

# FloralNet – Un modèle de génération d’images photoréalistes à partir de dessins de fleurs

Hugo PEYRON

hugo.peyron@student-cs.fr

Stéphane MARTIN-RICHTER

stephane.martin-richter@student-cs.fr

Eva FEILLET

eva.feillet@student.ecp.fr

**Abstract**—Dans ce projet, nous proposons une méthode permettant de transformer des dessins en des images photoréalistes grâce à un modèle génératif. Nous avons appliqué cette idée à un jeu de données de photos de fleurs, photos que nous avons tout d’abord transformées pour leur donner un style proche du dessin. Ces dessins de fleurs en noir et blanc servent de patron au réseau de neurones pour fournir des images recolorées et aussi nettes que des photographies. Au delà de l’aspect ludique, ce processus a de nombreuses applications, des beaux arts à l’identification de suspects grâce à des portraits robots, en passant par l’augmentation de données. Le cas d’usage que nous avons exploré est celui d’un assistant graphique pour le dessin, avec comme prolongement à notre modèle la possibilité d’appliquer un transfert de style.

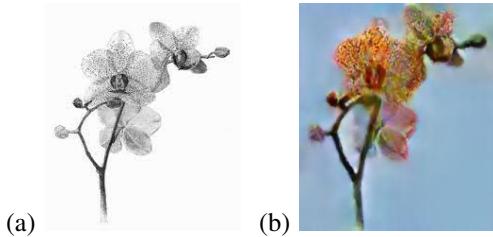


Fig. 1. Exemple de résultat obtenu avec FloralNet : (a) orchidée dessinée au crayon, donnée en entrée à FloralNet (b) Image photo-réaliste proposée par FloralNet

## I. INTRODUCTION

L’intelligence artificielle a fait une entrée retentissante dans le monde de l’Art avec Le Portrait d’Edmond de Belamy, une impression sur toile produite par un GAN. Première œuvre d’art à être présentée dans une salle des ventes, ce portrait d’un personnage fictif a été vendu 432 500 dollars chez Christie’s en 2018 [1]. Depuis, le robot Sofia, créé par Hanson Robotics, a vendu son autoportrait - réalisé par le robot manuellement en interprétant un portrait fait d’elle par un artiste humain – près de 700 000 dollars. [2]. Du côté du grand public, transformer des photos en des dessins est une fonctionnalité déjà bien rodée de logiciels comme Photoshop, avec des techniques reposant sur des transformations au niveau des pixels. Aujourd’hui, avec le développement du deep learning, et l’engouement du grand public pour des applications visuelles comme le transfert de style, des entreprises proposent de transformer des photographies en dessins, tableaux ou autres imitations, permettant ainsi aux particuliers d’acquérir des œuvres d’art personnalisées [3]. Photoshop s’est ainsi mis à la page avec une option « neural

filters » proposant des modèles neuronaux pour lisser le grain de la peau et faire du transfert de style sur des photographies, et même, actuellement en bêta, pour accentuer une émotion sur un visage, modifier la direction du regard, ou coloriser des photographies argentiques [4]. Dans le cadre de ce projet, nous avons pris le contre-pied de cette tendance en partant d’un dessin plutôt que d’une photographie. Notre réseau de neurone FloralNet, entraîné sur des dessins de fleurs simulés, est capable de donner chair à un dessin noir et blanc au crayon.

Notre travail s’organisera de la manière suivante. Après avoir présenté divers travaux en lien avec notre sujet de recherche, nous expliciterons plus en détail l’approche adoptée. Nous évoquerons par la suite les divers choix que nous avons pris en termes de modèles, d’optimisation d’hyperparamètres et de méthodes d’analyse de performance pour pouvoir mener à bien notre projet. Cela nous permettra de discuter ensuite des résultats que nous avons obtenus en suivant notre approche. Enfin, nous conclurons sur les résultats obtenus et les approches complémentaires que nous souhaiterions entreprendre pour donner suite à ce travail.

## II. ÉTAT DE L’ART

### A. Mélanges de styles

Les approches récentes de génération d’images peuvent s’appuyer d’une part sur des réseaux de neurones profonds avec une architecture convolutive, d’autre part sur des réseaux adversaires comme on le détaillera par la suite. Une approche convolutive a été proposée par Gatys et al. dans le cadre de transfert de texture [5] et utilisait un réseau VGG-19 pré-entraîné sur ImageNet. Une des observations faites par les chercheurs était que les représentations du contenu et du style d’une image étaient séparables dans un réseau de neurone convolutionnel. C’est ce qui leur a permis de mixer des images entre elles, conservant le motif de l’une en lui appliquant le style de l’autre. Cette méthode a été prolongée par Li et al. [6] en la combinant à des Markov Random Fields pour synthétiser des images à partir de deux photographies et ce avec un meilleur photoréalisme. Les premières approches se bornaient à appliquer le style d’une image « texture » à une image « motif » et donnaient des résultats assez peu photoréalistes, notamment à cause de bordures floues ou fragmentées et de couleurs incohérentes. Li et al. ont montré que l’utilisation de MRF pouvait donner des résultats plus

plausibles grâce à l'extraction de représentations intermédiaires plus robustes au bruit et permettant une meilleure interpolation.

