



June 2021

Valentin BRAUX-GUILLIN
Michael PILCER



DESCRIPTION

Dans ce rapport nous présentons notre implémentation du papier de recherche introduisant le réseau ArtGAN [1], un réseau génératif adversarial conditionnel servant à générer des images d'art. L'objectif du réseau est d'apprendre à générer des images d'art plus complexes et abstraites que celles créées par les réseaux génératifs traditionnels. L'approche utilisée est de conditionner l'apprentissage à la classe de labels à laquelle appartient l'oeuvre (selon son genre, son style ou l'artiste qui l'a peinte) afin de mimer l'apprentissage humain.

La méthode se base sur celle utilisée pour entraîner un GAN classique, c'est-à-dire l'entraînement en parallèle d'un générateur et d'un discriminateur. Le générateur reçoit en entrée un bruit gaussien ainsi qu'un label représentant la classe à générer. Il produit alors une image simulant une oeuvre d'art de la classe en question. Le discriminateur reçoit en entrée une image, simulée ou réelle, et doit la classifier soit dans un des labels réel, soit dans un label supplémentaire représentant les images fausses.

Le discriminateur est composé d'un *encoder* qui produit un ensemble de features à partir de l'image reçue en entrée (à l'aide d'opérations de convolution, comme dans un réseau convolutif classique), et d'un classifieur appelé *clsNet* qui retourne un label à partir de ces features.

Le générateur est composé d'un réseau appelé *zNet* qui transforme la concaténation du bruit et du label requis en un ensemble de features correspondant à ceux produits par l'*encoder*, et d'un *decoder* qui produit une image à partir de ces features (à l'aide d'opérations de déconvolution).

Pour l'apprentissage des réseaux, quatre pertes différentes sont utilisées. Il y a d'abord une perte d'entropie croisée visant à entraîner le discriminateur à reconnaître les différents labels (avec en entrée les prédictions sur des images réelles et leurs labels), et les pertes classiques du GAN qui mesurent l'entropie croisée entre les prédictions sur les images générées et leurs labels requis (pour le générateur) ou le label faux (pour le discriminateur). À cela s'ajoute une perte L^2 mesurant la distorsion d'une image par le passage par l'*encoder* puis le *decoder*, afin d'augmenter la correspondance entre les deux réseaux et ainsi de rendre l'entraînement plus stable.

IMPLÉMENTATION

Nous avons utilisé les datasets fournis par la bibliothèque WikiArt, qui contiennent environ 80,000 images annotées (toutes pour les styles, environ 75% pour les genres, environ 25% pour les artistes) séparées en 56,000 images pour l'entraînement et 24,000 pour la validation. Nous avons éliminé les images corrompues et transformé les labels en version one-hot, afin de pouvoir directement les utiliser dans les mesures d'entropie croisée. Nous avons également réduit la taille des images à 64 * 64, que nous avons obtenue en calculant la taille d'image produite par le modèle de générateur proposé avec l'évolution des paramètres des tenseurs au fur et à mesure des déconvolutions.

Nous avons ensuite créé notre modèle avec PyTorch, séparé en quatre modules correspondant aux réseaux décrits précédemment. Nous présentons le modèle détaillé de notre implémentation avec les paramètres de taille de *kernel*, de *stride* et de *padding* pour chaque couche de convolution ou de déconvolution dans les figure 1 et 2. Les nombres indiqués au dessus des tenseurs indiquent leur taille (hauteur ou largeur qui sont égales) et ceux en bas correspondent au nombre de *channels*. Nous avons calculé ces paramètres en utilisant la formule donnant les transformations des tailles par convolution ou déconvolution. Nous faisons enfin le choix d'un bruit de taille 100.

