pas de structures comme les maisons dans l'œuvre précédente, les zones de détail comme la foule, les yeux et les dents sont moins bien rendus.



Figure 7: Une photo pendant l'arrivée du Tour de France à Paris

J'ai choisi cette image pour les raisons mentionnées plus haut mais aussi car on peut apercevoir un visage au second plan et un arbre qui est une structure droite qui possède quelques détails puis en arrière-plan une foule et quelques structures complexes comme les stores. Il y a matière à voir différents effets sur cette image donc.



Figure 8: Gauche: Nuit étoilée de Van Gogh - Droite: Impression Soleil Levant de Monet

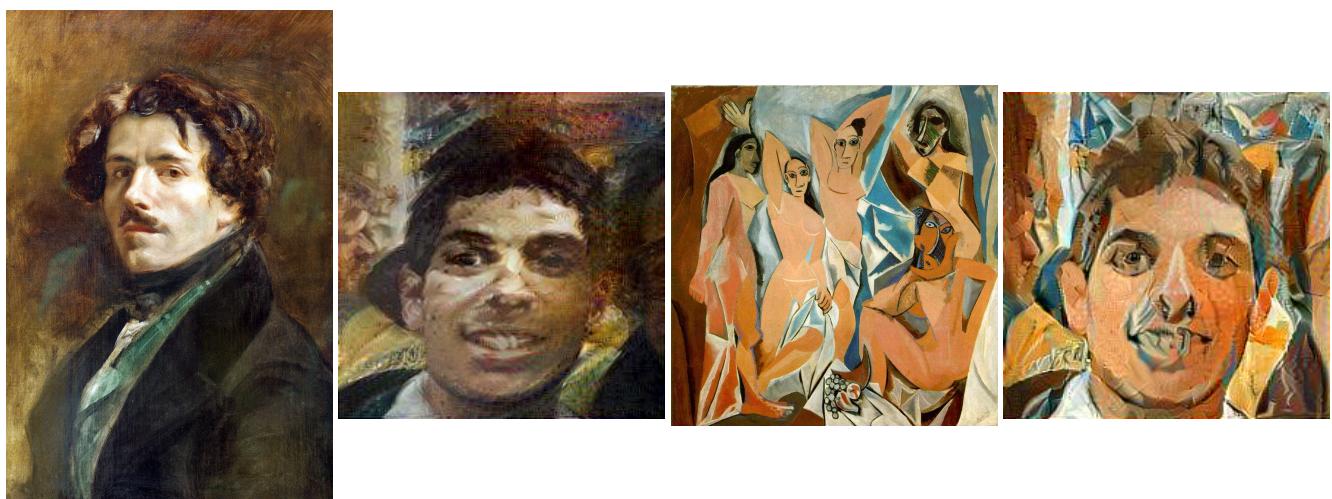


Figure 9: Gauche: Autoportrait au gilet vert de Delacroix - Droite: Les Demoiselles d'Avignon de Picasso

Autoportrait au gilet vert - fig 9 L'intérêt ici est qu'on transpose un portrait sur un visage donc on s'attend à ce que le visage ressorte mieux. C'est majoritairement le cas. L'algorithme se permet même de transposer avec

quelques bémols le jeu de lumière dans les creux du visage de Delacroix vers mon visage. Mon pull bleu devient même verdâtre et mon manteau bleu prend la couleur de la redingote du peintre. Le visage en second plan est lui pour le coup très réussi car moins contrasté. Par contre la foule au fond est très mal rendue car le tableau ne possède pas de fond à part un ensemble marron teinté de rouge et de bleu qu'on voit apparaître dans le rendu de la foule. Il est à noter que l'œil droit est remarquablement bien rendu alors que le gauche moins (question d'illumination à mon avis). Le seul blanc sur le tableau provient de la chemise du peintre donc le rendu pour une zone spécifique comme les dents n'est pas très bon.