### B. Sketch Inversion

Notre principale source d'inspiration est l'article Convolutional Sketch Inversion publié par Y. Güçlütürk and U. Güçlü [7], qui présente une méthode de transformation de visages dessinés en des visages de qualité photographique. En partant des datasets publics standards CelebA et Labeled Faces in the Wild, dont ils ont automatiquement transformé les photographies en visages à l'aspect dessiné, ils ont entraîné un réseau de neurones profond basée sur une architecture de transfert de style comme proposé par Johnson et Fei-Fei [8]. Après une série de couches convolutives et de blocs à résidus réduisant progressivement la taille du dessin tout en augmentant le nombre de feature maps, l'image est progressivement déconvoluée jusqu'à retrouver sa taille d'origine, cette fois-ci avec un style photographique. Les chercheurs ont comparé 3 méthodes de conversion de photographie en dessin, et entraîné un modèle pour chaque type de dessin : lignes (contours nets en noir et blanc), nuances de gris (aspect croquis en noir et blanc), couleur (aspect croquis en couleur).

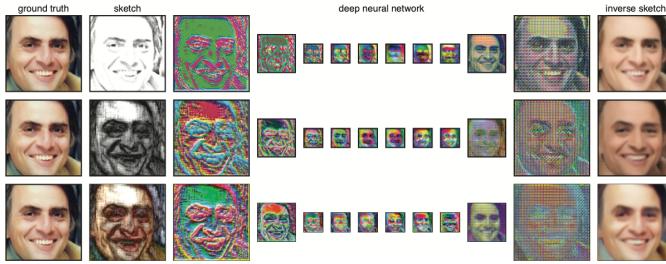


Fig. 2. Chaque ligne correspond à un des 3 styles de dessin. La photographie originale est transformée avec un style « lignes », « croquis noir et blanc » ou « croquis couleur ». Un réseau de neurones est entraîné pour chaque version du prétraitement. On peut observer que le rendu est légèrement différent d'un réseau à un autre.

La fonction de coût utilisée dans ce papier est originale : elle se compose de trois termes permettant de maîtriser différentes qualités visuelles des images finales. Le premier terme est la norme euclidienne classique pixel à pixel (pixel loss), calculée entre la photographie originale et la photographie synthétique. Le deuxième terme est la norme euclidienne entre deux feature maps (feature loss), correspondant l'une à l'image originale, l'autre à l'image de sortie. Cette feature maps correspond à la sortie d'une des premières couches de convolution du réseau de neurones. Enfin le troisième terme est égal au coût total associé à la variation de couleurs dans l'image de sortie (total variation loss) i.e. un visage avec un grain de peau uniforme obtient un coût plus faible qu'un visage avec de fortes variations de couleurs dans des petits voisinages de pixels. Les deux premières composantes ont un poids égal dans la fonction de coût totale, et le troisième terme est pondéré à 10-5. Le constat des chercheurs est que la combinaison du feature loss avec le pixel loss augmente

la qualité de l'image tout en conservant un aspect plaisant, et que le troisième terme permet d'éliminer des artefacts visuels introduits par la feature loss, permettant ainsi un rendu visuel plus lisse.

Les résultats présentés dans cet article sont visuellement impressionnantes mais la portée des modèles est en fait limitée par le prétraitement automatique des images. En effet, lorsque l'on applique le modèle à un dessin réalisé par un humain, si le style de ce dessin s'éloigne de celui obtenu par transformation automatique des photographies par les chercheurs, alors le résultat est moins fidèle à la réalité que ce qu'on observe pour des dessins synthétiques. Une autre équipe de chercheurs s'est intéressée à ce problème en proposant une méthode permettant d'obtenir un visage photoréaliste à partir de seulement quelques coups de crayons [9]. Pour cela, Chen et al. ont mis en place une approche « local-to-global » s'appuyant sur l'apprentissage implicite d'un espace de visages plausibles à partir d'un ensemble de photographies. A partir d'un croquis rudimentaire dessiné par l'utilisateur, ils proposent le visage le plus proche dans cet espace latent. Ils utilisent aussi des features maps intermédiaires pour plusieurs zones du visage, qu'ils combinent pour obtenir une image réaliste de haute qualité.

### C. GAN

Le terme GAN fut abordé pour la première fois en 2014 par Ian Goodfellow dans son papier intitulé : *Generative Adversarial Networks* [10]. Un GAN représente un modèle génératif dans lequel deux réseaux de neurones (le **générateur** et le **discriminateur**) sont placés en compétition l'un contre l'autre. Le but du discriminateur est de déterminer si la donnée  $X$  qu'il reçoit en entrée est réelle, dans le sens où cette dernière provient du jeu de données d'entraînement, ou si elle est fausse car elle est l'œuvre du générateur. Vu la façon dont nous décrivons le discriminateur, nous pouvons nous rendre compte que ce dernier s'apparente à un simple classifieur qui est l'architecture de réseaux de neurones la plus répandue et la plus connue en apprentissage automatique. Contrairement au réseau discriminateur, le générateur qui est un modèle génératif ne prend pas en entrée une donnée<sup>1</sup> mais il en produit une en sortie [11]. Par contre, il est important de mentionner que l'affirmation qui a été faite n'est valable que pour l'architecture de base telle qu'elle a été mentionnée par Ian Goodfellow dans son papier de 2014. En effet, il est possible de passer en entrée du modèle génératif une certaine donnée pour que son apprentissage se fasse sous contrainte. C'est notamment avec ce type d'approche que nous avons entraîné notre réseau.

