Université de Paris
UFR Mathématiques et Informatique

Projet TER - StyleGANov'

Master 1 Vision Machine Intelligente

Jiaxin XUE – Billal IHADDADEN
Encadré par Olivier RISSER-MAROIX

Année universitaire 2020-2021

Table des matières

Chapitre 1

Introduction

Le jugement de la similarité dimages par lhumain sappuie sur beaucoup de choses notamment les éléments de la scène et les aspects culturels. Pour nous les humains il nous est facile de juger de la similarité entre deux images, cependant la prédiction de la similarité perceptive humaine est un sujet de recherche difficile. Le processus visuel sous-jacent à cet aspect de la vision humaine fait appel à plusieurs niveaux différents danalyse visuelle (formes, objets, texture, disposition, couleuretc). Dans le cas de ce projet la similarité purement visuelle sera traitée sans prendre en compte la sémantique. Le transfert de style et la génération dimage par des architectures modernes de réseaux de neurones (VAE, ADalN, PIX2PIX, CycleGAN, BigGANetc). La première partie consiste à expérimenter la génération dimage avec pix2pix en utilisant les images de lensemble de données PASCAL avec et sans larrière-plan. Ensuite Nous allons expérimenter le BigGAN pré entrainé sur lensemble de données dImageNet.

Chapitre 2

État de l'art

2.1 Similarité visuelle en science cognitives

Les objets peuvent être caractérisés selon un grand nombre de critères possibles, mais certaines caractéristiques sont plus utiles que d'autres pour donner un sens aux objets qui nous entourent. Dans [1] ils ont développé un modèle informatique de jugements de similarité pour les images du monde réel de 1854 objets. Le modèle a capturé la plupart de la variance explicable dans le jugement de la similarité et a produit 49 dimensions d'objets hautement reproductibles et significatives qui reflètent diverses propriétés conceptuelles et perceptuelles de ces objets. Ces dimensions prédisent le comportement de catégorisation externe et reflètent les jugements de similarité de ces catégories. D'après eux les humains peuvent évaluer avec précision les objets selon ces dimensions, ce qui conclue que les jugements de similarité humains peuvent être représentés par un ensemble de dimensions relativement réduit, interprétable et généralisable aux comportements externes.

2.2 Similarité visuelle

Les progrès récents des réseaux de neurones artificiels ont révolutionné la vision par ordinateur, mais ces systèmes de conception sont toujours surpassés par les humains, dans [2] ils ont comparé la perception des objets par le cerveau humain et par les machines (certain nombre de modèles informatiques, par exemple : Tal, Gabor, Hog, split-halfetc.). Ils ont recueilli un vaste ensemble de données comprenant 26 675 mesures de dissimilarité perçue pour 2801 objets visuels chez 269 sujets humains. Afin de mesurer la dissimilarité chez les humains, ils ont demandé de localiser une bille étrange dans un tableau contenant un objet parmi de multiples occurrences de l'autre. La réciproque du temps de recherche visuelle a été considérée comme une estimation de la dissimilarité perçue. Cette mesure se comporte comme une distance mathématique, elle présente une somme linéaire de plusieurs caractéristiques, elle explique la catégorisation visuelle rapide et elle est fortement corrélée avec les évaluations subjectives de la dissimilarité. Ils ont testé l'ensemble de mesures sur des modèles de calcul très répandu. Le meilleur modèle était un CNN mais il a été surpassé par la combinaison de tous les autres modèles. Leur conclusion était que tous les modèles informatiques montrent des modèles similaires d'écart par rapport à la perception humaine.

Dans [3] ils voulaient savoir est-ce que l'apprentissage automatique peut expliquer les

