UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

REDES ANTAGÓNICAS GENERATIVAS EFICIENTES USANDO ATENCIÓN ADITIVA Y CONVOLUCIONES.

T E S I S

QUE PARA OPTAR POR EL GRADO DE: MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

P R E S E N T A:

EMILIO ALEJANDRO MORALES JUÁREZ

Director de Tesis:

DR. GIBRÁN FUENTES PINEDA

IIMAS, UNAM

Ciudad Universitaria, CDMX. Octubre, 2022

Resumen

Aunque la capacidad de los modelos profundos generativos ha mejorado recientemente, su uso y entrenamiento requiere de una gran capacidad de cómputo. Esto incrementa la huella de carbono, además de limitar el acceso al uso de estos modelos, así como sus aplicaciones. Sin embargo, el desarrollo de modelos eficientes no ha sido de las principales prioridades en la investigación de modelos profundos generativos.

En este trabajo se presentan redes antagónicas generativas eficientes. El modulo principal de estas arquitecturas es el mecanismo de atención aditiva. Esta atención modela dependencias de largo rango para tareas generativas de manera eficiente. Estos modelos consiguen una evaluación competitiva con el estado del arte, reduciendo significativamente el poder de computo requerido en FLOPs.

Agradecimientos

Este trabajo se llevó a cabo gracias a la beca proporcionada por el Consejo Nacional de Ciencia y Tecnología.

Agradezco al Dr. Gibrán Fuentes Pineda por sus enseñanzas y consejos de investigación, por la libertad de explorar ideas, su apoyo y su retroalimentación en este trabajo. Gracias por todas las conversaciones y los interesantes intercambios de ideas, y por hacer que esta experiencia fuera tan interesante y entretenida.

Al Dr. Victor Mijangos, al Dr. Carlos Hernández, a la Dra. Wendy Aguilar y al Dr. Iván Meza, por los valiosos comentarios y correcciones hechos en este trabajo.

A mi papá, por siempre alentarme a ser mejor, por sus consejos, y su siempre incondicional apoyo. Y a mi mamá, por sus consejos y su incondicional amor.

Índice general

Índice de tablas

Índice de figuras

Notación

Esta sección contiene la notación utilizada a lo largo de este documento.

$\mathbb{R}$	Conjunto de los números reales
$[a, b]$	Intervalo real incluyendo a y b
$(a, b]$	Intervalo real excluyendo a pero incluyendo b
x	Un vector
X	Una matriz
$\mathbf{b}$	Un lote
$\mathbf{h}$	Una capa
$\mathrm{X}^\top$	Transpuesta de matriz X
$\mathrm{A} \odot \mathrm{B}$	Producto de Hadamard de matriz A y matriz B
$\frac{\partial y}{\partial x}$	Derivada de y respecto a x
$p(\mathrm{a})$	Una distribución de probabilidad sobre una variable
$\mathrm{a} \sim p$	Variable aleatoria a tiene distribución p
$\mathbb{E}_{x \sim p(x)}[f(x)]$	Valor esperado de $f(x)$ respecto a $p(x)$
$H(\mathcal{X})$	Entropía de Shannon de variable aleatoria $\mathcal{X}$
$KL(P\|Q)$	Divergencia Kullback-Leibler de P y Q
$JSD(P\|Q)$	Divergencia Jansen-Shannon de P y Q
$f : A \to B$	Función f con dominio A y rango B
$\|\|x\|\|_2$	Norma L^2 de x
p_r	Distribución del conjunto de datos reales
p_g	Distribución del conjunto de datos generados

Capítulo 1

Introducción

Las redes generativas antagónicas (GANs, por las siglas en inglés de *generative adverserial networks*) (Goodfellow et al., 2014) son modelos profundos generativos que pueden modelar distribuciones de probabilidad de datos de alta dimensionalidad como imágenes (Sauer et al., 2022; Mokady et al., 2022) y audio (Kumar et al., 2019; Bińkowski et al., 2019; Kong et al., 2020). Sin embargo, entrenar estos modelos para generar nuevas muestras indistinguibles del conjunto de los datos reales es una tarea difícil debido al alto costo computacional requerido. Además, diseñar arquitecturas eficientes de GANs requiere también abordar el problema de la inestabilidad de los entrenamientos que las diferentes funciones objetivo y módulos neuronales pueden ocasionar (Arjovsky and Bottou, 2017; Lee et al., 2021).

Por otro lado, en el área del Procesamiento del Lenguaje Natural (PLN), los Transformadores (Vaswani et al., 2017) se han convertido en el bloque principal de los modelos para resolver tareas desde clasificación de texto hasta modelado del lenguaje (Radford et al., 2018; Raffel et al., 2020; Brown et al., 2020). Uno de los módulos más importantes del Transformador es la autoatención multicabeza (Bahdanau et al., 2014; Vaswani et al., 2017). Este mecanismo de atención ha sido propuesto para aprender dependencias de largo rango (Graves, 2013; Sutskever et al., 2014), lo que ha permitido resolver múltiples tareas modelando secuencias largas (la longitud de secuencia máxima de un modelo T5 (Raffel et al., 2020) es 512). Debido a la efectividad de este modelo, recientes trabajos han tenido como objetivo adaptar los Transformadores en el campo de visión por computadora (Dosovitskiy et al., 2021; Touvron et al., 2021a; Yuan et al., 2021). Una de las principales dificultades de esta tarea es la manera en la que el Transformador procesa las representaciones de los datos. En el caso de las imágenes, procesarlas como secuencias de píxeles no es viable al momento de escalar modelos a imágenes de altas resoluciones (1024×1024). Para resolver esta limitante, ViT (Dosovitskiy et al., 2021) propone procesar las imágenes como una secuencia de parches para la tarea de clasificación de imágenes. Este Transformador de Visión (Tu et al., 2022; Zhai et al., 2022) ha mostrado superar los resultados de las arquitecturas convolucionales de estado del arte (Liu et al., 2022), consiguiendo también mayor capacidad para escalar.

En GANs (Goodfellow et al., 2014), los Transformadores también han mostrado un desempeño competitivo comparado con arquitecturas convolucionales para la tarea de generación de imágenes (Zhang et al., 2019; Park and Kim, 2022; Lee et al.,

2021; Jiang et al., 2021; Zhang et al., 2022; Zhao et al., 2021a). Estos modelos proponen generadores libres de convoluciones para sintetizar imágenes. Sin embargo, estas GANs requieren de múltiples GPUs o TPUs para entrenarse. Además, sus arquitecturas requieren de ingeniería elaborada para reducir su costo computacional y estabilizar los entrenamientos para así conseguir resultados de estado del arte.

Por otro lado, la operación de producto punto de la autoatención es uno de los módulos más importantes al entrenar Transformadores que requieren un gran poder de cómputo. Debido a esto, se han propuesto alternativas en el procesamiento del PLN para reducir el costo computacional (Child et al., 2019; Wang et al., 2020; Kitaev et al., 2020; Zaheer et al., 2020). Asimismo, en el campo la visión por computadora se han empezado a desarrollar mecanismos de atención más eficientes (Liu et al., 2021). Sin embargo, a pesar de que estos mecanismos de atención consiguen resultados similares a la operación de producto punto, entrenar Transformadores tanto para tareas de PLN como para tareas de visión por computadora sigue requiriendo un alto costo computacional. Esto limita el acceso al uso de estos modelos, así como sus aplicaciones. Además, los Transformadores antagónicos generativos eficientes son modelos que han tenido un menor desarrollo en comparación con los modelos propuestos para la clasificación de imágenes y el PLN.

La hipótesis de este trabajo es que utilizando Transformadores de atención aditiva en conjunto con convoluciones en GANs puede ser más eficiente en términos de FLOPs que los modelos de GANs existentes, consiguiendo resultados competitivos con el estado del arte (Brock et al., 2019; Karras et al., 2019, 2020; Lee et al., 2021) utilizando una sola GPU. Esta hipótesis se basa en la mejora en los resultados que los mecanismos de atención han tenido en el PLN y en el campo de visión por computadora al trabajar con grandes cantidades de datos (Radford et al., 2018; Brown et al., 2020; Dosovitskiy et al., 2021). Esta mejora se debe principalmente al aprendizaje de dependencias largas (Bahdanau et al., 2014).