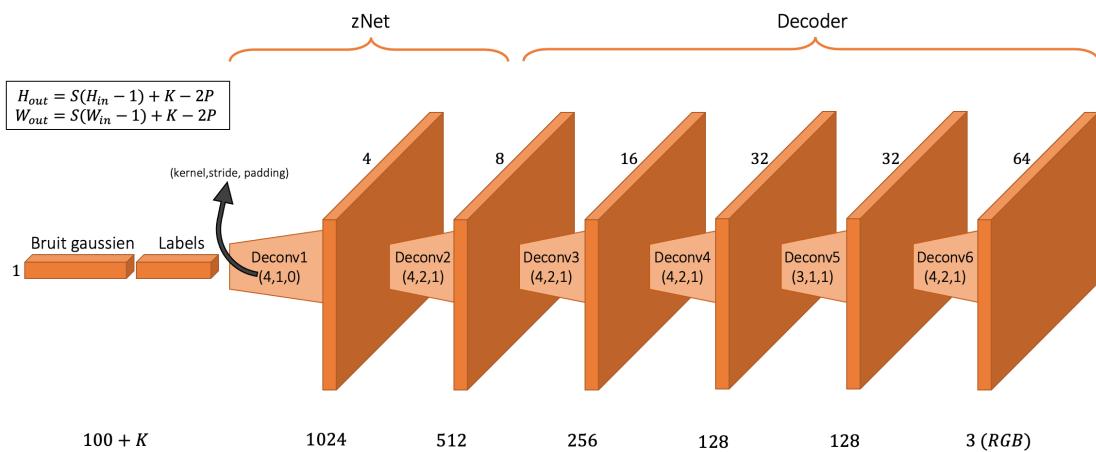


FIG. 1 – Schéma du modèle de générateur.

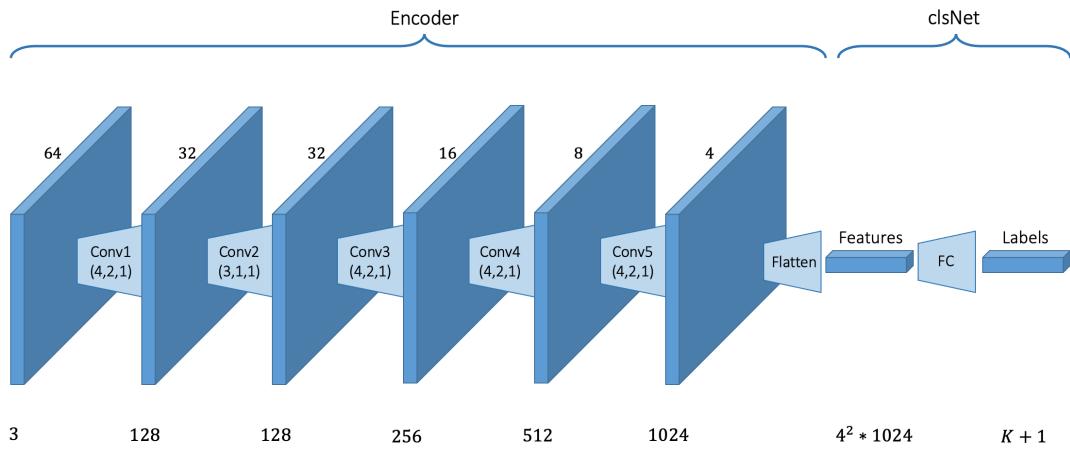


FIG. 2 – Schéma du modèle de discriminateur.

Chaque couche de convolution (à l'exception de la première) et de déconvolution (à l'exception de la dernière) est immédiatement suivie par une couche de *batch normalization* pour rendre le modèle plus stable. Les fonctions d'activation sont de type *ReLU* pour le générateur et *LeakyReLU* pour le discriminateur, à l'exception des sorties qui sont des sigmoïdes.

La taille du vecteur de labels diffère entre le générateur et le discriminateur, puisque le premier prend en entrée un vecteur encodant la classe requise parmi les K possibles mais que le second doit produire une prédiction parmi les K classes possibles ainsi que la classe fausse. Pour l'initialisation des poids des modèles, nous avons choisi l'initialisation de Kaiming pour la couche dense et nous nous sommes inspirés des initialisations du GAN proposé par le tutoriel de PyTorch [2] pour les couches de convolution et de déconvolution.

Nous avons ensuite implémenté la fonction visant à entraîner le modèle. L'optimisation a été réalisée par *batchs* de 128 images. Lors d'une itération de l'algorithme sur un *batch*, nous entraînons d'abord le discriminateur, avec la perte sur la classification des images réelles et celle sur la classification des images générées, puis le générateur avec la perte sur la classification des images générées et celle sur la reconstruction d'une image par une combinaison *encoder-decoder*. Ici nous avons dû modifier légèrement l'algorithme du papier, puisque nous avons rencontré un problème dans la rétropropagation des gradients des pertes. En effet, l'appel de la fonction *backward* sur la perte de reconstruction entraînait une erreur puisqu'elle agit sur l'*encoder* dont les poids ont été modifiés lors de l'entraînement du discriminateur. Pour éviter cela, nous avons simplement généré des images une première fois pour entraîner le discriminateur et une autre pour le générateur.