jugements humains de similarité de forme d'objets visuels. Alors ils ont analysé la performance des systèmes d'apprentissage métrique (distance ou similarité) y compris les DNN, sur un nouvel ensemble de données de jugement de similarité de forme d'objet visuel humain. Contrairement aux autres études ou ils demandaient aux participants de juger de la similarité lorsque les objets ou les scènes étaient rendus à partir d'un seul point de vue, eux ils ont utilisé un rendu à partir de plusieurs points vues et ils ont demandé aux participants de juger de la similarité de forme de manière variable. Ils ont trouvé que les DNN ne parviennent pas à expliquer les données expérimentales, mais une méthode entraînée avec une représentation variable basée sur des parties produit un bon ajustement, ils ont aussi constaté que même si les DNN puissent apprendre à extraire la représentation basée sur les parties et devrait être capable d'apprendre à modéliser leurs données. Les réseaux entraînés avec une fonction triplet loss basée sur le jugement de similarité ne donne pas un bon résultat. Le mauvais résultat du DNN est causé par la non-convexité du problème d'optimisation dans l'espace des poids du réseau. Ils concluent que l'insensibilité du point de vue est un aspect critique de la perception de la forme visuelle humaine, et que les réseaux de neurones et d'autres méthodes d'apprentissage automatique devront apprendre des représentations insensibles au point de vue afin de rendre compte des jugements de similarité de forme d'objets visuels des humains.

La comparaison des représentations formées par les DNN avec celles utilisées par les humains est un défi, car les représentations psychologiques humaines ne peuvent pas être observées directement. Dans [4] ils ont évalué et proposé une amélioration de la correspondance entre les DNN et les représentations humaines. Leur approche consiste à résoudre le problème de comparaison en exploitant la relation étroite entre représentation et similarité, ce qui veut dire que pour chaque fonction de similarité sur un ensemble de paires de points de données correspond à une représentation implicite de ces points. Ce qui offre une base empirique pour la première évaluation de DNN en tant qu'une approximation des représentations psychologiques humaines.

Dans [5] ils ont démontré leur méthode sur le jeu de données CUB-200-2011 et Stanford Cars en appliquant leur architecture du DNN ProtoPNet. Leur expérience a montré que l'architecture de DNN qu'ils ont créé pouvait atteindre une précision comparable à avec ses analogues. Et lorsque plusieurs ProtoPNet sont combinés en un réseau plus vaste, celui-ci peut atteindre une précision équivalente à celle de certains des modèles profonds les plus performants. De plus, leur modèle offre un niveau d'interprétabilité qui est absent dans les modèles existants.

2.3 transfert du style et GAN

Le transfert de style désigne une catégorie d'algorithme qui manipules les images numérique afin d'adopter l'apparence ou le style. La transformation de l'image se fait grâce aux réseaux neuronaux profonds. La géométrie et la texture sont des aspects fondamentaux du style visuel. Ils existent beaucoup de méthodes de transfert de style mais qui se concentrent que sur la texture et pas la géométrie. Dans [6] ils ont proposé un transfert de style déformable DTS, une approche basée sur l'optimisation qui stylise à la fois la texture et la géométrie d'une image pour mieux correspondre à une autre image (similarité). Leur approche n'est pas limitée à un domaine particulier (comme le visage) et surtout elle n'a pas besoin d'un apprentissage de paires style/contenu. Elle est basée sur des points clés sélectionnés dans l'image source et l'image but ensuite ils vont rapprocher ces points clés

afin de changer la forme et enfin ils changent la texture.

En 2014, Ian Goodfellow et al. [7] ont publié un article intitulé « Adversarial Generative Networks », qui propose une nouvelle méthode de training le modèle génératif. L'article indique que le GAN peut générer de nouveaux images des numériques manuscrites de MNIST, des CIFAR-10, et de TFD après un entrainement comme jeu. Ces dernière années, GAN a beaucoup d'application dans le domaine de génération d'uvres d'art, et de transfert du style.

Chapitre 3

Méthodes étudiées

Depuis que Ian Goodfellow a proposé le GAN (Generative Adversarial Network) en 2014, la recherche sur le GAN est très active. Diverses variantes du GAN ne cessent d'apparaître. Yann LeCun a même déclaré que le GAN était « adversarial training is the coolest thing since sliced bread. » en 2016. Nous nous donc focaliserons dans un sur la génération de paires d'images visuellement similaires via des GANs.