Les Demoiselles d'Avignon - fig 9 Picasso peint la chair et l'algorithme le comprend tout à fait et transpose à merveille les différentes teintes de peau sur le visage en respectant l'aspect cubique des personnages. Le nez prend aussi l'aspect deux points de vus en un. Les yeux sont bien rendus car il y a cinq exemples dans le tableau. L'algorithme comprend qu'il y a des personnages et un fond et décide donc de recréer ce fond en arrière-plan ce qui produit cet effet cohérent à l'arrière tout en respectant le fait qu'il y a un arbre qui n'appartient pas au même plan et qui est donc d'une couleur différente. Le personnage en second plan est mélangé avec l'arbre car ses cheveux peuvent déjà se "confondre" avec le tronc. Les dents posent encore problème avec un léger flou autour de la lèvre inférieure. Ce qui est étrange car c'est clairement dans le cubisme qu'on s'attend à trouver des formes pareilles !

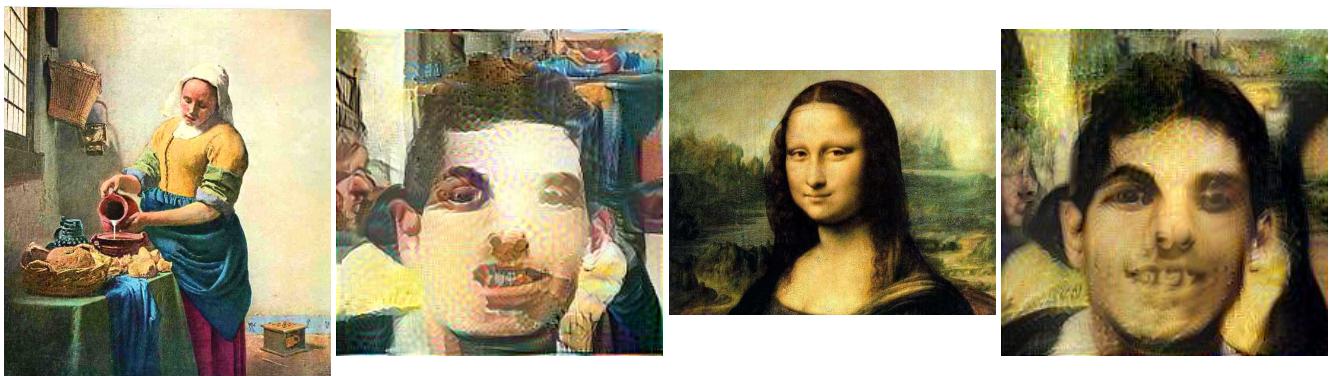


Figure 10: Gauche: La laitière de Vermeer - Droite: La Joconde de De Vinci

La Laitière - fig 10 Gros fiasco pour la Laitière de Vermeer en ce qui me concerne ! Le rendu du visage est très mauvais car la zone uniforme de la chair est prise pour le mur en arrière-plan. Par contre, très belle réussite pour la foule en arrière-plan qui est prise pour une table ! Le bout de tête de mon ami est fondu avec une nappe et on aperçoit un panier avec des fruits et des petits pots bleus carrés. Le réseau a donc appris le style d'une table façon Vermeer et reproduit la seule zone de détails là où il y en a. Les dents et les yeux font aussi apparaître des paniers ou des aliments ! enfin le tronc devient un mur sur lequel y a un objet qui ressemble un peu à la fenêtre qu'on peut voir sur le tableau. Au-delà de reproduire le style du tableau, le réseau a recréé un décors proche de celui du tableau et ça, ce n'était pas gagné.

La Joconde - fig 10 Le résultat est à nouveau mitigé. Puisqu'on voit le visage de Monna Lisa, les yeux sont modifiés selon (mon regard ne vous suit pas comme celui de Monna). Le nez est par contre grotesquement élargis et le détail des dents est perdu à nouveau. L'arrière-plan possédant du contenu varié, l'algorithme choisit d'appliquer les teintes verdâtres des arbres plutôt que le ciel qui est choisi pour le tronc et les objets jaunes. Il est à noter aussi que les cheveux font apparaître des mèches sur le front et la tempe droite du visage à l'image des mèches de la Joconde. Encore une fois l'algorithme apprend le paternité des cheveux et l'applique à la photographie.