Este trabajo tiene como objetivo diseñar GANs utilizando mecanismos de atención para explotar el aprendizaje de dependencias largas y así conseguir resultados competitivos comparados con los modelos de estado del arte, manteniendo un costo computacional de entrenamiento bajo. En resumen, las contribuciones del presente trabajo son las siguientes:

- Se presenta una arquitectura GAN en la que el generador y el discriminador utilizan Transformadores con atención de complejidad $O(N)$, lo que permite procesar secuencias de gran longitud de manera eficiente en ambos modelos. Para lograr estabilidad durante el entrenamiento, se evalúa el efecto de los diferentes mecanismos de atención de complejidad $O(N)$ con la penalización de gradientes y su compatibilidad con el sesgo inductivo de las convoluciones. El mecanismo de atención de AdatGAN demuestra ser un mecanismo eficiente y compatible con convoluciones para tareas generativas. El generador de AdatGAN requiere, en términos de FLOPs, el 11.53 % de StyleGAN2 (Karras et al., 2020) y el 34.61 % de VITGAN (Lee et al., 2021) para la resolución de 64×64 y $\sim$27.8 % para la resolución de 128×128.

- Se realiza una exhaustiva evaluación del modelo propuesto en diferentes resoluciones y conjuntos de datos, como CIFAR-10, CelebA y FFHQ, así como en

datos de gran escala como LSUN Bedroom. Además, se consigue una evaluación FID competitiva con el estado del arte de GANs en CIFAR-10 y CelebA de manera más eficiente.

Capítulo 2

Trabajo relacionado

En este capítulo se describe la investigación sobre la que se construye este trabajo. Primero, se mencionan las propuestas más importantes de atención eficiente. Este módulo será el principal de la arquitectura GAN. Después, se mencionan las arquitecturas de GANs, así como sus dificultades de entrenamiento, las cuales se abordarán con diversas estrategias como el uso de Transformadores con convoluciones y la compatibilidad de los mecanismos de atención con la penalización de gradientes.

2.1. Atención eficiente

En el campo del PLN se han propuesto alternativas para reducir la complejidad cuadrática de la autoatención original. Por ejemplo, Sparse-Transformer (Child et al., 2019) propone una factorización de la matriz de atención. Longformer (Beltagy et al., 2020) agrega ventanas dilatadas desplazables. Linformer (Wang et al., 2020) reduce la longitud de la matriz de *key* y *value* proyectándolas en una menor dimensión. Reformer (Kitaev et al., 2020) utiliza *Locally-sensitive hashing* y Big-Bird (Zaheer et al., 2020) propone un aproximador universal de secuencias. En el campo de la visión por computadora, propuestas eficientes son Swin Transformer (Liu et al., 2021). Este esquema limita la atención a ventanas locales. Por otro lado, CvT (Wu et al., 2021b) reduce la longitud de la matriz *key* y *value* utilizando convoluciones separables en profundidad. LS Transformer (Zhu et al., 2021) es un enfoque que modela correlaciones locales y distantes diseñado para tareas de visión por computadora y PLN. Aunque estas propuestas reducen el costo computacional requerido por la autoatención original, algunas de sus implementaciones son demasiado elaboradas y no logran una complejidad lineal.

2.2. Arquitecturas de GANs

En el marco de trabajo de GANs (Goodfellow et al., 2014), se han propuesto arquitecturas que consisten únicamente de Transformadores. Estas arquitecturas consiguen resultados competitivos comparados con modelos convolucionales de estado del arte como BigGAN (Brock et al., 2019) y StyleGANs (Karras et al., 2019, 2020). TransGAN (Jiang et al., 2021) fue la primer propuesta, seguida de VITGAN (Lee

et al., 2021). TransGAN consiste únicamente de bloques de Transformador y utiliza penalización de gradientes (Arjovsky et al., 2017; Gulrajani et al., 2017) para estabilizar el entrenamiento del discriminador Transformador. Aborda la limitación cuadrática en altas resoluciones utilizando *grid self-attention*, consiguiendo mejores resultados que autoatención de Nyström (Xiong et al., 2021). Por otro lado, VITGAN genera parches, lo que reduce la longitud de la secuencia de salida del Transformador. Además, este modelo emplea representaciones neuronales implícitas (Anokhin et al., 2021), y propone una modificación a la normalización espectral original (Miyato et al., 2018) para estabilizar el discriminador Transformador. Sin embargo, estas arquitecturas requieren más de una GPU para poder entrenarse. TransGAN se entrena en 16 GPUs V100. VITGAN se entrena en TPU y requiere de 52.1B de FLOPs para imágenes de 256×256.

Modelos con mayor capacidad como HiT (Zhao et al., 2021a) y StyleSwin (Zhang et al., 2022) proponen un generador Transformador. Estos modelos trabajan utilizando el discriminador de StyleGAN2 (Karras et al., 2020) debido a la dificultad de adaptar Transformadores en el discriminador. HiT aborda la complejidad cuadrática utilizando *multi-axis blocked self-attention*, y elimina la autoatención en las etapas de alta resolución. Por otro lado, el bloque principal de StyleSwin consiste en Swin Transformer. Sin embargo, estas arquitecturas además de no aprovechar los Transformadores en el discriminador, están diseñadas únicamente para altas resoluciones, por lo que su costo computación no es viable para una sola GPU. StyleSwin se entrena en 8 GPUs V100 de 32 GB, lo que permite un tamaño de lote de 32 para imágenes de 256×256 y 16 para 1024×1024. Este modelo requiere 50.90B de FLOPs para 1024×1024, lo que es menos que los 74.27B requeridos por StyleGAN2. HiT se entrena en TPU.

Por otro lado, GANsformer (Hudson and Zitnick, 2021) propone un grafo bipartito, y a diferencia de los Transformadores GANs anteriores, utiliza convoluciones y autoatención en conjunto. Esta arquitectura resulta en una generalización de StyleGAN, por lo que no aprovecha por completo la capacidad de los Transformadores. No obstante, aunque el uso de convoluciones y Transformadores ha mostrado mejorar las arquitecturas neuronales en tareas de clasificación de imágenes (Touvron et al., 2021b; Wu et al., 2021b; Park and Kim, 2021), este método ha sido poco explorado en la síntesis de imágenes con Transformadores GANs.

Capítulo 3

Redes generativas antagónicas

Las GANs (Goodfellow et al., 2014) consisten en una estrategia de entrenamiento entre dos modelos. Este marco de trabajo se puede utilizar para modelar la distribución de probabilidad de un conjunto de datos reales dado. En las siguientes secciones se explica la función objetivo del entrenamiento así como las arquitecturas de los modelos.

3.1. Información y entropía

Originalmente propuesta como una teoría de la comunicación (Shannon, 1948), dada una variable aleatoria $\mathcal{X}$ cuyos posibles valores están expresados por x, la información $I(x)$ de un evento se define como:

$$I(x) = \log \frac{1}{p(x)} \tag{3.1}$$

donde $p(x)$ es la función de masa de probabilidad que mapea el estado de $\mathcal{X} = x$ a su probabilidad. El dominio de p son todos los posibles valores de $\mathcal{X}$ y $\sum_{x \in \mathcal{X}} p(x) = 1$. Una generalización de la medida de información es le entropía, la cual se define como el valor esperado de la información de una variable aleatoria $\mathcal{X}$ discreta:

$$H(\mathcal{X}) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{3.2}$$

La unidad de medida de la entropía son nats cuando se usa el logaritmo natural, y bits cuando se usa el logaritmo base 2.

3.1.1. Divergencias Kullback-Leibler y Jensen-Shannon

Una medida estadística entre una distribución de probabilidad comparada con otra es una divergencia.