3.1 Generative Adversarial Network

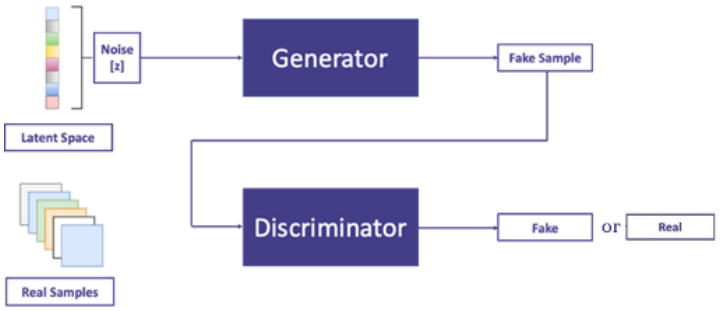

FIGURE 3.1 – architecture GAN

Generative Adversarial Network (GAN) est une méthode d'apprentissage non supervisé, consistant à faire jouer deux réseaux neuronaux l'un contre l'autre. Les GANs se composent d'un réseau génératif (générateur) et d'un réseau discriminatif (discriminateur).

— **Le générateur G**, qui apprend à générer des données vraisemblables. Les instances générées deviennent des exemples d'entrainement négatifs pour le discriminateur.
— **Le discriminateur D**, qui apprend à distinguer les fausses données réelles. Il pénalise le générateur lorsqu'il produit des résultats invraisemblables.

Dans entraînement de GAN, le générateur génère un échantillon (ex. une image), tandis que son adversaire, le discriminateur essaie de distinguer si cet échantillon est réel ou non. Le but du générateur est de pouvoir tromper le discriminateur autant que possible. Les

deux réseaux jouent l'un contre l'autre et ajustent constamment leurs paramètres, dans le but ultime de rendre le discriminateur incapable de déterminer si la sortie du réseau génératif est fausse.

Au début de l'étape d'entrainement, le générateur produit des données clairement fausses, et le discriminateur apprend rapidement à dire que c'est des fausses données. Au fur et à mesure que l'entrainement progresse, le générateur se rapproche de la production de données qui peuvent tromper le discriminateur. Enfin, le discriminateur a de plus en plus du mal à détecter les fausses données. Et commence à les classer comme vraies.

Un des principales contributions de GAN est la stratégie de training qui rend le discriminateur pouvoir reconnaitre la distribution appris par générateur et la distribution réelle. Et, face à des problèmes tels que la difficulté de training, il y a de nombreux améliorations et développements sur l'original tels que c-GAN, cycle-GAN, styleGAN.

3.2 pixp2ix

pix2pix modèle propose un cadre général pour la tâche de la traduction d'image à image en combinant cGAN pour réaliser la traduction d'image du domaine source au domaine cible. Le réseau apprend non seulement la correspondance entre l'image d'entrée à l'image de sortie, mais apprennent également une fonction de perte pour entraîner cette correspondance. Cela permet d'appliquer la même approche générique à des problèmes qui, traditionnellement nécessiteraient des formulations de perte très différentes[8].

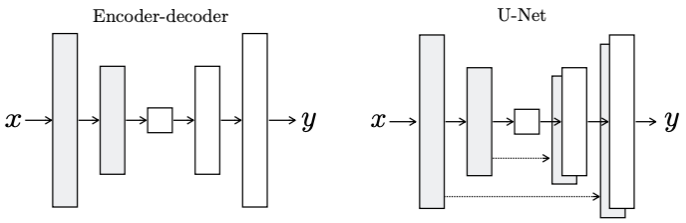

FIGURE 3.2 – U-Net est un encoder-decoder avec skip-connections en forme de miroir entre layers dans l'encoder et dans le decoder

En termes de générateur, pix2pix utilise Unet comme générateur, en considérant que l'aspect de surface des images d'entrée et de sortie doit être différent alors que la structure de base doit être similaire, et que pour la tâche de traduction d'image, l'entrée et la sortie doivent partager certaines informations de base (ex. contours), ils appliquent donc une connexion de layer-skipping comme l'approche de connexion dans Unet.

En termes de discriminateur, l'auteur a crée PatchGAN comme discriminateur, au lieu de discriminateur qui base sur les distance traditionnelles L1, L2 dans les travaux précédents. L'idée de PatchGAN est de diviser l'image en partie avec over-lapping, ensuite juger la vérité ou la fausseté de chaque patch séparément. Les auteurs concluent en disant que le PatchGAN proposé peut être considéré comme une autre forme de perte de texture ou de perte de style.