Pour mieux comprendre la façon dont marche les réseaux antagonistes génératifs, il faut que nous abordons les **auto-encodeurs**. Ces derniers se composent d'un **réseau descriptif** et d'un **réseau génératif**. Le but du premier réseau est de fournir à partir d'une donnée d'entrée, que l'on considérera être une image, un descripteur de cette dernière, c'est à dire une transformation de cette image dans un espace que l'on

1. Par "donnée", nous entendons par exemple une image.

appelle : **espace latent**. Il est important de noter que l'espace latent est de dimension plus faible que l'espace d'où provient l'image. Le but de la projection est donc de ne retenir que les informations/caractéristiques essentielles de l'image et de supprimer tout ce qui est superflu. Par exemple si l'image représente un visage, on va passer d'une donnée de départ très riche : la couleur de chacun des pixels, à une description plus petite mais plus évocatrice : la position des yeux mais aussi leurs couleurs, la position de la bouche ou encore la couleur des cheveux. Au lieu de conserver l'intensité de chacun des pixels composant le visage, on va simplement synthétiser cette information au travers d'un attribut que l'on pourrait nommer : couleur de la peau. Le second réseau qui constitue un auto-encodeur, le réseau génératif donc, prend en entrée la description produite par le réseau descripteur et va produire en sortie une reconstruction de l'image d'origine. Comme notre lecteur l'aura compris, le but est que l'image générée ressemble le plus possible à l'image originale. On voit également que le réseau ne génère donc pas à partir de rien, car il prend en entrée un vecteur Z qui correspond à la projection de l'image dans l'espace latent, autrement dit à la description de ses caractéristiques principales ou *features*. Dans les GAN traditionnels, le générateur synthétise une image en sélectionnant de façon aléatoire un vecteur latent issu d'un espace de faible dimension. Toutefois, comme nous le mentionnons précédemment, il est possible de contrôler la structure de l'image qui est générée. Pour cela, les réseaux adoptent une approche de type auto-encodeur, c'est notamment le cas du modèle pix2pix [12] qui utilise une architecture de type U-Net [13].

### III. APPROCHE

Comme précisé plus haut le but de notre papier est de proposer une méthode permettant de générer des photos de fleurs à partir de dessin. Pour pouvoir effectuer cette tâche de synthèse sous contrainte consistant à transformer automatiquement une image d'entrée en une image synthétique, nous avons décidé d'utiliser une architecture de type *conditional Generative Adversarial Network*. Nous avons effectué ce choix car c'est ce genre d'architectures qui permettent de contrôler la sélection du vecteur latent retenu pour générer l'image de synthèse et donc de contrôler la structure de cette dernière. Dans une tout autre mesure, il est important de rappeler que l'efficacité des modèles antagonistes génératifs est fortement corrélée à la quantité de données utilisées pour entraîner ces modèles. Or, nous pouvons attester qu'il est assez difficile de trouver des paires dessin/image de fleurs sur Internet. C'est donc pour cette raison que nous avons décidé de générer nous mêmes ces paires. Notre approche consiste à télécharger un grand nombre de photos florales et à appliquer un traitement spécifique sur ces images, pour qu'en sortie, on puisse obtenir une image qui donne l'illusion qu'il s'agit d'un dessin fait par l'Homme. Le traitement appliqué aux diverses images sera abordé plus en détails dans la section suivante.



Fig. 3. De gauche à droite : Croquis généré grâce à une fonction se basant sur le filtre Sobel et Croquis généré grâce à une fonction se basant sur un détecteur de contour Canny.

## IV. EXPÉRIENCES

### A. Matériel utilisé

Pour pouvoir entraîner les divers modèles dont nous avons eu besoin au cours de notre projet, nous nous sommes principalement basés sur la solution : Google Colab. Cette dernière nous a permis d'accélérer l'apprentissage des modèles profonds notamment grâce à l'usage d'une carte GPU Tesla P100-PCIE ayant une capacité mémoire de 16GB.

### B. Librairies utilisées

Tous les modèles que nous avons créés au cours de cette étude ont été implémentés à partir de la bibliothèque **PyTorch**. Pour ce qui est des traitements qui ont été appliqués aux diverses images florales, nous nous sommes principalement servis de la librairie **OpenCV**.

### C. Génération des dessins

Dans le but de générer des données d'entraînement pour notre modèle, nous avons décidé d'implémenter un processus de génération de dessins à partir d'un jeu de photos de fleurs. Afin de générer des dessins réalistes à partir de photos, nous avons mis en place plusieurs fonctions, de la plus simple à la plus complexe pour tester différentes approches et différents aspects visuels. Lors de nos premiers essais, nous sommes partis du constat que les dessins humains se focalisent souvent sur les contours marqués d'une scène pour transmettre en quelques traits la structure globale ainsi que la quantité d'information nécessaire à l'œil humain pour reconstituer mentalement la scène. Nous avons donc implémenté une première fonction simple qui applique une détection de contours grâce à des filtres de type Sobel ou encore un détecteur de contours Canny.