Été - fig 11 Pour ce tableau d'Arcimboldo, l'aspect des fruits et légumes ne ressort pas sur mon visage qui est plutôt compris comme le manteau de blé du personnage sauf dans les zones de détails comme le nez. Le visage de mon ami et celui en second plan par contre possèdent chacun une texture plus proche de celle des fruits et des



Figure 11: Gauche: Été d'Arcimboldo - Droite: Vitrail

fleurs. Je pense que celui de mon ami fait apparaître des pétales car il est sur le bord comme là où se situent les plantes. L'arbre est complètement flouté et la foule en arrière-plan laisse difficilement percevoir des fruits et légumes de bonne résolution même sur les zones rouges ressemblent à des tomates ou des cerises.

Vitrail - fig 11 Voici un exemple présentant des lignes très bien définies et un découpage en rectangles. Le réseau interprète les changements de contraste comme des bordures de vitrail ce qui laisse apparaître ces bandes noires. De plus, il s'efforce à rendre la géométrie en rectangles et le fond bleu sur la rue. L'arbre est compris comme un pilier, le badaud en second plan est perdu la foule aussi. Les zones uniformes du visages produisent un effet patchwork car il n'y pas de grandes zones uniformes sur le vitrail. Les structures carrées comme les dents sont bien découpées mais la couleur est mauvaise et les yeux donnent aussi cet effet patchwork car la résolution des yeux de saint-Louis n'est pas assez bonne pour transposer les yeux comme dans le cas de Delacroix. Il est amusant de noter que comme pour Arcimboldo, le cadre doré est présent autour de l'image.



Figure 12: Gauche: L'empereur Charlemagne - Droite: Morceau de papyrus

Charlemagne - fig 12 Comme dans le cas du vitrail, cette scène médiévale est représentée dans une image à l'aspect très géométrique et structuré. On ne voit pas bien les visages des protagonistes et le rouge est la couleur dominante avec quelques pointes de bleu. Le transfert se fait assez mal du coup et le rendu est très mauvais avec beaucoup de rouge pour représenter la chair, une autre teinte de rouge pour les cheveux bruns et encore un rouge tendant vers l'ocre en fond. L'arbre est blanc-gris car c'est la couleur la plus proche présente dans la scène. Je pense



Figure 13: *Vue de Paris*

aussi que la position et la forme de l'autel joue beaucoup car l'autel peut être assimilé à l'arbre. C'est d'ailleurs pour cette raison qu'on perd le visage du badaud à mon avis. Les découpage en fonction du contraste et des zones détaillées comme les yeux et les dents sont bien respectées par contre grâce à la structure en maillage de l'image médiévale.

Papyrus - fig 12 On contenu dans les images à forte structure avec ce morceau de papyrus qui est un ensemble de colonnes avec des symboles. Comme dans [2], le réseau apprend à créer des symboles (ici des hiéroglyphes égyptiens), certes flous mais cohérents avec l'image de départ. De plus, on retrouve l'aspect colonne du parchemin mais les lignes suivent les contour de mon visage et du bout de visage de mon ami en donnant cet effet "cell-shading". Les zones détaillées laissent encore une fois apparaître des contours et des détails plus marqués grâce à la structure très géométrique de l'image de départ.

Photo de Paris - fig 13 Pour clore cette série de transferts, je me suis demandé ce que pouvait produire le transfert de style d'une image naturelle vers une image naturelle; le résultat est très mauvais. La structure de l'image de départ est très bien reconnaissable car le style est le même ! On remarque quand même que ce qui marque particulièrement le réseau aussi, ce sont les fenêtres des immeubles en premier plan qui créent cet aspect mosaïque sur ma photo ! Mon visage étant une zone presque uniforme, le ciel y est transposé et mes yeux étant des zones de détails, on peut y voir des bouts de tour Eiffel.