Avec pix2pix, si nous fournissons des pairs d'images similaires, nous pouvons former un modèle qui prend une image d'entrée et nous génère une image similaire.

3.3 BigGAN et Espace de Latent

FIGURE 3.3 – exemples des images généré par BigGAN avec différents vecteur de classe comme entrée[9]

Grâce à l'avènement des GANs, les algorithmes de modélisation générative d'images ont fait de grands progrès ces dernières années pour générer des images réalistes et diverses. Cependant, la génération d'images diverses et à haute résolution à partir d'ensembles de données complexes comme ImageNet reste une tâche difficile. BigGAN est le modèle génératif le plus grand et le mieux entrainé à ce jour. Les images générées par BigGAN atteignent un niveau de réalisme extrêmement élevé[9].

Bien sûr, nous n'avons pas l'intention de former un GAN à partir de zéro, et les modèles de type BigGAN nécessitent d'énormes ressources informatiques pour être formés. Avec l'aide du BigGAN déjà bien formé sur ImageNet, nous considérons essayer de faire varier l'entrée du générateur de manière conditionnelle. Cela implique l'espace latent.

FIGURE 3.4 – exemples des images généré par BigGAN en controlant l'espace latent[10]

Härkönen, Erik, et al[10] décrit une technique simple pour analyser GAN et créer des commandes interprétables pour la synthèse d'images, comme le changement de point de vue, le vieillissement, l'éclairage et l'heure de la journée.

Inspiré par les deux méthodes ci-dessus, nous avons commencé à faire la génération d'image similaire sous condition avec le bien formé BigGAN.

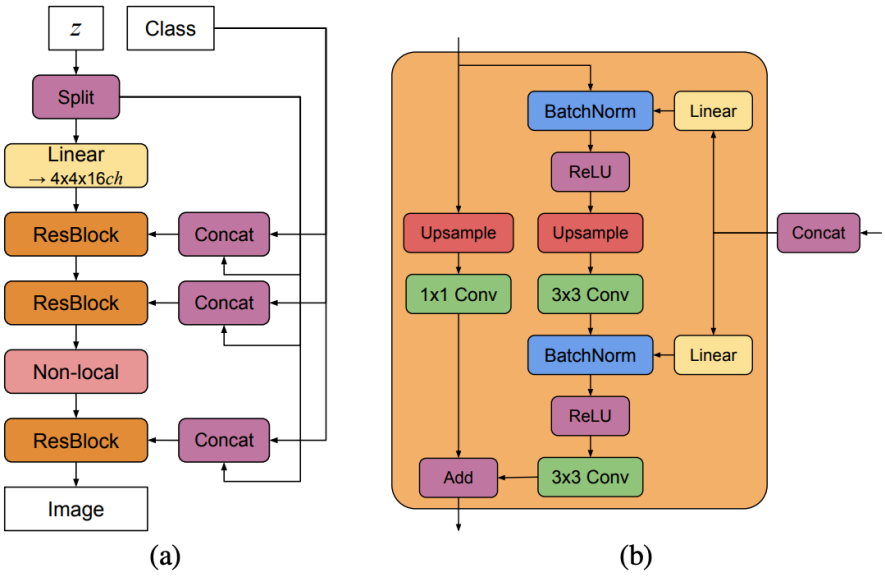

FIGURE 3.5 – architecture de BigGAN, (a) est générateur typique, (b) est un Residual Block de générateur

Si on observe l'architecture, on trouver le générateur prend deux types de vecteur comme latent, le vecteur de bruit (normalement obtenu par la distribution Gaussinne) et le vecteur de classe. En changeant les vecteurs d'entrée, on obtient l'image générée différente. Nous donc déroulons les expériences suivantes, esseyons de définir un loss fonction qui permet d'optimiser la similarité entre l'image généré par BigGAN et l'image donnée. Par exemple construire un singe le plus ressemblant à Einstein qui tire la langue.