Divergencia Kullback-Leibler Si se tienen dos distribuciones de probabilidad $p_g(\mathcal{X})$ y $p_r(\mathcal{X})$ sobre la misma variable $\mathcal{X}$, la divergencia Kullback-Leibler (también escrita como divergencia KL) calcula la divergencia entre dos distribuciones p_r

y p_g como el valor esperado de la diferencia logarítmica. Para distribuciones de probabilidad discretas:

$$KL(p_r\|p_g) = \sum_{x\in\mathcal{X}} p_r(x)\log\frac{p_r(x)}{p_g(x)}. \tag{3.3}$$

Para distribuciones de una variable aleatoria continua:

$$KL(p_r\|p_g) = \int_x p_r(x)\log\frac{p_r(x)}{p_g(x)}dx. \tag{3.4}$$

Algunas propiedades importantes son que esta divergencia no es negativa $KL(p_g\|p_r) \geq 0$, y no es simétrica, por lo que $KL(p_r\|p_g) \neq KL(p_g\|p_r)$. Además, si la distribución p_g es igual a p_r, $KL(p_r\|p_g) = 0$.

Divergencia Jansen-Shannon La divergencia Jansen Shannon (JSD, por las siglas en inglés de *Jansen Shannon divergence*) se define como:

$$JSD(p_r\|p_g) = \frac{1}{2}KL(p_r\|p_m) + \frac{1}{2}KL(p_g\|p_m) \tag{3.5}$$

donde p_m es $(p_r + p_g)/2$. A diferencia de la divergencia KL, JSD es simétrica.

3.2. Funciones objetivo y regularización de pesos y gradientes

En el entrenamiento de GANs (Goodfellow et al., 2014), se define p_g con un generador $G(z)$ que transforma una variable latente $G : \mathcal{Z} \to \mathcal{X}$, y un discriminador $D(x)$ que es optimizado para distinguir si una instancia es generada por G o pertenece al conjunto de datos reales. Ambos G y D son redes neuronales parametrizadas. El aprendizaje de los parámetros de cada modelo es un juego *minimax* con la función de valor:

$$\min_G \max_D V(D,G) \tag{3.6}$$

$$V(G,D) = \mathbb{E}_{x\sim p_{\mathrm{r}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.7}$$

donde G modela la distribución de los datos reales p_{r}, y z es muestreada de una distribución p_z. El objetivo de G es engañar a D al aproximar p_r, donde D regresa como salida un escalar entre $[0, 1]$. Escribiendo en 3.7 la definición del valor esperado en términos de x, donde p_g es la distribución de las instancias generadas:

$$\begin{aligned} V(G,D) &= \int_x p_{\mathrm{r}}(x)\log(D(x))dx + \int_z p_z(z)\log(1 - D(G(z)))dz \\ &= \int_x p_{\mathrm{r}}(x)\log(D(x))dx + \int_x p_g(x)\log(1 - D(x))dx \\ &= \int_x p_{\mathrm{r}}(x)\log(D(x)) + p_g(x)\log(1 - D(x))dx \end{aligned} \tag{3.8}$$

el discriminador óptimo D^* derivando:

$$\frac{\partial}{\partial D(x)}(p_{\mathrm{r}}(x)\log(D(x)) + p_g(x)\log(1 - D(x))) = 0 \tag{3.9}$$

es:

$$D^* = \frac{p_r(x)}{p_r(x) + p_g(x)} \tag{3.10}$$

remplazando D^* en 3.8 y escribiendo la expresión como una divergencia:

$$\begin{aligned}
\max_D V(G, D^*) &= \int_x p_{\mathrm{r}}(x)\log(\frac{p_r(x)}{p_r(x) + p_g(x)}) + p_g(x)\log(\frac{p_g(x)}{p_r(x) + p_g(x)})dx \\
&= -\log(4) + \int_x p_{\mathrm{r}}(x)\log(\frac{p_r(x)}{\frac{p_r(x)+p_g(x)}{2}}) + p_g(x)\log(\frac{p_g(x)}{\frac{p_r(x)+p_g(x)}{2}})dx \\
&= -\log(4) + KL\left(p_r \middle\| \frac{p_{\mathrm{r}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_r + p_g}{2}\right) \\
&= -\log(4) + 2 \cdot JSD\left(p_r \| p_g\right).
\end{aligned} \tag{3.11}$$

Por lo tanto, si D es el óptimo tal que puede distinguir perfectamente qué instancias pertenecen a p_r y qué instancias pertenecen a p_g, G minimiza JSD. Si $JSD\left(p_r \| p_g\right) = 0$, entonces $p_r = p_g$. En otras palabras, el modelo aprendió la distribución de probabilidad de los datos reales, por lo que $V(G^*, D^*) = -\log(4)$. El procedimiento original para entrenar GANs se muestra en el Algoritmo 1. En la práctica, el generador trata de maximizar $\log\left(D(G(z))\right)$ en lugar de minimizar $\log\left(1 - D(G(z))\right)$. Esta modificación resulta en la función objetivo no saturada, la cual tiene un mejor comportamiento de gradientes.

3.2.1. Wasserstein GAN

En la práctica, los entrenamientos de GANs son difíciles de estabilizar ya que los gradientes de JSD no proveen suficiente información al generador cuando los soportes de p_r y p_g son disjuntos (Arjovsky and Bottou, 2017). Aunque existen elaboradas técnicas para mejorar estos modelos (Salimans et al., 2016), una propuesta diferente ha sido modificar JSD por la distancia Wasserstein-1 (Arjovsky et al., 2017), también llamada *Earth Mover's distance* (EM):

$$W(p_r, p_g) = \inf_{\gamma \in \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right], \tag{3.12}$$

donde $\Pi(p_r, p_g)$ denota el conjunto de todas las distribuciones conjuntas con marginales p_r y p_g. Esta distancia es equivalente al costo mínimo requerido de transporte de masa para transformar la distribución p_g a p_r. Además, es continua y siempre diferenciable, lo que permite entrenar a D hasta que sea óptimo debido a un mejor comportamiento de los gradientes. En WGAN (Arjovsky et al., 2017) se busca al

Algoritmo 1 Algoritmo de entrenamiento original de GAN (Goodfellow et al., 2014)

Require: α es la tasa de aprendizaje, m es el número de elementos por mini-lote y k es el número de iteraciones que se entrena D.

for número de iteraciones **do**
  for $t = 0, ..., k$ **do**
    Muestrea $\{x^{(i)}\}_{i=1}^{m} \sim p_r$ mini-lote de la distribución real
    Muestrea $\{z^{(i)}\}_{i=1}^{m} \sim p_z$ mini-lote de ruido
    $\tilde{x} \leftarrow G(z)$ Genera mini-lote
    $o_r \leftarrow D(x)$ Muestra real
    $o_g \leftarrow D(\tilde{x})$ Muestra generada
    $\mathcal{L}_D \leftarrow \log(o_r) + \log(1 - o_g)$
    $D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$
  end for
  Muestrea $\{z^{(i)}\}_{i=1}^{m} \sim p_z$ mini-lote de ruido
  $\tilde{x} \leftarrow G(z)$ Genera mini-lote
  $o_g \leftarrow D(\tilde{x})$ Muestra generada
  $\mathcal{L}_G \leftarrow \log(o_g)$
  $G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$

discriminador D del conjunto de funciones continuas K-Lipschitz, ya que se utiliza la dualidad Kantorovich-Rubinstein (Villani, 2009), puesto que es intratable calcular el ínfimo sobre todas las posibles distribuciones conjuntas $\Pi(p_r, p_g)$:

$$\min_G \max_{D \in \mathcal{D}} = \mathbb{E}_{x \sim p_r(x)}[D(x)] - \mathbb{E}_{z \sim p_z(z)}[D(G(z))], \tag{3.13}$$

donde $\mathcal{D}$ es el conjunto de funciones 1-Lipschitz.

Función lipschitziana Dados dos espacios métricos (A, d_a) y (B, d_b), se dice que $f : A \to B$ es una función lipschitziana si existe una constante $K \geq$ tal que:

$$d_B(f(a_1) - f(a_2)) \leq K d_A(a_1, a_2) \tag{3.14}$$

para toda $a_1, a_2 \in A$.