Commentaires

On remarque que le transfert de style ne se fait pas n'importe comment: les zones de mêmes couleurs et formes sont favorisées. Les géométries de l'image de style se retrouvent dans l'image générées ou alors s'adaptent à la structure de l'image de départ. Dans le cas échéant, le réseau applique le style du tableau et génère de l'information pour combler les différences comme pour l'Été d'Arcimboldo ou la Laitière de Vermeer ce qui est véritablement magique ! Ou pas.

On peut comprendre ces divers phénomènes en revenant à la structure même du réseau. Chaque couche applique des filtres capturant des informations différentes. Les filtres eux-mêmes possèdent des aspects différents comme le montrent les auteurs de [7]. Les premières couches prennent l'information sur les structures simples de l'image et ressemblent à des filtres DCT ou des ondelettes. Les couches plus hautes possèdent des filtres bizarres qui captent donc des informations plus en rapport avec les caractéristiques propres d'un objet. Le réseau produit donc des vecteurs de features dont certaines dimensions correspondent aux zones de basse-fréquences et d'autres aux zones de hautes fréquences et met en relation ces différentes features dimension par dimension lors du calcul de la perte. Je pense que si les informations dans les deux vecteurs dans une dimension sont proches (couleur, géométrie, etc), alors la descente de gradient trouve un compromis entre les deux images qui conserve beaucoup d'information de l'image de départ et modifie proportionnellement au ratio $\frac{\beta}{\alpha}$ l'information de départ. Dans le cas contraire, en fonction du même ratio, l'image de style écrase l'information de départ comme dans le cas de la Laitière en figure

10 où la foule étant un détail très différent des détails du tableau, le réseau préfère écraser la foule et mettre à la place une table avec des paniers.

4 Etude des différentes couches

Dans cette partie, je propose de nous attarder plus en détail sur les filtres utilisés à chaque couche ainsi que d'étudier quelques réponses des activations dans chaque couche d'intérêt pour la combinaison photo de Paris et Nuit étoilée, comme dans la partie 2 de ce rapport. Un exemple de figures obtenues se trouve dans les divers vidéos fournies avec ce rapport (et aussi présentent dans le dossier "Vidéos").

Filtres des couches basses: Pour ces couches, les réponses de ces filtres suivent assez fidèlement les contours des éléments des images. Ceci concorde avec l'allure des filtres présentés dans [7] où on retrouve des filtres de Gabor ou de DCT pour conv1-1 et conv2-1. On remarque aussi qu'au fil des itérations, les activations font ressortir des zones différentes. Par exemple, pour la première couche conv1-1, au fil des déformations des rues, les réponses de ces zones deviennent non nulles alors qu'au contraire, le ciel qui est un élément qui ressort particulièrement pour un certain filtre devient de moins en moins actif pour ce neurone et seules les formes obtenues au fil des déformations ressortent. On peut l'expliquer par le fait que dans la photo originale, les structures sont assez régulières donc les contours sont très simples ce qui permet aux filtres de type Gabor ou DCT de produire des représentations très parcimonieuses. Lorsque l'image se déforme, beaucoup de complexité et d'information est apportée. Les pixels d'une même zone peuvent donc ne plus être en correspondance en terme de régularité et les filtres s'affolent et font ressortir beaucoup plus d'information !