Chapitre 4

Méthodes expérimentales

4.1 pix2pix avec PASCAL

Nous avons implémenté le modèle pix2pix sur un sous ensemble de données d'image prévenant de la dataset Pascal, cet ensemble contient 2913 paires d'images (image d'origine et segmenté) sur 20 classes différentes :

- Personne : personne
- Animal : oiseau, chat, vache, chien, cheval, mouton
- Véhicule : avion, bicyclette, bateau, bus, voiture, moto, train
- Intérieur : bouteille, chaise, table à manger, plante en pot, canapé, télévision/moniteur

Comme la plupart des utilisateurs de pix2pix sont basées sur des jeux de données à une classe, tels que le visage, la façadeetc. Nous menons des expériences sur le jeu de données qui a une scène plus complexe, qui souvent contient plus d'une classe dans une image, par exemple :

FIGURE 4.1 – Image segmenté et image d'origine contenant une personne et un cheval

Nous avons effectué notre expérience sur le pix2pix en trois parties :

(a) Sur les voitures

Tout d'abord, nous avons mené notre expérience sur le sous-ensemble de données qui contient principalement que des voitures. Après 200 époques d'apprentissage avec une taille de lot de 1, le modèle est capable de générer le profil général d'une voiture, et il est capable de remarquer la position des pneus et des fenêtres. De plus, comme la plupart des voitures dans l'ensemble du jeu de données sont noirs et rouge, et pix2pix est un GAN basé sur la théorie statique. Les voitures générées sont principalement en noir avec un mélange de rouge. (Exemple en annexe).

(b) Sur toute les images avec leurs arrière-plans

Deuxièmement nous avons nous avons mené l'expérience avec toute des images, et la plupart de ces images présente des scènes complexes avec plusieurs classe en même temps. Après 400 époques d'apprentissage avec une taille de lot de 1, Malgré que la classe personne est la plus redondante dans nos images, il est difficile de générer une personne réelle à partir de l'image d'entrée. Et pour certaines classes simples comme les avions, vélos, bus et les bateaux qui apparaissent souvent dans des scènes monotones (ciel, routes, meretc.), le modèle est capable de générer un objet très similaires à un objet réel. (Exemple en annexe).

(c) Sur toute les images sans leurs arrière-plans

Enfin, nous avons expérimenté sur l'ensemble des images en augmentant le nombre des images par la duplication (miroir et rotation) et en enlevant l'arrière-plan afin de savoir si l'arrière-plan influence. L'une des raisons est que les scènes ou les personnes apparaissent sont très variées et après 300 époques d'entrainement avec une taille de lot de 1, nous avons générer des images (Exemple en annexe). Et nous avons observé qu'il n'y a pas de d'amélioration par rapport à l'étape avec les images contenants l'arrière-plan.

À la fin de ces expériences nous avons dessiné les graphes de loss afin d'observer le comportement de la loss du générateur et notamment la loss du discriminateur. Nous notons que le D_real avec la ligne verte signifie la loss du discriminateur pour les images réelles, le D_fake signifie la loss du discriminateur pour les images fausses g.n.r.es enfin nous avons G_GAN qui signifie la loss du générateur pour les images fausses. Nous notons aussi que la stabilité de notre modèle est liée à la stabilité des loss de D_real, D_fake et G_GAN.