Nous pouvons observer dans les résultats de la figure 3 que bien que ces fonctions nous permettent de conserver la plupart de l'information et que l'on y retrouve une certaine texture de dessin, nous sommes encore loin d'un dessin humain. De plus, beaucoup d'éléments de contexte viennent polluer le dessin de fleur en prenant en compte des éléments d'arrière-plan. Afin de favoriser un style de dessin de contours de fleurs, nous avons appliqué à ces résultats une binarisation par seuillage adaptatif. Cette opération permet de créer des images plus nettes avec des traits plus marqués. Ces

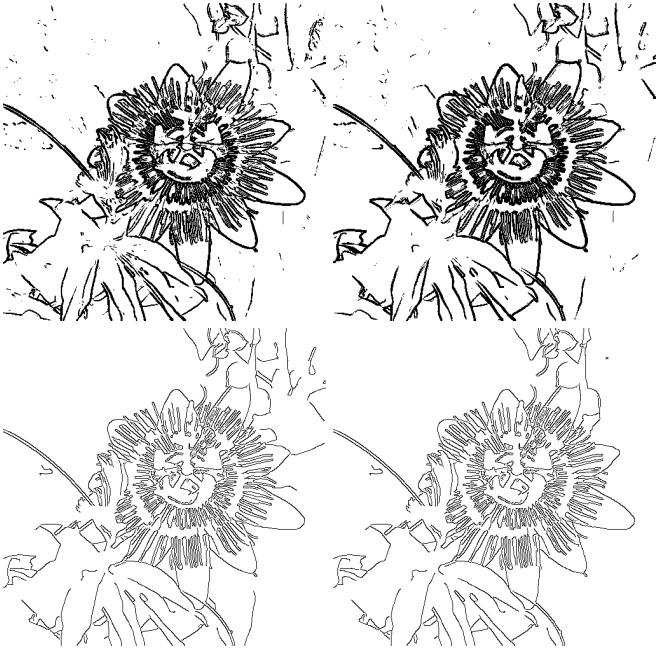


Fig. 4. De gauche à droite et de haut en bas : Croquis générés par respectivement des fonctions se basant sur un filtre sobel et le seuillage, un filtre sobel ainsi que du seuillage et de la suppression de bruit, une détection de contours de type Canny et de réduction de bruit et enfin une détection de contours avec la méthode Canny ainsi que de la réduction de bruit et de la suppression d'éléments isolés.

résultats étant encore fortement bruités, nous avons appliqué différents processus afin de tenter de réduire le bruit ou de supprimer les éléments isolés qui ne sont pas forcément utiles à la structure de l'image. Les résultats de ces expériences sont présentées dans la figure 4.

Nous avons envisagé de réaliser une segmentation des fleurs pour obtenir des dessins de fleurs sans arrière-plan. Face à notre jeu de données très varié, il était difficile d'émettre des postulats qui nous permettrait de discriminer efficacement la fleur de son arrière-plan. En effet, si, au premier abord, nous pouvons supposer que la fleur sera de couleur chaude sur un fond très souvent vert, nous avons observé dans notre jeu de données de nombreux contre-exemples qui limitent les approches de sélection de zone par valeur d'histogramme. Par exemple, certaines fleurs ne sont pas au centre de l'image et leur centre est de couleur verte alors que le fond est de couleur ocre (pot de fleur).

Nous avons donc testé des approches de segmentations agnostiques pour supprimer l'arrière-plan de nos images. Nous nous sommes notamment basés sur l'opération *grabcut* proposée par OpenCV. Cette fonction permet de supprimer l'arrière-plan sans action manuelle de l'utilisateur (au contraire de certains outils qui lui demandent de sélectionner lui-même les plans de l'image).

Nous pouvons observer dans les deux exemples présentés dans la Figure 5 que cette suppression de fond n'est pas évidente avec nos données. En effet, il est fréquent d'avoir des images où certaines feuilles sont au premier plan. Nous pouvons aussi observer dans d'autres images des éléments



Fig. 5. Image d'origine dans la colonne de gauche et résultat de suppression d'arrière plans dans celle de droite.

de contexte très marqués et qui sont facilement considérés comme de premier plan.

Face à cette diversité dans notre jeu de données, nous avons décidé de nous focaliser sur des approches qui permettraient de convertir l'intégralité de l'image en dessin. Nous avons notamment décidé d'implémenter l'approche proposée dans l'article [14]. Cette approche se base sur trois opérations à combiner pour appliquer un style dessiné à une photographie. Nous présentons ci-dessous ces trois étapes :

### 1) Génération d'une image dite "Stroke"

Cette première étape permet de générer une image à partir d'une étape de détection de contours. Une fois ces contours détectés, afin de passer à un style plus proche de celui du dessin, nous allons définir des directions et attribuer à chaque pixel de l'image une direction. Ensuite, grâce à une opération de convolution, nous allons pouvoir agréger les pixels proches associés à une même direction. Cette opération permet de connecter des pixels et de créer des petites lignes caractéristiques d'un dessin humain.

### 2) Génération d'une image dite "Tone"

Dans cette partie de la génération nous allons transformer l'histogramme d'une image en un histogramme caractéristique d'un dessin. En effet, les styles des dessinateurs sont souvent associés à des répartitions d'intensité bien différentes de celle d'une simple image en niveau de gris. Afin de prendre en compte ce facteur, cette étape de traitement permet de réaliser une mise en correspondance de deux histogrammes. Le premier histogramme est celui de l'image d'origine en niveau de gris et le second est celui que nous générions à partir de la distribution observée dans l'article [14].



