# ViTGAN: Training GANs with Vision Transformers

Youva ADDAD  Kamel NAIT SLIMANI  Ramdane SALHI
Sorbonne Université

## Introduction

Les réseaux de neurones convolutifs (CNN) dominent aujourd'hui la vision par ordinateur surtout dans la génération d'images (StyleGAN2, BigGAN) grâce à leur puissante capacité de convolution et de pooling. L'intérêt explosif récent pour les transformers a suggéré leur potentiel pour devenir de puissants modèles universels pour les tâches de vision par ordinateur.

Dans ce papier, nous examinons si les ViT peuvent effectuer la tâche de génération d'images, plus précisément, si les ViT peuvent être utilisés pour former des réseaux antagonistes génératifs (GAN) avec une qualité comparable aux GAN basés sur les CNN, le papier propose plusieurs modifications nécessaires pour stabiliser la dynamique d'apprentissage et faciliter la convergence.

## Discriminator

C'est un Vision Transformer (**ViT**) avec des régularisations pour faciliter la convergence.

$$\begin{aligned}
\mathbf{h}_0 &= \left[\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^L\mathbf{E}\right] + \mathbf{E}_{\text{pos}}, \quad \mathbf{E} \in \mathbb{R}^{\left(P^2 \cdot C\right) \times D}, \mathbf{E}_{\text{pos}} \in \mathbb{R}^{(L+1) \times D} \\
\mathbf{h}'_\ell &= \text{MSA}\left(\text{LN}\left(\mathbf{h}_{\ell-1}\right)\right) + \mathbf{h}_{\ell-1}, \qquad \ell = 1, \ldots, L \\
\mathbf{h}_\ell &= \text{MLP}\left(\text{LN}\left(\mathbf{h}'_\ell\right)\right) + \mathbf{h}'_\ell, \qquad \ell = 1, \ldots, L \\
\mathbf{y} &= \text{LN}\left(\mathbf{h}_L^0\right)
\end{aligned} \tag{1}$$

- $\mathbf{X}_{\text{class}}$ : Représente l'Embedding de classification, qui est apprenable et permet de classifier en Real/Fake.
- $\mathbf{E}_{\text{pos}}$ : Représente une projection linéaire de la position du patch (normalisé pour être entre −1,0 et 1,0) suivie d'une fonction d'activation sinusoïdale.
- $\mathbf{MSA}(\mathbf{X}) = \text{concat}_{h=1}^{H}[\text{ Attention }_h(\mathbf{X})]\,\mathbf{W} + \mathbf{b}\,, \quad \text{Attention }_h(\mathbf{X}) = \text{softmax}\left(\frac{\mathbf{Q}\boldsymbol{\kappa}^\top}{\sqrt{d_h}}\right)\mathbf{V}$

## Pré-processing

Pour manipuler les images 2D, une transformation de l'image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ en une séquence de patchs 2D aplatis $\mathbf{x}_p \in \mathbb{R}^{L \times \left(P^2 \cdot C\right)}$ où C'est le nombre de channels, $(P, P)$ est la résolution de chaque patch et $L = HW/P^2$ le nombre de patchs résultant est effectuée.

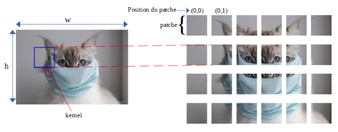



Figure 1. Image vers Patchs.

Pour **extraire les patchs**, une architecture hybride est utilisée, la séquence d'entrée peut être formée à partir des **feature maps** d'un CNN. Cette opération peut être vue comme une **extraction des patchs** suivie d'une **projection** sur une dimension D, elle permet de faire en plus du **overlapping** afin d'éviter au discriminator de l'overfitting (donc une forme de régularisation).

## Stabilisation du Discriminator

Forcer la Lipschitzness du Discriminator : Pour faire respecter lipschitzness du discriminateur, *l'attention L2* remplace le produit scalaire de points par la distance euclidienne. La self-attention L2 est Lipschitz pour $W_q = W_k$ (L2-MHA n'est pas Lipschitz pour $W_q$, $W_k$ arbitraire).

$$\text{Attention }_h(\mathbf{X}) = \text{softmax}\left(-\frac{d\left(\mathbf{X}\mathbf{W}_q, \mathbf{X}\mathbf{W}_k\right)}{\sqrt{d_h}}\right)\mathbf{X}\mathbf{W}_v, \quad \text{Avec} \quad \mathbf{W}_q = \mathbf{W}_k,$$

De plus pour permettre aux transformers de tirer parti du **contexte local** aussi bien que le **contexte global**, des convolutions sont appliquées lors du calcul de Q, K, V tout en gardant le weight tying.