Para conseguir que el discriminador pertenezca al espacio de funciones K-Lipschitz, WGAN utiliza recorte de pesos para mantener los parámetros del discriminador dentro del espacio compacto $[-c, c]$. Sin embargo, este método ocasiona que la arquitectura neuronal aprenda funciones extremadamente simples, además de requerir ajustar cuidadosamente el umbral de recorte c para evitar la explosión o desvanecimiento de los gradientes.

Aunque WGAN converge si se entrena hasta conseguir D óptimo, en la práctica D se entrena una cantidad de veces fija por cada actualización de G.

3.2.2. Penalización de gradientes

Para evitar las dificultades del recorte de pesos, WGAN-GP (Gulrajani et al., 2017) utiliza penalización de gradientes para hacer cumplir la condición de Lipschitz:

Algoritmo 2 Algoritmo de entrenamiento de WGAN-GP (Gulrajani et al., 2017)

Require: α es la tasa de aprendizaje, m es el número de elementos por mini-lote y k es el número de iteraciones que se entrena D y λ es el peso de la penalización de gradientes.

for número de iteraciones **do**
  for $t = 0, ..., k$ **do**
    Muestrea $\{x^{(i)}\}_{i=1}^{m} \sim p_r$ mini-lote de la distribución real
    Muestrea $\{z^{(i)}\}_{i=1}^{m} \sim p_z$ mini-lote de ruido
    $\tilde{x} \leftarrow G(z)$ Genera mini-lote
    $o_r \leftarrow D(x)$ Muestra real
    $o_g \leftarrow D(\tilde{x})$ Muestra generada
    $\hat{x} \leftarrow \epsilon x + (1 - \epsilon)\tilde{x}$
    $\mathcal{L}_D \leftarrow o_g - o_r + \lambda(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2$
    $D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$.
  end for
  Muestrea $\{z^{(i)}\}_{i=1}^{m} \sim p_z$ mini-lote de ruido
  $\tilde{x} \leftarrow G(z)$ Genera mini-lote
  $o_g \leftarrow D(\tilde{x})$ Muestra generada
  $\mathcal{L}_G \leftarrow -o_g$
  $G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$

$$\mathcal{L}_D = \mathbb{E}_{z \sim p_z(z)}[D(G(z))] - \mathbb{E}_{x \sim p_r(x)}[D(x)] + \lambda \mathbb{E}_{\hat{x} \sim p_{\hat{x}}(\hat{x})}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]. \quad (3.15)$$

Para entrenar un modelo utilizando penalización de gradientes, únicamente se añade un termino a la función objetivo del discriminador denotada como $\mathcal{L}_D$, donde λ es un peso que establece qué tanto se penaliza la norma de los gradientes. Esto se consigue sobre la interpolación lineal entre las muestras reales y generadas para suavizar el espacio entre las dos distribuciones. El procedimiento de entrenamiento de WGAN-GP se muestra en el Algoritmo 2.

3.2.3. Regularización R1

Aunque la función objetivo de WGAN-GP es efectiva, en la práctica puede funcionar mejor penalizar desde el principio del entrenamiento respecto a p_r. Esto se consigue únicamente tomando en cuenta las muestras reales para la penalización. Este método es llamado Regularización R1 (Mescheder et al., 2018) y experimentos han mostrado mejor convergencia que WGAN-GP. Aquí, la función objetivo del discriminador tiene la siguiente forma:

$$\mathcal{L}_D = \mathbb{E}_{z \sim p_z(z)}[D(G(z))] - \mathbb{E}_{x \sim p_r(x)}[D(x)] + \frac{\lambda}{2} \mathbb{E}_{x \sim p_r(x)}[(\|\nabla_x D(x)\|_2)]. \quad (3.16)$$

El procedimiento de entrenamiento con Regularización R1 se muestra en el Algoritmo 3. Arquitecturas que utilizan este método son StyleGANs (Karras et al., 2019, 2020), HiT (Zhao et al., 2021a) y StyleSwin (Zhang et al., 2022).

Algoritmo 3 Algoritmo de entrenamiento con Regularización R1 (Mescheder et al., 2018)

Require: α es la tasa de aprendizaje, m es el número de elementos por mini-lote y k es el número de iteraciones que se entrena D.

for número de iteraciones **do**
  for $t = 0, ..., k$ **do**
    Muestrea $\{x^{(i)}\}_{i=1}^{m} \sim p_r$ mini-lote de la distribución real
    Muestrea $\{z^{(i)}\}_{i=1}^{m} \sim p_z$ mini-lote de ruido
    $\tilde{x} \leftarrow G(z)$ Genera mini-lote
    $o_r \leftarrow D(x)$ Muestra real
    $o_g \leftarrow D(\tilde{x})$ Muestra generada
    $\mathcal{L}_D \leftarrow \log(o_r) + \log(1 - o_g) + \lambda(\|\nabla_x D(x)\|_2)$
    $D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$
  end for
  Muestrea $\{z^{(i)}\}_{i=1}^{m} \sim p_z$ mini-lote de ruido
  $\tilde{x} \leftarrow G(z)$ Genera mini-lote
  $o_g \leftarrow D(\tilde{x})$ Muestra generada
  $\mathcal{L}_G \leftarrow \log(o_g)$
  $G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$

3.2.4. Normalización espectral

La penalización del gradientes no es el único método para estabilizar GANs. Generalmente, cuando las arquitecturas del discriminador consisten en simples perceptrones o convoluciones (Radford et al., 2016), se puede regularizar la constante de Lipschitz del discriminador utilizando la Normalización Espectral (Miyato et al., 2018):

$$\bar{W}_{\text{SN}}(\text{W}) := \text{W}/\sigma(\text{W}) \tag{3.17}$$

donde $\text{W} \in \mathbb{R}^{m \times n}$ es la matriz de pesos y $\sigma(\text{W})$ es la norma espectral de la matriz W. Este método evita la explosión y el desvanecimiento de los gradientes (Lin et al., 2021), lo que estabiliza el entrenamiento de GANs al controlar la magnitud de los gradientes respecto a los parámetros. Además, a diferencia de WGAN-GP, requiere menor costo computacional, sin embargo, en bloques modulados (Karras et al., 2019) o de atención (Lee et al., 2021) no es efectivo o requiere ajustes adicionales para funcionar.

Modelos que utilizan la normalización espectral tanto en el generador como el discriminador son FastGAN (Liu et al., 2020). En esta arquitectura, se adopta la función de pérdida *hinge* (Lim and Ye, 2017) donde:

$$\begin{aligned} \mathcal{L}_D = & -\mathbb{E}_{x \sim p_{\text{r}}(x)}[\text{mín}(0, -1 + D(x))] \\ & -\mathbb{E}_{z \sim p_{\text{z}}(z)}[\text{mín}(0, -1 - D(G(z)))] \\ \mathcal{L}_G = & -\mathbb{E}_{z \sim p_{\text{z}}(z)}[D(G(z))]. \end{aligned} \tag{3.18}$$

3.3. Normalización y modulación

Además de las funciones objetivo, la normalización de las representaciones es uno de los elementos más importantes en las arquitecturas de redes neuronales. En las redes generativas, la normalización por lotes (Ioffe and Szegedy, 2015) o alguna variante son esenciales para estabilizar el discriminador y propagar la variable latente en el generador. En las siguientes secciones se sigue la notación de Huang and Belongie (2017). Cabe señalar que en las ecuaciones de normalización, la división se realiza elemento a elemento.