Fig. 6. Dessins générés à partir d'une même image avec différents styles de dessins appliqués lors de la transformation.

### 3) Application d'un "Style" de dessin

Cette dernière opération permet de combiner les étapes précédentes et ce en ajoutant un style de dessin. En effet, nous allons combiner les images en y ajoutant un style de fond. Ce style de fond est issu d'une image de dessin uniforme au crayon selon une certaine direction et une certaine intensité. Afin de générer plus de diversité dans nos dessins, nous avons décidé de sélectionner aléatoirement le style de coup de crayon dans une liste d'images. Cette variété devait permettre par la suite à notre GAN d'apprendre sur des styles variés ce qui devrait faciliter la reconstitution d'images à partir de dessins réalisés par des humains.

Nous présentons dans la figure 6 un ensemble de génération de dessins à partir de notre fonction combinant les trois étapes décrites.

Les résultats obtenus ont permis de valider notre approche. En effet, comme nous pouvons le voir dans la figure 6, les résultats sont tout à fait convaincants et permettent d'obtenir de la variété dans la génération d'image grâce à la sélection aléatoire d'un style. De plus, ces résultats sont convaincants dans leur style de dessin humain contrairement à nos premiers essais qui avaient un aspect dessin "informatique".

### D. Colorisation des dessins

Afin de colorier nos dessins de fleurs, nous nous sommes tout d'abord penchés vers l'architecture **pix2pix** [12]. Nous avions fait ce choix car cette architecture est souvent mentionnée pour réaliser des tâches d'*image to image translation* d'un espace dimensionnel vers un espace de plus grande dimension tout en préservant les représentations du contenu original [15]. Toutefois, les résultats que nous avons obtenus

avec notre modèle n'étaient guère convaincants. En effet, on espérait que la fonction de coût de notre générateur soit de l'ordre de l'unité mais en pratique elle n'est jamais descendue en dessous de 10. Nous avons donc décidé de nous tourner vers une version améliorée de **pix2pix** qui se nomme **pix2pixHD** [16]. Cette nouvelle architecture se distingue de l'architecture pix2pix sur la façon dont les images sont synthétisées et la façon dont ces images sont discriminées. En ce qui concerne la synthèse des images, le modèle pix2pix se base sur l'architecture UNet [13]<sup>2</sup> pour pouvoir réaliser cette tâche. Le générateur de l'architecture pix2pixHD se base lui aussi sur un principe d'auto-encodage convolutif mais va également utiliser des blocs résiduels qui ont été popularisés par l'architecture ResNet de Microsoft [17]. Pour ce qui est de la partie discrimination, les deux architectures se basent sur un modèle de réseaux de neurones nommé **PatchGAN**. Il est important de rappeler que l'approche initiale du réseau discriminateur dans la tâche de création d'images est de définir si l'image est réelle ou fausse en l'observant dans son intégralité. Les discriminateurs de type PatchGAN vont quant à eux évaluer plusieurs régions de l'image, que l'on appelle des patches, et vont indiquer pour chaque patch si ce dernier a été généré ou s'il provient d'une "vraie" image. La subtilité de l'architecture pix2pixHD est d'utiliser plusieurs discriminateurs pour pouvoir étudier l'image à différentes échelles qui est une approche souvent utilisée en vision par ordinateur. Dans le cas de l'architecture pix2pixHD, le fait d'utiliser plusieurs discriminateurs en faisant passer des images à différentes échelles fait que chaque discriminateur va s'intéresser à des parties plus ou moins focalisées contenues dans l'image [18]. C'est le principe des **champs récepteurs**.

En ce qui concerne l'entraînement des deux réseaux de neurones, nous nous sommes servis d'images de fleurs issues du jeu de données « Oxford 102 flowers dataset » [19]. Ce dernier compte 8189 images couleurs de tailles diverses, chacune comportant environ un demi-mégapixel. À l'origine, ce jeu de données est destiné à un problème de classification à 102 classes, mais ici on cherche simplement à avoir un grand nombre de photos de fleurs de bonne qualité i.e. avec une résolution suffisante, une bonne netteté et un cadrage assez homogène des fleurs. Comme nous l'avons mentionné précédemment, l'efficacité du modèle génératif dépend de la quantité de données sur laquelle il s'est entraîné. C'est la raison pour laquelle nous avons décidé d'augmenter la taille de notre jeu de données en effectuant certains traitements sur les images de fleurs. De ce fait, nous avons procédé à des modifications sur les positions des fleurs, leur taille et leur couleur. Nous avons également effectué des opérations de rotation, cropping, symétries verticales ou horizontales, ajout de bruit gaussien, modification de la luminosité, du contraste, de la saturation ou de la palette de couleurs de

2. Toutefois, il est bon de préciser que certains points sont modifiés en comparaison avec le modèle original. Par exemple, au niveau de la fonction d'activation utilisée au niveau de la dernière couche de dé-convolution est la fonction *tanh* au lieu de la fonction *sigmoid* car l'intervalle de valeurs lorsqu'une image est normalisée est [-1,1].