Improved Spectral Normalization : Une technique de normalisation des poids, une grande spectral norm aide à empêcher l'uniformité des tokens a la fin du MSA.
Avec $\bar{W}_{\text{ISN}}(\mathbf{W}) := \sigma\left(\mathbf{W}_{\text{init}}\right) \cdot \mathbf{W}/\sigma(\mathbf{W})$, une version améliorée de la spectral norm.

## Generator

Le générateur lui aussi se compose de ViT et d'un mapping network afin d'injecter le bruit z ainsi que d'un neural implicit representation pour la la génération de couleur:

Mapping network: Une projection non linéaire d'un vecteur de bruit gaussien $z$ par un MLP qui produit un espace latent intermédiaire, et qui contrôle le generator. Il utilise des **Equalized Learning rate** et des **LeakyRelu** (StyleGAN2).

ViT:

$$\begin{aligned}
\mathbf{h}_0 &= \mathbf{E}_{\text{pos}}\,, & \mathbf{E}_{\text{pos}} &\in \mathbb{R}^{L \times D} \\
\mathbf{h}'_\ell &= \text{MSA}\left(\text{SLN}\left(\mathbf{h}_{\ell-1}, \mathbf{w}\right)\right) + \mathbf{h}_{\ell-1}, & \ell 1, \ldots, L, \mathbf{w} &\in \mathbb{R}^D \\
\mathbf{h}_\ell &= \text{MLP}\left(\text{SLN}\left(\mathbf{h}'_\ell, \mathbf{w}\right)\right) + \mathbf{h}'_\ell, & \ell &= 1, \ldots, L \\
\mathbf{y} &= \text{SLN}\left(\mathbf{h}_L, \mathbf{w}\right) = \left[\mathbf{y}^1, \cdots, \mathbf{y}^L\right] & \mathbf{y}^1, \ldots, \mathbf{y}^L &\in \mathbb{R}^D \\
\mathbf{x} &= \left[\mathbf{x}_p^1, \cdots, \mathbf{x}_p^L\right] = \left[f_\theta\left(\mathbf{E}_{\text{fou}}, \mathbf{y}^1\right), \ldots, f_\theta\left(\mathbf{E}_{\text{fou}}, \mathbf{y}^L\right)\right] & \mathbf{x}_p^i &\in \mathbb{R}^{P^2 \times C}, \mathbf{x} \in \mathbb{R}^{H \times W \times C}
\end{aligned}$$

Mapping de sortie : Un MLP basé sur les coordonnées, un mapping entre la coordonnée $\mathbf{v} = (x, y)$ et la valeur de pixel $\mathbf{y} = (\text{R}, \text{G}, \text{B})$ (Implicit neural representation). Chaque positional embedding est une projection linéaire de coordonnées de pixels suivie d'une fonction d'activation sinusoïdale (normalisé pour être entre −1,0 et 1,0) ce qui représente le $\mathbf{E}_{\text{fou}}$ .

## Generator en détail

Self-modulated LayerNorm (SLN) : Permet aux feature maps intermédiaires d'un générateur de changer en fonction du vecteur de bruit d'entrée (Un conditionnement fort sur la sortie basé sur l'espace latent intermédiaire w).

$$\text{SLN}\left(\mathbf{h}_\ell, \mathbf{w}\right) = \text{SLN}\left(\mathbf{h}_\ell, \text{MLP}(\mathbf{z})\right) = \gamma_\ell(\mathbf{w}) \odot \frac{\mathbf{h}_\ell - \boldsymbol{\mu}}{\sigma} + \beta_\ell(\mathbf{w})$$