3.3.1. Normalización por lotes

La normalización por lotes (Ioffe and Szegedy, 2015) consiste en normalizar los lotes de salida $\mathbf{b} \in \mathbb{R}^{N \times C \times H \times W}$ de una capa ℓ:

$$\mathrm{BN}(\mathbf{b}_\ell) = \boldsymbol{\gamma}_\ell \odot \left(\frac{\mathbf{b}_\ell - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \right) + \boldsymbol{\beta}_\ell \tag{3.19}$$

donde $\boldsymbol{\gamma}_\ell, \boldsymbol{\beta}_\ell \in \mathbb{R}^C$ son parámetros que se entrenan y $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^C$ son la media y la desviación estándar de los lotes:

$$\boldsymbol{\mu} = \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{b}_\ell^{nchw} \tag{3.20}$$

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} (\mathbf{b}_\ell^{nchw} - \boldsymbol{\mu})^2 + \epsilon}. \tag{3.21}$$

3.3.2. Normalización de instancia

A diferencia de la normalización por lotes, la normalización de instancia (Ulyanov et al., 2016) igualmente calcula estadísticas para normalizar las salidas de la capa:

$$\mathrm{IN}(\mathbf{b}_\ell) = \boldsymbol{\gamma}_\ell \odot \left(\frac{\mathbf{b}_\ell - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \right) + \boldsymbol{\beta}_\ell. \tag{3.22}$$

Sin embargo, aquí $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^{N \times C}$ se calculan para cada elemento del lote y para cada canal:

$$\boldsymbol{\mu} = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{b}_\ell^{nchw} \tag{3.23}$$

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} (\mathbf{b}_\ell^{nchw} - \boldsymbol{\mu})^2 + \epsilon}. \tag{3.24}$$

3.3.3. Normalización de instancia adaptable

En la instancia adaptable (Huang and Belongie, 2017), las estadísticas de cada elemento $\mathbf{b}_\ell \in \mathbb{R}^{N\times C\times H\times W}$ están condicionadas en las estadísticas $\boldsymbol{\mu}'$ y $\boldsymbol{\sigma}'$ de otra capa $\mathbf{b}'_\ell \in \mathbb{R}^{N\times C\times H\times W}$:

$$\text{AdaIN}(\mathbf{b}_\ell, \mathbf{b}'_\ell) = \boldsymbol{\sigma}' \odot \left(\frac{\mathbf{b}_\ell - \boldsymbol{\mu}}{\boldsymbol{\sigma}}\right) + \boldsymbol{\mu}'. \tag{3.25}$$

Aunque originalmente este método es propuesto para acelerar el proceso de transferencia de estilo (Gatys et al., 2015; Johnson et al., 2016), se puede generalizar para adaptarse a otras tareas. La automodulación en GANs es similar a AdaIN.

3.3.4. Automodulación

La manera en en la que el vector latente z se propaga a través de la red generadora puede mejorar significativamente el modelado de p_r. A diferencia de AdaIN, las estadísticas para una capa $\mathbf{h}$ dependen de la variable latente z; de esta manera, se define la modulación (Chen et al., 2019a):

$$\text{SM}(\mathbf{h}_\ell, \text{z}) = \boldsymbol{\gamma}_\ell(\text{z}) \odot \left(\frac{\mathbf{h}_\ell - \boldsymbol{\mu}}{\boldsymbol{\sigma}}\right) + \boldsymbol{\beta}_\ell(\text{z}), \tag{3.26}$$

donde $\boldsymbol{\mu}$ y $\boldsymbol{\sigma}$ son vectores. Aquí, se utiliza un perceptrón multicapa con activación ReLU para calcular $\boldsymbol{\gamma}_\ell(\cdot)$ y $\boldsymbol{\beta}_\ell(\cdot)$ utilizando dos matrices $\text{U}^{(\ell)}$ y $\text{V}^{(\ell)}$, y un vector de sesgo $\text{b}^{(\ell)}$:

$$\boldsymbol{\gamma}_\ell(\text{z}) = \text{V}^{(\ell)}\, \text{máx}(0, \text{U}^{(\ell)}\text{z} + \text{b}^{(\ell)}).$$

3.4. Atención

Aunque existe una gran variedad de GANs cuyas arquitecturas consiste principalmente en convoluciones (Radford et al., 2016; Brock et al., 2019; Karras et al., 2019, 2020), su sesgo inductivo limita el aprendizaje de dependencias largas. Por otro lado, en el campo del PLN, los modelos *secuencia a secuencia* (Sutskever et al., 2014) comenzaron a abordar el problema de dependencias largas. Primero, equipando redes recurrentes (Hochreiter and Schmidhuber, 1997) con mecanismos de atención (Bahdanau et al., 2014; Luong et al., 2015) para aliviar el desvanecimiento de gradientes (Pascanu et al., 2013). Después, diseñando arquitecturas utilizando únicamente atención, como los Transformadores (Vaswani et al., 2017). Debido a la efectividad de esta arquitectura (Radford et al., 2018; Raffel et al., 2020; Brown et al., 2020), los modelos de atención comenzaron a ser adoptados en el campo de visión por computadora (Dosovitskiy et al., 2021).

3.4.1. Autoatención producto punto

La arquitectura que ha conseguido superar a las redes recurrentes en el PLN y a las redes convolucionales en el campo de visión por computadora es el Transformador (Vaswani et al., 2017). El módulo principal de este bloque es la atención producto

punto de múltiples cabezas (MSA, por las siglas en inglés de *multi-headed self-attention*):

$$\text{MSA}(\text{X}) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)\text{W}^O, \tag{3.27}$$

Este mecanismo recibe como entrada la matriz $\text{X} \in \mathbb{R}^{N \times d}$, donde la longitud secuencia es N, la dimensión del *embedding* es d, y el número de cabezales es h:

$$\begin{aligned} \text{head}_i &= \text{Attention}(\text{XW}_i^Q, \text{XW}_i^K, \mathring{\text{X}}\text{W}_i^V) \\ &= \text{softmax}\left(\frac{\text{XW}_i^Q(\text{XW}_i^K)^T}{\sqrt{d_h}}\right)\text{XW}_i^V \end{aligned} \tag{3.28}$$

donde $\text{W}_i^Q, \text{W}_i^K, \text{W}_i^V \in \mathbb{R}^{d \times d_h}$ son parámetros del modelo que se entrenan y $d_h = d/h$. Como se muestra en la ecuación 3.28, la complejidad computacional es cuadrática respecto a la longitud de la secuencia de entrada N, debido a que se requiere de la multiplicación de dos matrices $N \times d_h$, lo que resulta en $O(N^2)$ en tiempo y memoria. Esta es la principal limitante en el uso de los Transformadores para procesamiento de secuencias largas de manera eficiente.

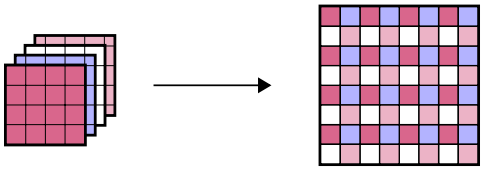


Figura 3.1: Pixel shuffle.

A pesar del costo computacional requerido de la autoatención producto punto, se han realizado trabajos que han adaptado este método en GANs como SAGAN (Zhang et al., 2019). En este modelo convolucional, las matrices $\text{Q} = \text{XW}_i^Q$, $\text{K} = \text{XW}_i^K$ y $\text{V} = \text{XW}_i^V$ son mapas de características generados mediante convoluciones. Más tarde, TransGAN (Jiang et al., 2021) realizó un estudio piloto proponiendo discriminadores y generadores empleando únicamente Transformadores. Esta arquitectura utiliza *pixel shuffle* (Shi et al., 2016) para progresivamente aumentar la longitud de la secuencia de los Transformadores. El entrenamiento en esta GAN se estabiliza utilizando WGAN-GP (Gulrajani et al., 2017). Sin embargo, a pesar de que este enfoque aborda el costo computacional de la operación producto punto en altas resoluciones, limitando la atención a ventanas locales utilizando *grid self-attention*, el modelo aún requiere una gran cantidad de poder computacional debido a que la arquitectura consiste en una pila profunda de Transformadores y las secuencias son generadas a nivel píxel (para resolución de 256 se generan secuencias de longitud 256×256). La operación *pixel shuffle* se muestra en la figura 3.1.