Fig. 7. Coquelicot. De haut en bas et de gauche à droite : image originale de dimension 760 fois 500, square center crop, flip vertical sans recadrage, color shift, random square crop

l'image.

Avec toutes ces opérations, nous avons pu accroître le volume de notre jeu de données et ainsi passer d'un jeu initial contenant 8189 images de fleurs à un jeu qui en contient plus 49000. Toutefois, il nous a pas été possible d'entraîner le modèle pix2pix HD sur les 49000 images<sup>3</sup>. À partir de notre nouveau jeu de données, nous avons réservé 50 images pour tester notre modèle et le reste fut utilisé pour entraîner notre modèle. Certains de nos lecteurs (comme

3. Nous tenons à préciser que seul le modèle pix2pix a pu être entraîné sur les 49000 images. En effet, par manque de temps, nous n'avons pas pu implémenter notre propre version du modèle pix2pixHD. Nous nous sommes donc servis du modèle mis à disposition par les chercheurs de NVIDIA. Toutefois, nous avons rencontré des difficultés à lancer l'entraînement de leur modèle sur le mésocentre de notre école et c'est donc pour cette raison que nous nous sommes rabattus sur Google Colab. Par contre, pour des raisons d'espace mémoire, nous n'avons pu charger que 6000 images. Nous conservons donc nos 49000 images pour de futures expérimentations.

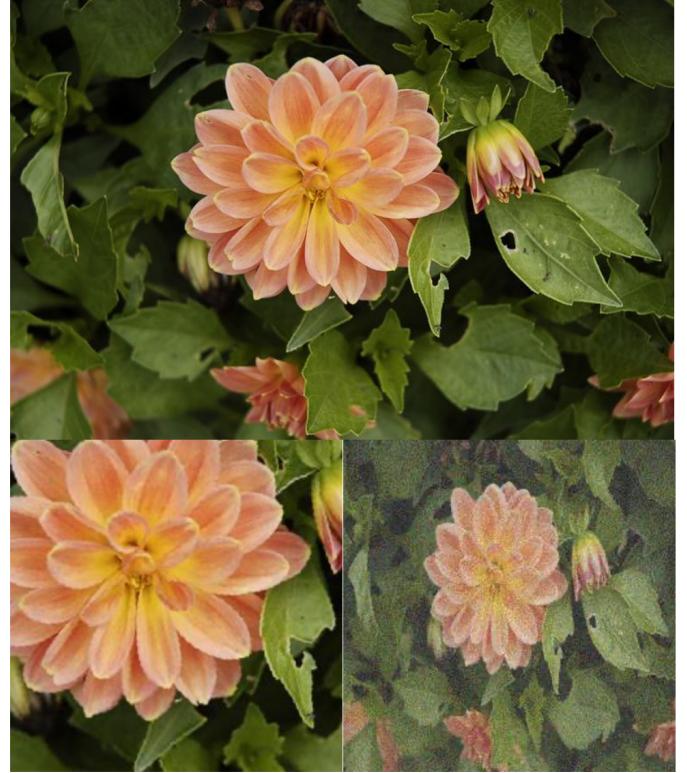


Fig. 8. Dahlia. De haut en bas et de gauche à droite : image originale de dimension 752 fois 500, random square crop et rotation, ajout de bruit gaussien.

notre professeur d'apprentissage profond) pourraient crier au scandale car nous n'avons pas défini un jeu de validation mais il revient souvent dans des blogs que les paramètres mentionnés dans les papiers de recherche sont suffisant pour avoir des modèle performant. De même, il est bon de rappeler qu'on n'essaye pas de développer un modèle qui surpasse les modèles existants mais qu'on souhaite valider une intuition pour générer des photo-réalistes. Par ailleurs, il est également important de préciser que nous effectuons une étape de redimensionnement des images avant de les passer à notre générateur pour accélérer la durée d'entraînement des modèles. Dans notre étude, nous avons défini la taille de redimensionnement à 256 par 256.

## V. ANALYSE DES RÉSULTATS

Dans cette section, nous allons analyser les différents résultats obtenus. Tout d'abord, nous allons présenter les résultats de notre modèle sur des images de test transformées par notre fonction de dessin. Ensuite, nous analyserons les résultats générés à partir de dessins humains obtenus sur internet.

### A. Résultats sur des dessins issus de notre jeu de données

Dans la figure 9, nous pouvons observer que notre modèle a bien appris à générer des images photo-réalistes à partir de nos dessins. En effet, nous pouvons observer que le modèle réussit bien à dissocier les éléments associés à la fleur et générer des couleurs réalistes. Il est important de noter



Fig. 9. Images photo-réalistes générées par notre modèle à partir d'images issues de notre jeu de données de test.

que l'image d'entrée ne contient plus aucune information associée à la couleur. Ici nous avons des fleurs de couleur rose pâle, rose orangé, jaunes ou encore rose bonbon. Nous pouvons aussi remarquer que le centre des fleurs est bien associé à une couleur différente, phénomène que l'on observe souvent dans la réalité. De plus, l'arrière plan est lui aussi associé à des textures et des couleurs photo-réalistes, que ce soit les bourgeons ou les feuilles. Par ailleurs, nous pouvons noter que la diversité de fond semble poser problème à notre modèle sur certains exemples où la dissociation de certains pétales du fond n'est pas parfaitement efficace. Comme nous

avions supposé lors de notre phase de génération automatique de dessin, il semblerait que l'arrière plan soit l'élément le plus limitant. De plus, pour des exemples réalisés par des dessinateurs humains, l'arrière plan est rarement présent.