Comme dans le papier, nous avons utilisé l'optimiseur RMSProp avec un *decay* de 0.9. Nous avons fait plusieurs choix de *learning rates* afin d'obtenir les meilleurs résultats possibles. Après plusieurs essais, nous avons remarqué qu'il était bon de diminuer la *learning rate* plus que ce qui est conseillé dans le papier (où il est diminué une unique fois par un facteur 10 à l'*epoch* 80). En effet à partir d'une trentaine d'*epochs*, la perte du discriminateur stagne mais celle du générateur a tendance à remonter. Nous avons donc opté pour une diminution plus progressive. Nous présentons dans le tableau 1 nos choix de schedules pour les différents modèles. Nous avons de plus remarqué qu'une alternance de petites augmentations et diminutions pouvait être bénéfique pour minimiser les pertes des réseaux.

Epoch	0	25	40	50	60	70	80	100	120	140
Modèle artistes	$1 \cdot 10^{-3}$	$2 \cdot 10^{-4}$	$4 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$4 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-6}$
Modèle genres	$1 \cdot 10^{-3}$	$2 \cdot 10^{-4}$	$4 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$4 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$1 \cdot 10^{-5}$			
Modèle styles	$1 \cdot 10^{-3}$	$2 \cdot 10^{-4}$	$4 \cdot 10^{-5}$	$2 \cdot 10^{-5}$						

TAB. 1 – *Évolution de la learning rate pour les différents modèles.*

Nous avons de plus entraîné en parallèle des modèles avec certains changements par rapport au papier, afin d'observer si nous pouvions en améliorer les performances. Notamment, nous avons entraîné un modèle en rajoutant des couches de *dropout* afin d'augmenter la robustesse des réseaux, un autre où le discriminateur a été pré-entraîné pour déjà connaître les classes réelles dès le début de l'entraînement, et enfin un où les cycles de générateur ont été doublés lorsque le discriminateur devenait trop performant.

Enfin pour l'analyse quantitative des performances, nous avons implémenté la mesure de la log-vraisemblance en utilisant la méthode d'estimation de densité Parzen-window dont il est question dans le papier. Puisque cette partie était très éloignée du thème du projet ou du cours, nous nous sommes inspirés d'une implémentation disponible [3] que nous avons adaptée à nos données.

RÉSULTATS

Pour évaluer la performance de nos modèles au cours de l’entraînement, nous avons affiché toutes les 5 *epochs* l’évolution des pertes du discriminateur et du générateur, ainsi qu’un échantillon d’images générées. Cela nous a permis de repérer rapidement les modèles qui créaient des phénomènes d’*overfit* afin de les rectifier.

Nous avons donc pu choisir la façon finale dont nous allions entraîner les 3 modèles (à créer des images conditionnellement aux genres, aux artistes et aux styles). Le modèle avec double entraînement du générateur, inspiré de papiers que nous avions lu par ailleurs, n’a pas été concluant. Par contre, le modèle avec *dropout* sur les classes d’artistes a donné des résultats sensiblement meilleurs que sa version sans *dropout*, c’est donc celui que nous avons gardé.

De plus, l’ajout d’un pré-entraînement de 10 *epochs* du discriminateur uniquement sur des images réelles avant l’entraînement du GAN complet (d’une durée moyenne de l’ordre de 100 *epochs*) a permis un apprentissage plus rapide, probablement en permettant au discriminateur de se concentrer plus spécifiquement sur la détermination de la classe fausse lors de l’entraînement.

Étant donné les tailles différentes des datasets selon les labels, nous avons obtenu des différences importantes sur les durées des *epochs*, ce qui explique pourquoi nous avons pu en faire jusqu’à 170 pour les artistes contre seulement 60 pour les styles. La durée moyenne d’une *epoch* était de 15 minutes pour les artistes, 49 minutes pour les genres, et 58 minutes pour les styles. Nous avons entraîné les 3 modèles, pendant 90 *epochs* (3,1 jours) pour les genres, 60 *epochs* (2,4 jours) pour les styles, et 170 *epochs* (1,8 jours) pour les artistes.