3.4.2. Transformador de Visión

Por otro lado, el Transformador de Visión (ViT, por las siglas en inglés de *Vision Transformer*) (Dosovitskiy et al., 2021), a diferencia de los enfoques anteriores de autoatención, propone procesar las imágenes como secuencias de parches

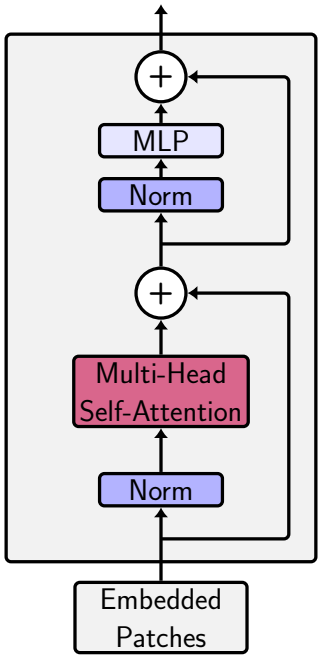


Figura 3.2: Transformador de ViT.

$[\mathrm{x}_1, \ldots, \mathrm{x}_N] \in \mathbb{R}^{N \times (P^2 \cdot C)}$ utilizando una proyección lineal $\mathrm{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}$, donde $N = \frac{H \times W}{P^2}$ es la longitud de secuencia, y $P^2 \times C$ es la dimensión de cada parche.

$$\mathbf{h}_0 = [\mathrm{x}_{\text{class}}, \mathrm{x}_1\mathrm{E}, \ldots, \mathrm{x}_N\mathrm{E}] + \mathrm{E}_{pos}, \qquad \mathrm{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{3.29}$$

$$\mathbf{h}'_\ell = \mathrm{MSA}(\mathrm{LN}(\mathbf{h}_{\ell-1})) + \mathbf{h}_{\ell-1}, \qquad \ell = 1, \ldots, L \tag{3.30}$$

$$\mathbf{h}_\ell = \mathrm{MLP}(\mathrm{LN}(\mathbf{h}'_\ell)) + \mathbf{h}'_\ell, \qquad \ell = 1, \ldots, L \tag{3.31}$$

$$\mathrm{y} = \mathrm{LN}(\mathbf{h}_L^0) \tag{3.32}$$

Aquí, MLP denota el perceptrón multicapa y LN (Ba et al., 2016) la normalización de capa. Al igual que en BERT (Devlin et al., 2019), se agrega un *embedding* de clase $\mathrm{x}_{\text{class}}$ con su *embedding* de posición E_{pos}. La figura 3.2 muestra un bloque de ViT. Este modelo fue adaptado por primera vez en GANs en VITGAN (Lee et al., 2021), consiguiendo resultados competitivos en comparación con GANs convolucionales y procesando secuencias de menor longitud que TransGAN. El generador de VITGAN se define como:

$$\mathbf{h}_0 = \mathrm{E}_{pos}, \qquad \mathrm{E}_{pos} \in \mathbb{R}^{N \times D}, \tag{3.33}$$

$$\mathbf{h}'_\ell = \mathrm{MSA}(\mathrm{SLN}(\mathbf{h}_{\ell-1}, \mathrm{w})) + \mathbf{h}_{\ell-1}, \qquad \ell = 1, \ldots, L, \mathrm{w} \in \mathbb{R}^D \tag{3.34}$$

$$\mathbf{h}_\ell = \mathrm{MLP}(\mathrm{SLN}(\mathbf{h}'_\ell, \mathrm{w})) + \mathbf{h}'_\ell, \qquad \ell = 1, \ldots, L \tag{3.35}$$

$$\mathrm{Y} = \mathrm{SLN}(\mathbf{h}_L, \mathrm{w}) = [\mathrm{y}_1, \ldots, \mathrm{y}_N] \qquad \mathrm{y}_1, \ldots, \mathrm{y}_N \in \mathbb{R}^D \tag{3.36}$$

$$\mathrm{X} = [\mathrm{x}_1, \ldots, \mathrm{x}_N] = [f_\theta(\mathrm{E}_{fou}, \mathrm{y}_1), \ldots, f_\theta(\mathrm{E}_{fou}, \mathrm{y}_N)] \qquad \mathrm{x}_i \in \mathbb{R}^{P^2 \times C}, \mathrm{X} \in \mathbb{R}^{H \times W \times C} \tag{3.37}$$

Aquí, SLN denota la normalización de capa modulada. A diferencia de TransGAN, este generador utiliza una red de mapeo (Karras et al., 2019) que consiste en múltiples MLPs para generar w a partir de z, tal que w = MLP(z). Sin embargo, VITGAN requiere de *embeddings* de Fourier, denotados como E_{fou}, para poder aproximar correctamente p_r con generación de parches. Para la definición de E_{fou}, consultar la sección 3.5.

3.4.3. Mecanismos de atención eficientes

La autoatención puede modelar y generar secuencias largas de forma muy efectiva debido a su aprendizaje de dependencias de largo rango (Radford et al., 2018; Brown et al., 2020). Sin embargo, la principal limitante para aprovechar esta capacidad es su complejidad cuadrática. En consecuencia, se han diseñado mecanismos eficientes para abordar este problema (Tay et al., 2020).

Aproximación de bajo rango Linformer (Wang et al., 2020) utiliza la aproximación de bajo rango para hacer la autoatención $O(N)$. Esto se consigue agregando al modelo dos matrices de proyección $\mathrm{E}_i, \mathrm{F}_i \in \mathbb{R}^{k \times N}$:

$$\begin{aligned} \mathrm{head}_i &= \mathrm{Attention}(\mathrm{X}\mathrm{W}_i^Q, \mathrm{E}_i\mathrm{X}\mathrm{W}_i^K, \mathrm{F}_i\mathrm{X}\mathrm{W}_i^V) \\ &= \mathrm{softmax}\left(\frac{\mathrm{X}\mathrm{W}_i^Q(\mathrm{E}_i\mathrm{X}\mathrm{W}_i^K)^T}{\sqrt{d_h}}\right)\mathrm{F}_i\mathrm{X}\mathrm{W}_i^V. \end{aligned} \tag{3.38}$$

De esta manera, las matrices $\mathrm{X}\mathrm{W}_i^K$ y $\mathrm{X}\mathrm{W}_i^V$ son proyectadas de $(N \times d_h)$ a $(k \times d_h)$. Este método se ha utilizado únicamente en generadores como Styleformer (Park and Kim, 2022), sin embargo, esta GAN requiere de un discriminador convolucional.

Reducción de muestreo La reducción de muestreo de Swin Transformer (Liu et al., 2021) consiste en limitar la autoatención a ventanas que contienen $M \times M$ parches y a ventanas desplazadas como se muestra en la figura 3.3. Las ventanas desplazadas permiten al modelo aprender información entre las ventanas vecinas. Primero, el mapa de características de 8×8 parches se divide en 2×2 ventanas de 4×4 ($M = 4$) parches. En el siguiente bloque, las ventanas se desplazan $(M/2, M/2)$ parches. Este mecanismo de atención es utilizado en GANs como StyleSwin (Zhang et al., 2022), sin embargo, al igual que Styleformer, se utiliza únicamente en el generador debido a la inestabilidad que genera este método en el discriminador (Lee et al., 2021).

3.4.4. Continuidad de Lipschitz y autoatención

Como se describe en la sección 3.2, entrenar GANs para conseguir que la distribución de los datos generados p_g sea lo más similar a los datos reales p_r, requiere penalizar los gradientes o normalizar los pesos. Sin embargo, estas estrategias han sido diseñadas para redes convolucionales. Al momento de trabajar con autoatención en el discriminador, la normalización espectral no es igual de efectiva (Lee et al., 2021).

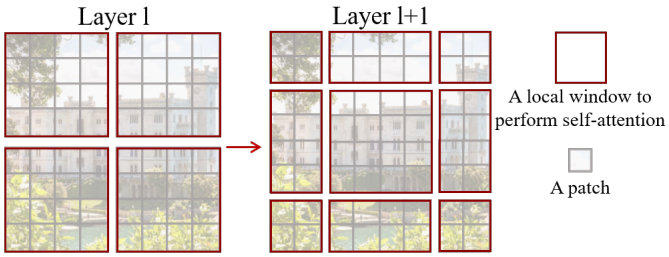


Figura 3.3: Ventana regular (izquierda) y ventana despasada (derecha) de Swin-Transformer. Imagen de Liu et al. (2021)

.