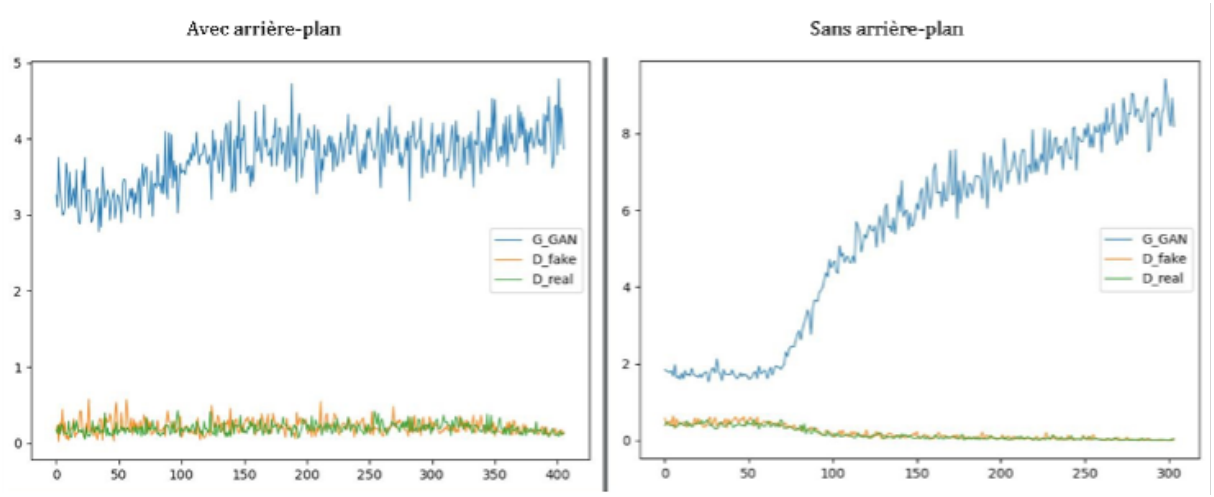

FIGURE 4.2 – Graphe de loss de pix2pix avec les images contenant un arrière-plan et non.

Nous remarquant que notre modèle avec des images contenant un arrière-plan est stable car G_GAN, D_fake et D_real sont stable. Et pour le modèle avec les images sans arrière-plan, il est stable jusqu'à l'époque 75, puis devient instable car la loss de G_GAN augment énormément. Nous avons aussi appliqué le modèle avec l'arrière-plan sur des images qui ne contiennent pas d'arrière-plan afin le modèle permet de générer un arrière-plan presque parfait pour certaines classes par exemple pour les avions il génère le ciel, pour les animaux il génère la verdure (Exemple en annexe).

4.2 Espace latent de BigGAN

Après avoir fini l'étape 1 ou nous avons obtenu des résultats pas très satisfaisants sur la génération des images similaires, nous nous sommes redirigés vers l'étude de BigGAN. Dans cette partie nous avons expérimenté BigGan de DeepMind, ce GAN a été entrainé sur l'imageNet qui contient 1000 classes différentes. Il a été entrainé à une résolution de 128*128, 256*256 et 512*512. Nous avons utilisé le modèle avec la résolution de 128*128.

Nous avons fait deux expériences pour cette partie qui sont :

1. Nous fixons la classe et nous changeons le bruit.

2. Nous changeons la classe et le bruit.

4.2.1 Expérience 1

Pour la première expérience dans cette étape nous commençons par générer l'image target au hasard ensuite nous allons fixer le vecteur de classe encoder en one-hot et nous changeons le bruit en l'optimisant avec les loss en utilisant la descente de gradient stochastique

Adam.

Entrées *une image target* I_0 ;
Résultat : imge I_final le plus ressemblant à I_0
$n = 1$
Donner un vecteur de bruit aléatoirement
for $n <$ *numbre d'iteration* **do**
    Générer Image I_n par BigGAN avec le vecteur de bruit
    Calcule de la loss sémantique L2 entre I_0 et I_n (replace classify block de squeezenet by a flatten layer)
    Calcule de la loss pixel entre I_0 et I_n
    loss = − loss sémantique + 30*loss pixel (on se focalise beaucoup plus sur la loss pixel pour cette étape)
    faire backpropagation et optimiser le vecteur de bruit
end

Algorithme 1 : vecteur de classe fixe

Nous avons appliqué notre algorithme sur 4 exemples (les images cadré en rouge sont des images but et les images cadré en vers sont le résultat final et les autres sont les étapes intermédiaires.)

L'exemple 1 nous avons essayé de générer un tigre à partir d'un chat sans changer la classe, en optimisant que le bruit.

FIGURE 4.3 – Chat vers le tigre

L'exemple 2 nous avons essayé de générer un TGV à partir d'un camion sans changer la classe, en optimisant que le bruit.

FIGURE 4.4 – Camion vers train

L'exemple 3 nous avons essayé de générer un chien à partir d'un chat sans changer la classe, en optimisant que le bruit.

FIGURE 4.5 – chat vers chien

L'exemple 4 nous avons essayé de générer une voiture à partir d'un chien sans changer la classe, en optimisant que le bruit.

FIGURE 4.6 – chien vers voiture

4.2.2 Expérience 2

Pour la deuxième expérience dans cette étape nous commençons par générer l'image target au hasard ensuite nous allons changer le vecteur de classe et le bruit en les optimisant avec la descente de gradient stochastique Adam, et vu que le vecteur de classe est encoder en one-hot nous allons le transformer avec la fonction softmax afin qu'il puisse être optimisé.