De façon logique, nous avons obtenu les meilleurs résultats sur le modèle conditionné par les genres. En effet, le dataset des genres était composé de 10 classes contenant chacune environ 4200 images, contre 23 classes d’environ 600 images pour les artistes et 27 classes d’environ 2000 images pour les styles. Nous visualisons l’évolution des pertes du générateur et du discriminateur du modèle des genres sur la figure 3. Les changements de *learning rate* sont bien remarquables.

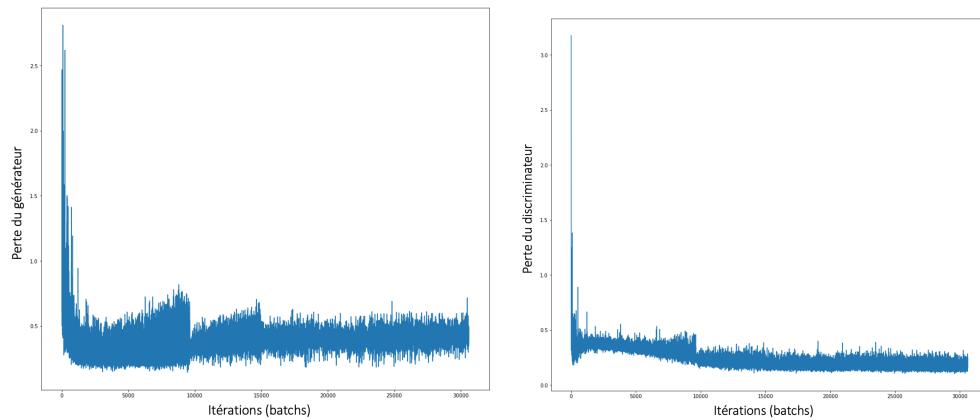


FIG. 3 – Évolution des pertes du générateur et du discriminateur pendant 90 epochs.

Nous présentons les résultats du modèle visant à créer des œuvres conditionnellement aux genres, qui est le modèle qui nous a donné les résultats les plus satisfaisants visuellement, dans la figure 4.