Reciente trabajo muestra que el MLP del Transformador incrementa la constante de Lipschitz para evitar que la salida del módulo converja a rango 1, lo que ocasiona grandes variaciones en los gradientes, dificultando la optimización (Dong et al., 2021).

Una manera de adaptar la normalización espectral a ViTs es utilizando la normalización espectral mejorada (Lee et al., 2021):

$$\bar{W}_{\text{ISN}}(\text{W}) := \sigma(\text{W}_{init}) \cdot \text{W}/\sigma(\text{W}) \tag{3.39}$$

donde $\sigma(\text{W}_{init})$ es la norma espectral a la inicialización. Esta normalización es acompañada de la atención L2 (Kim et al., 2021) para complementar la estabilización del discriminador, puesto que la Regularización R1 es inestable en los entrenamientos de Transformadores GANs cuando se usan discriminadores ViTs.

La inestabilidad de los mecanismos de atención en los discriminadores Transformadores no permite emplear métodos para reducir la complejidad cuadrática. Esto limita procesar secuencias largas de manera eficiente (Tay et al., 2020) con el discriminador para aprovechar el aprendizaje de dependencias largas. Una solución sencilla para abordar esta limitante es utilizar discriminadores convolucionales como en VITGAN (Lee et al., 2021), Styleformer (Park and Kim, 2022), HiT (Zhao et al., 2021a) y StyleSwin (Zhang et al., 2022).

3.5. Representaciones neuronales implícitas

Las representaciones neuronales implícitas (INRs, por las siglas en inglés de *Implicit neural representations*), son modelos diseñados principalmente para modelado en 3D (Mildenhall et al., 2020). Sin embargo, recientes trabajos en GANs como CIPS (Anokhin et al., 2021) y VITGAN (Lee et al., 2021) han adoptado estos modelos para la generación en 2D. CIPS consigue resultados similares a StyleGAN2 ingresando las coordenadas $(x, y) \in \{0, \ldots, W-1\} \times \{0, \ldots, H-1\}$ al generador para la generación de imágenes de dimensión $H \times W$, tal que $G(x, y, z)$ [1]. Haciendo uso de estas coordenadas con activaciones sinusoidales (Sitzmann et al., 2020; Tancik et al., 2020) se definen los *embeddings* de Fourier denotados como E_{fou}:

$$\text{E}_{fou}(x, y) = \sin[\text{W}_{fou}(x', y')^T] \tag{3.40}$$

[1] Únicamente en esta sección, las variables x y y se utilizan para denotar las coordenadas.

donde $x' = \frac{2x}{W-1} - 1$ y $\frac{y'}{H-1} - 1$ son las coordenadas de los píxeles que van del rango $[-1, 1]$ y $\mathrm{W}_{fou} \in \mathbb{R}^{2\times n}$ es una transformación lineal.

Además, CIPS requiere también de *embeddings* de coordenadas $\mathrm{E}_{co}^{(x,y)}$, los cuales consisten de $H \times W$ vectores que se entrenan. De esa manera, los *embeddings* posicionales son la concatenación de los *embeddings* de Fourier y los *embeddings* de coordenadas:

$$\mathrm{E}(x, y) = \mathrm{concat}[\mathrm{E}_{fou}(x, y), \mathrm{E}_{co}^{(x,y)}]. \tag{3.41}$$

Estos *embeddings* son recibidos como entrada por capas de MLPs, las cuales son moduladas por w utilizando una red de mapeo, como se describe en la sección 3.4.2. Por otro lado, en VITGAN, E_{fou} consiste únicamente en la proyección de cada coordenada de P^2 píxeles normalizadas entre el rango $[-1, 1]$, seguida de una activación sinusoidal. De esta manera, E_{fou} es transformado por un MLP de dos capas modulado por el *embedding* del parche y_i, el cual consiste en la salida del Transformador.

3.6. Aumento de datos y regularización

3.6.1. Aumento diferenciable

El aumento de datos más general en problemas de clasificación consiste en aplicar un aumento T al conjunto de datos reales. En GANs, dadas las funciones f_D y f_G, la función objetivo con aumento de datos es:

$$\mathcal{L}_D = \mathbb{E}_{x\sim p_r(x)}[f_D(-D(T(x)))] + \mathbb{E}_{z\sim p_z(z)}[f_D(D(G(z)))], \tag{3.42}$$
$$\mathcal{L}_G = \mathbb{E}_{z\sim p_z(z)}[f_G(-D(G(z)))]. \tag{3.43}$$

Sin embargo, aumentar solo los datos reales ocasiona que G aprenda a generar $T(x)$ en lugar de x, que es igual a generar muestras con aumento de datos.

Por otro lado, si se aplica el aumento T también al conjunto de datos generados, la función es la siguiente:

$$\mathcal{L}_D = \mathbb{E}_{x\sim p_r(x)}[f_D(-D(T(x)))] + \mathbb{E}_{z\sim p_z(z)}[f_D(D(T(G(z))))], \tag{3.44}$$
$$\mathcal{L}_G = \mathbb{E}_{z\sim p_z(z)}[f_G(-D(G(z)))]. \tag{3.45}$$

Donde se aumentan todos los datos para la función objetivo de D. Sin embargo, esta estrategia ocasiona que G engañe por completo a D, y así no pueda obtener suficiente información para modelar p_r.

Para resolver estos problemas, el aumento diferenciable (Zhao et al., 2020) propaga el aumento de datos a través de G de la siguiente manera:

$$\mathcal{L}_D = \mathbb{E}_{x\sim p_r}(x)[f_D(-D(T(x)))] + \mathbb{E}_{z\sim p_z(z)}[f_D(D(T(G(z))))], \tag{3.46}$$
$$\mathcal{L}_G = \mathbb{E}_{z\sim p_z(z)}[f_G(-D(T(G(z))))]. \tag{3.47}$$

Algoritmo 4 Regularización balanceo de consistencia (Zhao et al., 2021b)

Require: α es la tasa de aprendizaje, m es el número de elementos por mini-lote y k es el número de iteraciones que se entrena D, λ_{real} y λ_{fake} los coeficientes de regularización y T la transformación de aumento de datos.

for número de iteraciones **do**
  for $t = 0, ..., k$ **do**
    Muestrea $\{x^{(i)}\}_{i=1}^{m} \sim p_r$ mini-lote de la distribución real
    Muestrea $\{z^{(i)}\}_{i=1}^{m} \sim p_z$ mini-lote de ruido
    $\tilde{x} \leftarrow G(z)$ Genera mini-lote
    $o_r \leftarrow D(x)$ Muestra real
    $o_g \leftarrow D(\tilde{x})$ Muestra generada
    $L_{real} \leftarrow ||D(x) - D(T(x))||^2$
    $L_{fake} \leftarrow ||D(\tilde{x}) - D(T(\tilde{x}))||^2$
    $\mathcal{L}_D \leftarrow \log(o_r) + \log(1 - o_g) + \lambda_{real} L_{real} + \lambda_{fake} L_{fake}$
    $D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$
  end for
  Muestrea $\{z^{(i)}\}_{i=1}^{m} \sim p_z$ mini-lote de ruido
  $\tilde{x} \leftarrow G(z)$ Genera mini-lote
  $o_g \leftarrow D(\tilde{x})$ Muestra generada
  $\mathcal{L}_G \leftarrow \log(o_g)$
  $G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$

Este método mejora significativamente el aprendizaje en modelos convolucionales (Brock et al., 2019; Karras et al., 2020), sin embargo, el efecto en Transformadores (Jiang et al., 2021; Lee et al., 2021) es aún más efectivo al modelar la distribución p_r. Eso es debido a que los Transformadores requieren de una cantidad mayor de datos para entrenarse. Además de contar con una mayor capacidad comparada con los modelos convolucionales.