Entrées *une image target* I_0 ;
Résultat : imge I_final le plus ressemblant à I_0
$n = 1$
Donner un vecteur de bruit aléatoirement
Donner un vecteur de classe en forme de one hot coding
for $n <$ *numbre d'iteration* **do**
    vecteur de classe = softmax(vecteur de classe)
    Générer Image I_n par BigGAN avec le vecteur de bruit et le vecteur de classee
    Calcule de la loss sémantique cosine entre I_0 et I_n (replace classify block de squeezenet by a flatten layer)
    Calcule de la loss pixel entre I_0 et I_n
    loss = $-100*$ loss sémantique + 30*loss pixel
    faire backpropagation et optimiser le vecteur de bruit et le vecteur de classe
end

Algorithme 2 : vecteur de classe non fixe

Nous avons appliqué notre algorithme sur 3 exemples (les images cadré en rouge sont des images but et les images cadré en vers sont le résultat final et les autres sont les étapes intermédiaires).

L'exemple 1 nous avons essayé de générer un tigre à partir d'un chat en optimisant la classe et le bruit.

FIGURE 4.7 – Chat vers tigre

L'exemple 2 nous avons essayé de générer un chien à partir d'un chat en optimisant la classe et le bruit.

FIGURE 4.8 – Chat vers chien

L'exemple 3 nous avons essayé de générer une voiture à partir d'un chien en optimisant la classe et le bruit.

FIGURE 4.9 – Chien vers voiture

Nous remarquant que quand on fixe la classe et on optimise juste le bruit notre modèle converge vers quelque chose de similaire à notre image but, sauf si les deux classes qui se ressemble (chien vers chat, chat vers tigre, camion vers train..etc)

Nous remarquant que quand nous optimisant le bruit et le vecteur de classe nous avons beaucoup d'objets déformer sans aucun sens qui se génèrent et on converge généralement vers une image qui est complétement différente à notre image but.

Un principale raison est que dans l'architecture de BigGAN, le classe vecteur n'existe que dans l'espace latent, il est connecté à plusieurs intermédiaire layer, voire il est une variable

de haute dimension. Cela apport la labilité sur l'optimisation. Et l'autre raison vient de softmax fonction, étant donnée qu'il y a un mile classes, après avoir utilisé softmax sur one hot coding, la classes vecteur devient (0.001, 0.001, ...), qui mal initialise le vecteur au début de l'optimisation. Et face à cette problème, on a aussi essayé retourner la classes vecteur à la forme one hot coding lord d'entrer à BigGAN, en assurant l'usage de autograd de Pytorch. On a utilisé le gumbel softmax qui peut-etre approprié. Cependant, il demande trop d'itération pour être convergé et il rend le processus de l'optimiser le vecteur de classe extrêmement aléatoire puisque c'est une variable de mile dimension.

4.2.3 Expérience supplémentaire, Singe ver Einstein

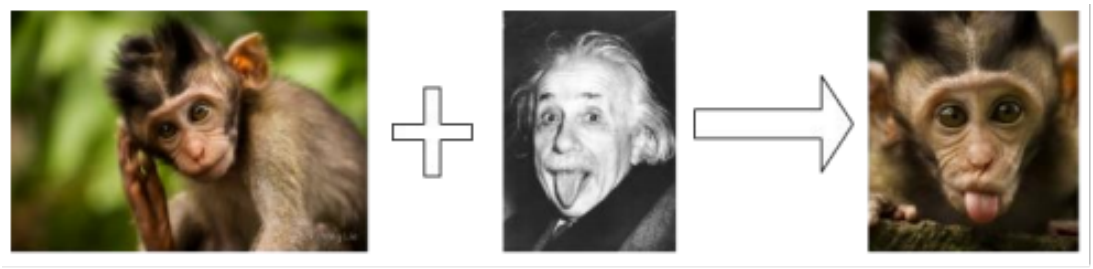

FIGURE 4.10 – exemple but

En fait toutes les expérience ont faites dans le cadre de ImageNet, c'est à dire que toutes images target et images généré sont de ImageNet. Donc, on pense à faire une expérience supplémentaire, qui est dehors ImageNet. Nous avons donc essayé d'obtenir le résultat de la figure 10 qui est présente sur l'énoncé du sujet, et cela en utilisant l'algorithme 2 et nous avons obtenu le résultat de la figure 11.