Filtres des couches hautes: Pour ces couches, les filtres montrés dans [7] sont beaucoup plus complexes que les simples filtres de Gabor des couches basses. L'effet des compositions d'opérations linéaires puis non-linéaires produisent des fonctions robustes au groupe des déformations (théorie vue dans le cours de Stéphane Mallat). Les activations de ces couches ne se localisent donc plus forcément sur des structures simples comme des bords ou des variations de couleur. Les vidéos des couches 4-1, 4-2 et surtout 5-1 font apparaître des motifs très sensibles aux déformations apportées par les différents itérations de l'algorithme d'optimisation. La couche 4-2 est remarquable dans le sens où on peut encore apercevoir des filtres sensibles aux structures. On peut deviner la position de certaines barres d'immeubles ou de la tour Eiffel suivant les zones où les activations sont non-nulles dans la vidéo même si on a perdu la netteté des premières couches. La couche 5-1 perd totalement cette notion de logique spatiale au sens du positionnement de la géométrie de l'image de Paris et gagne en abstraction. Les filtres de cette couche étant très nombreux (512 filtres 3×3 d'après [5]), ils peuvent capturer une information très précise dans l'image. On remarque que certains filtres répondent de plus en plus fortement là où se trouvent les étoiles dans l'image générée et d'autres répondent fortement là où la tour est générée en bas à gauche de l'image. On peut imaginer que ces filtres capturent le concept d'une rue et substitue le concept de la tour de "Nuit étoilé" car les réponses de ce même filtre pour la photographie et l'œuvre d'art peuvent être semblables. Effectivement, conv5-1 est proche des couches "fully-connected" donc chaque filtre commence à faire apparaître l'idée de dimension dans les vecteurs de features des couches fc. Le raisonnement fait dans le commentaire des observations de la partie précédente prend sens ici. Le réseau, dans le but de minimiser la perte, va associer des features proches entre-elles et par rétro propagation du gradient, cette idée de proximité de l'information contenue dans les features apparaît dans les réponses des filtres de couches hautes.

Conclusion

Pour conclure, on a vu que le problème de transfert de style n'est qu'un problème d'optimisation sans contrainte. Au vu de la qualité des images dans la partie 3, on pourrait s'amuser à trouver des formes de régularisation pour améliorer le rendu de la génération et effacer ainsi certains artefacts (flou autour de la bouche pour le Picasso de la figure 9 par exemple). On pourrait se baser par exemple sur le modèle de [3].

La structure du réseau est au cœur de la qualité et de la quantité des éléments qui transitent d'une image vers une

photographie et le fait que le calcul des distances inter-classes et intra-classes pour la classification se fassent de cette manière avec ces filtres en particulier permet donc de jouer sur le style et la structure et d'obtenir ainsi des vecteurs de features avec lesquels on peut jouer pour générer de nouvelles images. L'algorithme de rétro-propagation du gradient est l'autre clef de voûte de la solution au problème de transfert de style.

Ce rapport a donc permis tout au long de mettre en évidence les intérêts des différents paramètres dans le problème d'optimisation proposé dans la partie 2 puis le rôle joué par la génération de features dans la partie 3 et enfin comprendre le fonctionnement du réseau de neurone sur l'image à différents niveaux dans la partie 4, chose que l'on fait rarement !

Sources des images originales

1. Photo de Paris: www.tourmontparnasse56.com
2. Papyrus: www.isere.gouv.fr
3. Charlemagne: www.historyweb.fr
4. Vitrail de saint Louis: www.catholictradition.org/

References

- [1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.
- [2] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. *CoRR*, abs/1505.07376, 2015.
- [3] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. *CoRR*, abs/1412.0035, 2014.
- [4] S. Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society of London Series A*, 374:20150203, April 2016.
- [5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [6] Matthew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *CoRR*, abs/1301.3557, 2013.
- [7] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

Appendice A: Ce que j'ai fait dans ce projet

Dans ce projet, j'ai tout d'abord fait un certain travail de documentation dans le domaine de l'apprentissage profond et de l'application des CNNs au traitement des images et à la vision par ordinateur. Une fois ce travail théorique effectué, j'ai codé l'algorithme, tout d'abord avec Tensorflow puis avec Caffe car ce framework convenait plus à mon sens au problème à résoudre. Je me suis toutefois inspiré grandement d'une implémentation similaire pour la fonction d'optimisation, que ce soit dans l'implémentation de *scipy.optimize.minimize* ou dans l'implémentation des bornes des contraintes que j'ai naturellement mises entre 0 et 255 alors qu'on soustrait une moyenne par canal au début !

La suite de mon travail a donc été de jouer sur les différents paramètre de cet algorithme et les effets induits par différents styles et différentes peintures. Enfin j'ai cherché à comprendre concrètement le propos de Stéphane Mallat dans [4] en visualisant les réponses de différentes couches lors de la phase d'optimisation.