À ce stade, nous pouvons tout de même souligner que notre modèle semble bien réaliser la tâche attendue. Toutefois, dû à notre puissance de calcul limité, nous n'avons pas pu entraîner notre modèle sur nos 49000 dessins. Ce point sera sûrement limitant lorsque nous allons tester la généralisation de notre modèle à des dessins de styles différents. En effet, notre modèle a bien appris à générer des photos à partir de nos dessins mais nous sommes partis du postulat que notre transformation en dessin était une transformation très proche de celle effectuée par un dessinateur humain. Nous allons voir par la suite que ce postulat était sûrement trop fort et que nous aurions dû, si le temps nous l'avais permis, implémenter d'autres fonctions de dessin pour obtenir plus de diversité de styles. Cela aurait sûrement permis à notre modèle de mieux approximer la fonction inconnue de transformation de photo de fleur en dessin "humain".

#### B. Résultats sur des dessins humains

Nous pouvons observer dans la figure 10 que notre modèle performe très bien sur des instances dont le style est plus proche de notre transformation, comme dans le cas de la première fleur. Les quelques indications d'arrière-plan permettent à notre modèle de générer un fond plus réaliste et le style de dessin permet de reconstituer la fleur à un niveau proche du photo-réalisme. Dans l'exemple de l'orchidée, nous pouvons observer les principales limitations de notre modèle, lorsque l'on se retrouve face à un style de dessin différent. Notre modèle réussit bien à identifier les éléments de la fleur, de la branche, des bourgeons et ce avec des couleurs réalistes. Toutefois, le bruit observé sur le dessin au niveau des fleurs perturbe fortement la génération de couleur. Même si le style de dessin est différent, nous pouvons supposer qu'en entraînant notre modèle sur nos 49000 dessins, nous pourrions minimiser ce genre de défauts.

#### C. Transfert de style

En appliquant la colorisation d'image à des dessins produits par des humains, nous nous sommes rendus compte que le fond de l'image couleur obtenue était souvent uniforme, bleu ou vert, un peu flou, ce qui n'est pas très réaliste. Cela vient du fait que les dessins humains accordent peu ou pas d'importance à la représentation de l'arrière-plan, alors que cet élément visuel est essentiel au photo-réalisme de l'image finale. Nous avons alors pensé à un transfert de style entre un dessin synthétique et un dessin humain pour améliorer l'arrière-plan. Les résultats sont encourageants comme on peut le constater dans l'exemple suivant.