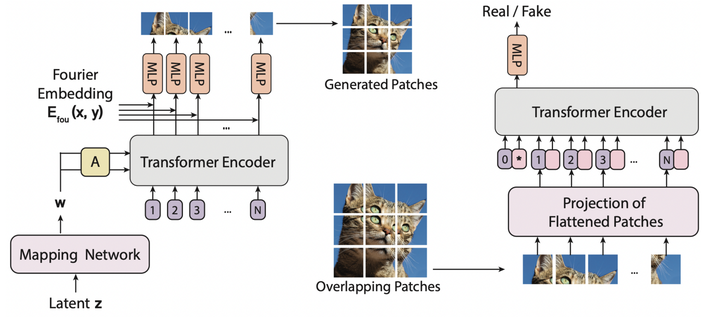
Implicit Neural Representation : Un MLP suivie d'une projection de base de Fourier peut apprendre un signal de grande dimension. $\text{E}_{fou} \in \mathbb{R}^{P^2.D}$, sera conditionné par les $y^i$ via weight modulation. Un Siren où un fourier features network avec 2 couches équipé de weight modulation sera utilisé.

Equalized Learning rate : Utiliser une initialisation triviale $\mathcal{N}(0, 1)$ puis mettre explicitement à l'échelle les poids au moment du forward. L'avantage de le faire dynamiquement plutôt que pendant l'initialisation est quelque peu subtil et concerne l'invariance d'échelle dans les méthodes de descente de gradient stochastique adaptatif couramment utilisées telles que Adam.

Weight Modulation : Un Module Linaire qui permet donc de conditionner $\text{E}_{fou}$ par $y^i$, la modulation ce fait en multipliant $y^i$ par les poids du module en gardant le même principe que Equalized Linear.

$$\hat{W}_{ij} = y_j W_{ij}$$

## Aperçu de l'architecture ViTGAN




## Expérimentation

Dataset et paramètres : Nous avons effectué les expérimentations sur le jeu de données $CIFAR10$.

| $\#epochs$ | $batch\_size$ | $lr_d$ | $lr_g$ | $\beta_1$ | $\beta_2$ | $\lambda_{real}$ | $\lambda_{fake}$ |
|---|---|---|---|---|---|---|---|
| 100 | 32 | 2e-3 | 2e-3 | 0.0 | 0.99 | 10. | 10. |

| $emb\_dim$ | $latent\_dim$ | $\#heads$ | $\#blocks$ | $mlp\_ratio$ | $patch\_size$ |
|---|---|---|---|---|---|
| 300 | 128 | 5 | 3 | 4 | 4 |

Loss et DiffAugment : De plus, nous utilisons la non-saturating logistic loss couplée avec une balanced consistency regularization, qui consiste à utiliser un ensemble d'augmentations $T$ (color, translation, cutout celles que nous avons utilisées), en prenant $T(x)$ et $T(y)$ avec $x \sim P_{data}$ et $y \sim p_g$ comme régularisation.

$$L_D \leftarrow D(G(z)) - D(x) + \lambda_{\text{real}} * |D(x) - D(T(x))\|^2 + \lambda_{\text{fake}} * |D(G(z)) - D(T(G(z)))\|^2, \quad L_G \leftarrow -D(G(z))$$

## Résultats qualitatifs

Les résultats des expérimentations obtenus avec le modèle que nous avons implémenté et les paramètres cités précédemment sur le dataset CIFAR10:

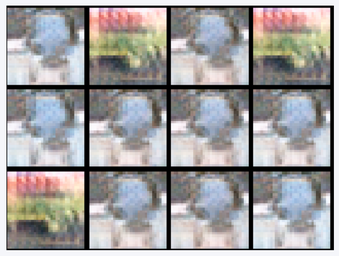


Figure 2. Résultats Qualitatifs CIFAR10 32*32

Les résultats des expérimentations obtenus dans l'article :

| Method | Data Augmentation | Conv | FID ↓ | IS ↑ |
|---|---|---|---|---|
| StyleGAN2 | None | Y | 5.60 | 9.41 |
| StyleGAN2 | DiffAug | Y | 9.89 | 9.40 |
| ViTGAN | None | N | 30.72 | 7.75 |
| ViTGAN | DiffAug | N | 6.66 | 9.30 |
| ViTGAN w/o. bCR | DiffAug | N | 8.84 | 9.02 |

Comparaison avec l'état de l'art sur CIFAR10

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS
DEPUIS 1257

[1] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33:7559–7570, 2020.

[2] Zhengli Zhao, Sameer Singh, Honglak Lee, Zizhao Zhang, Augustus Odena, and Han Zhang. Improved consistency regularization for gans. *arXiv preprint arXiv:2002.04724*, 2020.