FIGURE 4.11 – Singe vers Einstein

Nous remarquons que sur les images des étapes intermédiaires il essaie de générer une tête similaire à celle d'Einstein en générant les mêmes cheveux au niveau de son oreille et aussi il essaie de trouver comment créer la langue. Malheureusement il ne pourra pas faire cela et vers la fin nous remarquons que l'émotion d'Einstein et le singe sont différentes.

Chapitre 5

Conclusion et Perspectives

Pour la première étape nous avons réussis à générer des images grâce à pix2pix nous avons eu 3 modèles différents (uniquement avec les voitures, toutes les images de la dataset PASCAL avec et sans le background), Nous avons obtenu des résultats pas très satisfaisant ou les images générées sont presque carrément différentes. Ensuite pour la deuxième étape nous avons expérimenté le BigGan pré entrainé avec la dataset imageNet ou nous avons changé un peu l'algorithme en modifiant l'espace Latent pour la première phase de cette expérimentation nous avons changé que le bruit ensuit dans la deuxième phase nous avons réussis à changer le vecteur de classe et le bruit afin de comparer les résultats. Nous avons remarqué que la génération d'image après avoir modifié que le bruit en choisissant des classes de qui se ressemble juste un peu, notre algorithme permet de générer des images similaire (ex. chat vers chien) par contre quand nous choisissant deux classes qui sont complètement différentes notre algorithme génère une image complétement dissimilaire à notre image but.

La similarité est vraiment un monde vaste, nous avons réussis à obtenir des résultats qui sont assez satisfaisants, mais il existe d'autres approches. Afin d'améliorer les résultats il faudrait essayer de générer plusieurs exemple en modifiant l'espace latent (bruit, classe, bruit et classe) et cela en changeant l'optimiseur et en changeant aussi les poids des loss sémantique et pixels. A la fin une comparaison devra être faite par des sujets humains sur les images pour choisir les meilleurs poids et le meilleur optimiseur et bien évidement le choix du changement de l'espace latent.

Références

[1] Martin N Hebart, Charles Y Zheng, Francisco Pereira et Chris I Baker. « Revealing the multidimensional mental representations of natural objects underlying human similarity judgements ». In : *Nature Human Behaviour* 4.11 (2020), p. 1173-1185 (page 2).

[2] RT Pramod et SP Arun. « Do computational models differ systematically from human object perception ? » In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, p. 1601-1609 (page 2).

[3] Joseph Scott German et Robert A. Jacobs. « Can machine learning account for human visual object shape similarity judgments ». In : *Vision Research* 167 (2020), p. 87-99 (page 2).

[4] Joshua C Peterson, Joshua T Abbott et Thomas L Griffiths. « Evaluating (and improving) the correspondence between deep neural networks and human representations ». In : *Cognitive science* 42.8 (2018), p. 2648-2669 (page 3).

[5] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin et Jonathan K. Su. « This Looks Like That : Deep Learning for Interpretable Image Recognition ». In : *Advances in Neural Information Processing Systems*. T. 32. 2019, p. 8930-8941 (page 3).

[6] Sunnie SY Kim, Nicholas Kolkin, Jason Salavon et Gregory Shakhnarovich. « Deformable Style Transfer ». In : *arXiv preprint arXiv :2003.11038* (2020) (page 3).

[7] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville et Yoshua Bengio. « Generative adversarial networks ». In : *arXiv preprint arXiv :1406.2661* (2014) (page 4).

[8] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou et Alexei A Efros. « Image-to-image translation with conditional adversarial networks ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 1125-1134 (page 6).

[9] Andrew Brock, Jeff Donahue et Karen Simonyan. « Large scale GAN training for high fidelity natural image synthesis ». In : *arXiv preprint arXiv :1809.11096* (2018) (page 7).

[10] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen et Sylvain Paris. « Ganspace : Discovering interpretable gan controls ». In : *arXiv preprint arXiv :2004.02546* (2020) (page 7).

Annexe A

Résultat de pix2pix

Veuillez voir Annexe.pdf