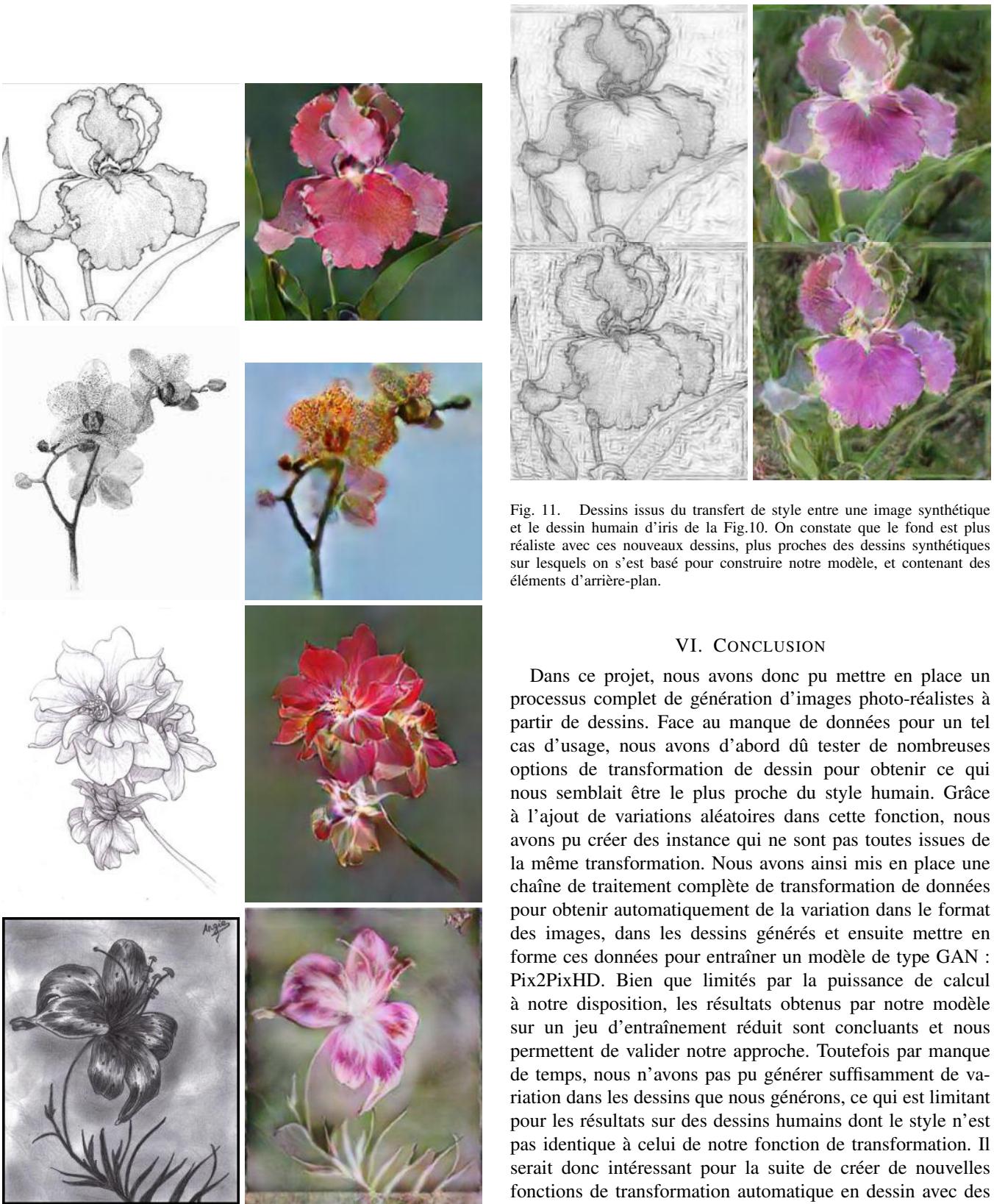


Fig. 10. Images photo-réalistes générées par notre modèle à partir d'images issues de dessins humains.



Fig. 11. Dessins issus du transfert de style entre une image synthétique et le dessin humain d'iris de la Fig.10. On constate que le fond est plus réaliste avec ces nouveaux dessins, plus proches des dessins synthétiques sur lesquels on s'est basé pour construire notre modèle, et contenant des éléments d'arrière-plan.

## VI. CONCLUSION

Dans ce projet, nous avons donc pu mettre en place un processus complet de génération d'images photo-réalistes à partir de dessins. Face au manque de données pour un tel cas d'usage, nous avons d'abord dû tester de nombreuses options de transformation de dessin pour obtenir ce qui nous semblait être le plus proche du style humain. Grâce à l'ajout de variations aléatoires dans cette fonction, nous avons pu créer des instances qui ne sont pas toutes issues de la même transformation. Nous avons ainsi mis en place une chaîne de traitement complète de transformation de données pour obtenir automatiquement de la variation dans le format des images, dans les dessins générés et ensuite mettre en forme ces données pour entraîner un modèle de type GAN : Pix2PixHD. Bien que limités par la puissance de calcul à notre disposition, les résultats obtenus par notre modèle sur un jeu d'entraînement réduit sont concluants et nous permettent de valider notre approche. Toutefois par manque de temps, nous n'avons pas pu générer suffisamment de variation dans les dessins que nous générerons, ce qui est limitant pour les résultats sur des dessins humains dont le style n'est pas identique à celui de notre fonction de transformation. Il serait donc intéressant pour la suite de créer de nouvelles fonctions de transformation automatique en dessin avec des styles plus variés. Nous pourrions ainsi générer notre jeu d'entraînement avec davantage de diversité et permettre à notre modèle de mieux généraliser aux différents types de dessins humains.

## REFERENCES

- [1] James Vincent. How three French students used borrowed code to put the first AI portrait in Christie's, October 2018.
- [2] Oscar Holland. NFT art : Sophia the Robot 'self-portrait' sells for almost \$700K at Nifty Gateway auction - CNN Style.
- [3] deepart.io. Turn any photo into an artwork - for free !
- [4] Derrière La Caméra. Photoshop 2021 - Neural Filters (les nouveautés), October 2020.
- [5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A Neural Algorithm of Artistic Style. *arXiv :1508.06576 [cs, q-bio]*, September 2015. arXiv : 1508.06576.
- [6] Chuan Li and Michael Wand. Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis. *arXiv :1601.04589 [cs]*, January 2016. arXiv : 1601.04589.
- [7] Yağmur Güçlütürk, Umut Güçlü, Rob van Lier, and Marcel A. J. van Gerven. Convolutional Sketch Inversion. *arXiv :1606.03073 [cs]*, 9913 :810–824, 2016. arXiv : 1606.03073.
- [8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. *arXiv :1603.08155 [cs]*, March 2016. arXiv : 1603.08155.
- [9] Shu-Yu Chen, Wanchao Su, Lin Gao, Shihong Xia, and Hongbo Fu. Deep Generation of Face Images from Sketches. *arXiv :2006.01047 [cs]*, June 2020. arXiv : 2006.01047.
- [10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv :1406.2661 [cs, stat]*, June 2014. arXiv : 1406.2661.
- [11] I - Principe · GANs.
- [12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.
- [13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pages 234–241. Springer International Publishing.
- [14] Cewu Lu, Li Xu, and Jiaya Jia. Combining sketch and tone for pencil drawing production. page 9.
- [15] Yingxue Pang, Jianxin Lin, Tao Qin, and Zhibo Chen. Image-to-Image Translation : Methods and Applications. *arXiv :2101.08629 [cs]*, January 2021. arXiv : 2101.08629.
- [16] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *arXiv :1711.11585 [cs]*, August 2018. arXiv : 1711.11585.
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv :1512.03385 [cs]*, December 2015. arXiv : 1512.03385.
- [18] UCF CRCV. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs, March 2018.
- [19] Visual Geometry Group - University of Oxford.