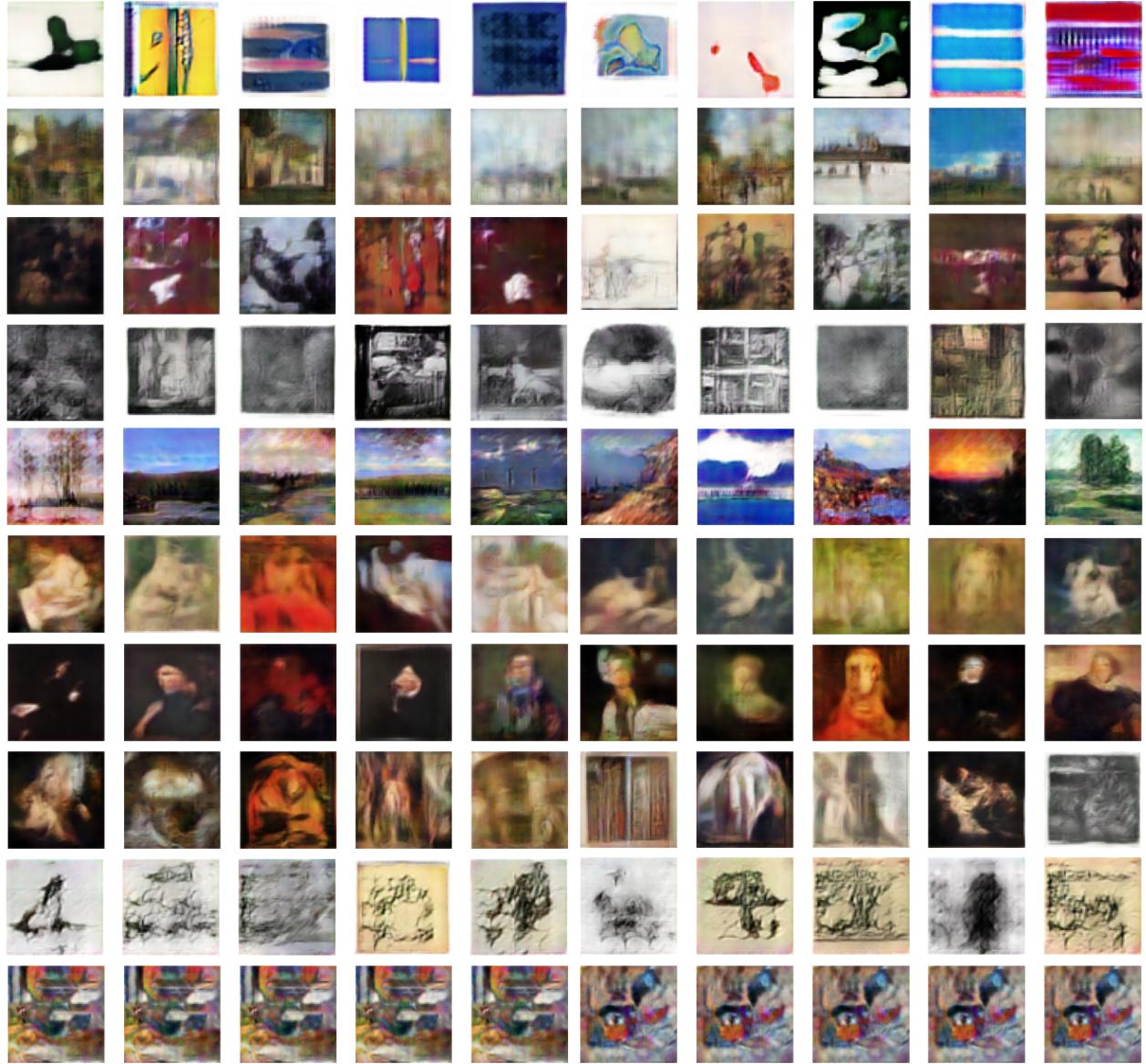


FIG. 4 – Images générées par le GAN conditionnellement aux genres. De haut en bas : (1) Abstract Painting, (2) Cityscape, (3) Genre Painting, (4) Illustration, (5) Landscape, (6) Nude Painting, (7) Portrait, (8) Religious Painting, (9) Sketch and Study, et (10) Still Life.

Nous remarquons que le succès de l'apprentissage est très fortement lié au degré de “généralité” de la classe en question, puisque les classes les mieux générées sont par exemple les paysages, les portraits ou encore l'art abstrait. Les classes où il y a une grande variance dans les œuvres comme la scène de genre ou la peinture religieuse sont ainsi bien plus difficiles à générer de façon satisfaisante. Pour la nature morte, on est face à un phénomène d'*overfit* très important, qui est dû au nombre trop faibles d'œuvres représentatives de la classe.

Nous avons ensuite réalisé une comparaison des images générées à leur plus proche voisin dans la base d'entraînement afin de vérifier que nous n'avions pas réalisé un *overfit* des données, c'est-à-dire que le générateur aurait simplement appris à reproduire certaines œuvres. Nous pouvons observer que ce n'est pas le cas dans la figure 5, où les images du tableau de gauche sont générées par le modèle des genres et celles du tableau de droite par le modèle des artistes.

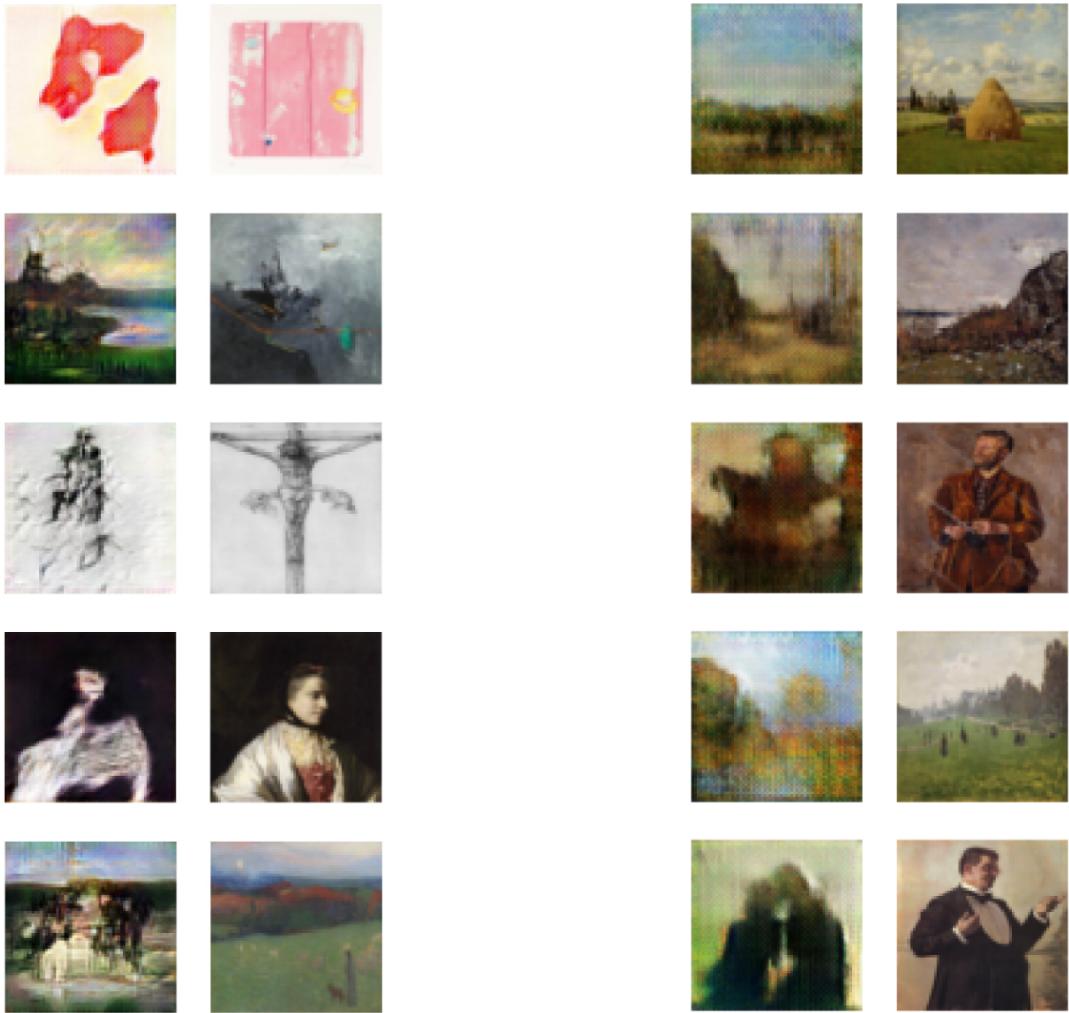


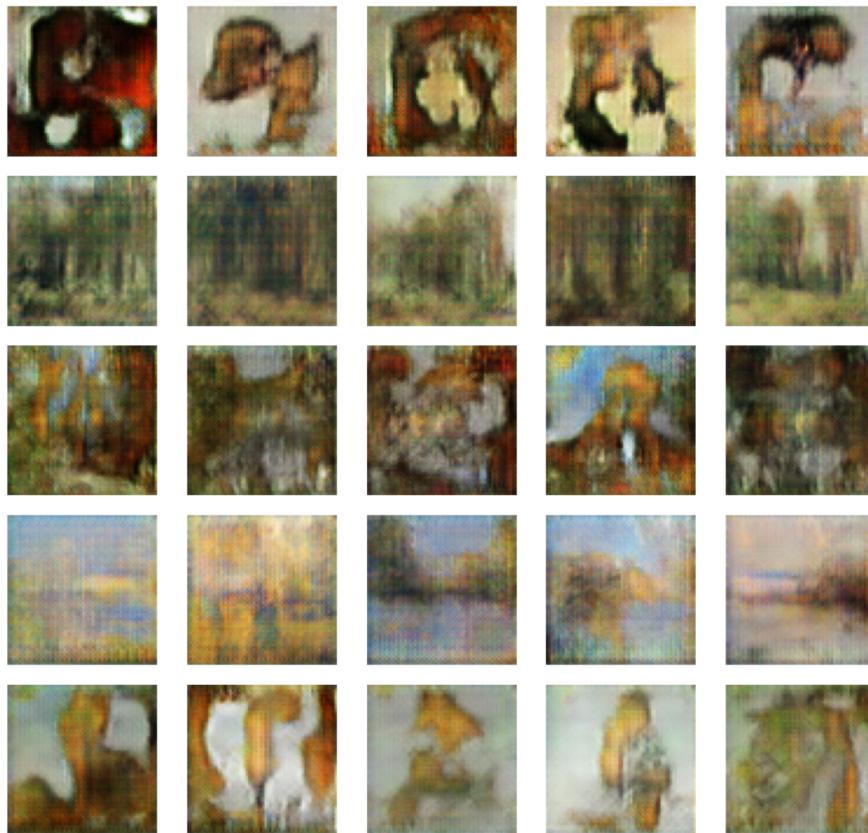
FIG. 5 – Comparaisons d’images générées (gauche) avec leur plus proche voisin (droite).

Nous avons de plus mesuré la métrique de validation proposée dans la papier, l'estimation de densité par Parzen-window. Nous avons trouvé une valeur de log-vraisemblance beaucoup plus faible que celle obtenue théoriquement par les chercheurs du papier. Cependant, le calcul de cette métrique possède un paramètre de bande de passante dont le résultat dépend et qui n'était pas précisé dans le papier. Les valeurs du papier et de notre implémentation sont visibles dans le tableau 2. Cet écart est toutefois probablement assez justifié, puisque notre modèle est moins satisfaisant visuellement que celui du papier.

Modèle	Log-vraisemblance
ArtGAN (papier)	2564 ± 67
ArtGAN (implémentation)	1041 ± 17

TAB. 2 – *Comparaison de mesures de log-vraisemblance avec estimation par Parzen-window.*

Sur les autres modèles, les résultats sont cependant moins bons. Il semble que le modèle n'arrive pas à apprendre à reproduire fidèlement les œuvres des artistes, probablement à cause d'un trop petit nombre d'images par classe couplé à un trop grand nombre de classes. Nous représentons les résultats pour quelques artistes sur la figure 6. Le trop grand nombre de classes rend également difficile la génération d'images satisfaisantes conditionnellement aux styles.

FIG. 6 – *Images générées par le GAN conditionnellement aux artistes. De haut en bas : (1) Pablo Picasso, (2) Ivan Shishkin, (3) Paul Cézanne, (4) Claude Monet, et (5) Salvador Dali.*

CONCLUSION

Ainsi nous avons implémenté un générateur adversarial qui permet de créer des images d’art aux caractéristiques complètes et abstraites à partir de bruits. Nous avons implémenté nos réseaux de neurones sur PyTorch et effectué un entraînement classique des durées de l’ordre de la centaine d’*epochs* à l’aide d’un optimiseur prédéfini. Bien que le modèle n’ait pas fonctionné aussi bien que ce qui est annoncé dans le papier, notamment sur les classes d’artistes et de styles, nous avons obtenu des résultats satisfaisants sur les classes de genres qui démontrent le potentiel du modèle.

Afin de rendre notre modèle plus performant et plus robuste, nous avons vu que la solution du *dropout* sur l’*encoder* et le *decoder* améliorait les performances, le généraliser à l’ensemble des modules du réseau et aux différents conditionnements pourrait donc encore améliorer l’efficacité. Il faudrait de plus optimiser les choix de *learning rate*, en testant une diminution encore plus graduelle et continue, par exemple une fonction décroissante de l’*epoch*. Enfin nous avons remarqué que certaines classes apprenaient plus vite que d’autres, et par exemple que les meilleurs résultats ne se situaient pas aux mêmes *epochs* selon les genres à générer. Il pourrait donc être envisageable d’adapter l’entraînement selon les classes.

Notre modèle reste très général, et pourrait donc s’appliquer à toute autre problématique de génération d’images conditionnellement à certaines caractéristiques. Toutefois, les paramètres que nous avons retenu pour notre entraînement sont probablement très spécifiques à notre tâche et ils devraient donc être adaptés.

RÉFÉRENCES

- [1] Wei Ren Tan, Chee Seng Chan, Hernan E. Aguirre, Kiyoshi Tanaka.
ArtGAN: Artwork Synthesis with Conditional Categorical GANs.
- [2] Nathan Inkawitch. *DCGAN Tutorial.* ([PyTorch.org](https://pytorch.org))
- [3] Nikhil Podila. *GAN-MNIST-CIFAR.* ([GitHub.com](https://github.com))