3.6.2. Regularización balanceo de consistencia

El balanceo de consistencia (bCR, por las siglas en inglés de *Balanced Consistency Regularization*) (Zhao et al., 2021b) es una técnica de regularización que consiste en hacer que el modelo produzca predicciones consistentes. Esto se consigue agregando un término a la función del discriminador que penaliza la diferencia entre las imágenes reales x y las imágenes aumentadas $T(x)$, y las imágenes generadas $\tilde{x}$ y las imágenes generadas aumentadas $T(\tilde{x})$. Este método elimina aumentos generados y mejora significativamente el modelado de la distribución p_r. El procedimiento de entrenamiento se muestra en el Algoritmo 4. Aunque originalmente este método se utilizó en BigGAN (Brock et al., 2019), al igual que el Aumento de datos diferenciable (Zhao et al., 2020), bCR tiene un impacto aún mayor en modelos Transformadores en comparación con los modelos convolucionales. Este método es utilizado en VITGAN (Lee et al., 2021), HiT (Zhao et al., 2021a) y StyleSwin (Zhang et al., 2022).

Capítulo 4

Transformadores antagónicos generativos de atención aditiva

En este capítulo se describen los Transformadores antagónicos generativos de atención aditiva (AdatGAN). Estos modelos generan secuencias de longitud 1024 utilizando un mecanismo de atención $O(N)$ con respecto a la longitud de la secuencia de entrada N. El mecanismo de atención aditiva consigue un mejor comportamiento de gradientes a diferencia del mecanismo de atención producto punto. Además, los Transformadores de atención aditiva son compatibles con convoluciones, lo que permite utilizar en conjunto el sesgo inductivo de las convoluciones y de los Transformadores para tareas generativas. Estas arquitecturas híbridas han conseguido resultados prometedores en la clasificación de imágenes. Finalmente, el poder de cómputo en FLOPs requerido por AdatGAN es menor al de GANs como StyleGANs y VITGAN, consiguiendo resultados de evaluación muy similares.

4.1. Atención aditiva

El objetivo del mecanismo de atención aditiva es poder entrenar Transformadores eficientes tanto en el generador como en el discriminador utilizando penalización de gradientes. Aunque la reducción de muestreo de Swin Transformer (Liu et al., 2021) hace más eficiente la autoatención producto punto, la penalización de gradientes es inestable con este mecanismo (Lee et al., 2021; Zhang et al., 2022). En consecuencia, VITGAN requiere de la normalización espectral mejorada, la cual emplea atención L2 (Kim et al., 2021), lo que resulta en una estrategia más elaborada que la penalización de gradientes, y no consigue superar los resultados de los discriminadores convolucionales. En otras palabras, la inestabilidad de los Transformadores en los discriminadores limita la capacidad de las GANs. Modelos como Styleformer (Park and Kim, 2022), HiT (Zhao et al., 2021a) y StyleSwin (Zhang et al., 2022) utilizan discriminadores convolucionales, por lo que tampoco aprovechan la capacidad de los Transformadores en el discriminador.

El mecanismo de atención de AdatGAN se basa en Fastformer (Wu et al., 2021a), el cual está diseñado para procesamiento de texto, y es diferente de la atención aditiva de Bahdanau et al. (2014). Los Transformadores de AdatGAN, además de ser $O(N)$ y conseguir un mejor comportamiento de gradientes, generan mapas de

atención aditiva, los cuales se expanden progresivamente para guiar la generación de parches y píxeles. La atención aditiva calcula los pesos α de la matriz query $\mathrm{Q} = [\mathrm{q}_1, ..., \mathrm{q}_N]$, proyectando con un vector $\mathrm{w} \in \mathbb{R}^d$, donde $\mathrm{Q} \in \mathbb{R}^{(H \times W) \times d}$, d es la dimensión del cabezal, y H, W denotan el alto y ancho del mapa de características:

$$\alpha_i = \frac{\exp(\mathrm{w}^T \mathrm{q}_i / \sqrt{d})}{\sum_{j=1}^{N} \exp(\mathrm{w}^T \mathrm{q}_j / \sqrt{d})}. \tag{4.1}$$

Los pesos de atención y los vectores de Q calculan el vector global $\mathrm{q}^G = \sum_{i=1}^{N} \alpha_i \mathrm{q}_i$ para modelar las interacciones entre los canales de los mapas de características y las dependencias de largo rango, donde $\mathrm{q}^G \in \mathbb{R}^d$. Para propagar la información aprendida, se realiza la operación elemento a elemento entre q^G y cada vector de la matriz $\mathrm{K} = [\mathrm{k}_1, ..., \mathrm{k}_N]$, donde $\mathrm{K} \in \mathbb{R}^{(H \times W) \times d}$. De esta manera, se obtienen los vectores p_i, tal que $\mathrm{p}_i = \mathrm{q}^G \odot \mathrm{k}_i$. A diferencia de Fastformer (Wu et al., 2021a), el mecanismo de AdatGAN no modela un vector global para K; en su lugar, se realiza la operación elemento a elemento entre p_i y los vectores $\mathrm{V} = [\mathrm{v}_1, ..., \mathrm{v}_N]$, donde $\mathrm{V} \in \mathbb{R}^{(H \times W) \times d}$. Esto permite propagar la información del mapa de atención aditiva a los valores V, en lugar de comprimirla. De esta manera, cada vector de la matriz de salida $\mathrm{R} = [\mathrm{r}_1, ..., \mathrm{r}_N] \in \mathbb{R}^{(H \times W) \times d}$ se obtiene como $\mathrm{r}_i = \mathrm{p}_i \odot \mathrm{v}_i$. En la figura 4.1 se muestra una ilustración del proceso de generación del mecanismo de atención.

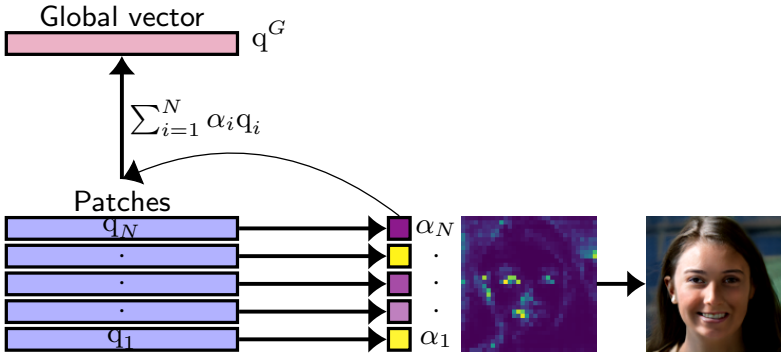


Figura 4.1: Mapa de atención aditiva para la generación de parches de 4×4.

4.2. Transformador generativo de atención aditiva

El Transformador generativo de atención aditiva (*Adat-Transformer*) en general consiste en la arquitectura del Transformador original (Vaswani et al., 2017; Dosovitskiy et al., 2021). Además, al igual que en VITGAN, se utiliza la normalización de capa modulada (Chen et al., 2019a; Lee et al., 2021), como se muestra en la figura 4.2. La diferencia más importante comparado con el Transformador de VITGAN es la complejidad de la autoatención, la cual se reduce de $O(N^2)$ a $O(N)$. También, a diferencia de ViT (Dosovitskiy et al., 2021) y Fastformer (Wu et al., 2021a), el

Transformador de atención aditiva no cuenta con la conexión residual que va de la salida del módulo de atención a la salida del MLP. Esta arquitectura se emplea en AdatGAN antes de realizar *pixel shuffle* (Shi et al., 2016). Esto permite que la expansión de la longitud de secuencia del Transformador al transformar la secuencia $X_i \in \mathbb{R}^{(H\times W)\times C}$ a la secuencia $X'_i \in \mathbb{R}^{(2H\times 2W)\times C/4}$ dependa únicamente del MLP, el cual tiene como entrada una representación modulada de la información del vector global y del mapa de atención, lo que precisamente resulta útil en *pixel shuffle*, ya que el vector global consiste de una representación de canales comprimida, la cual es aprendida con atención de dependencias de largo rango.

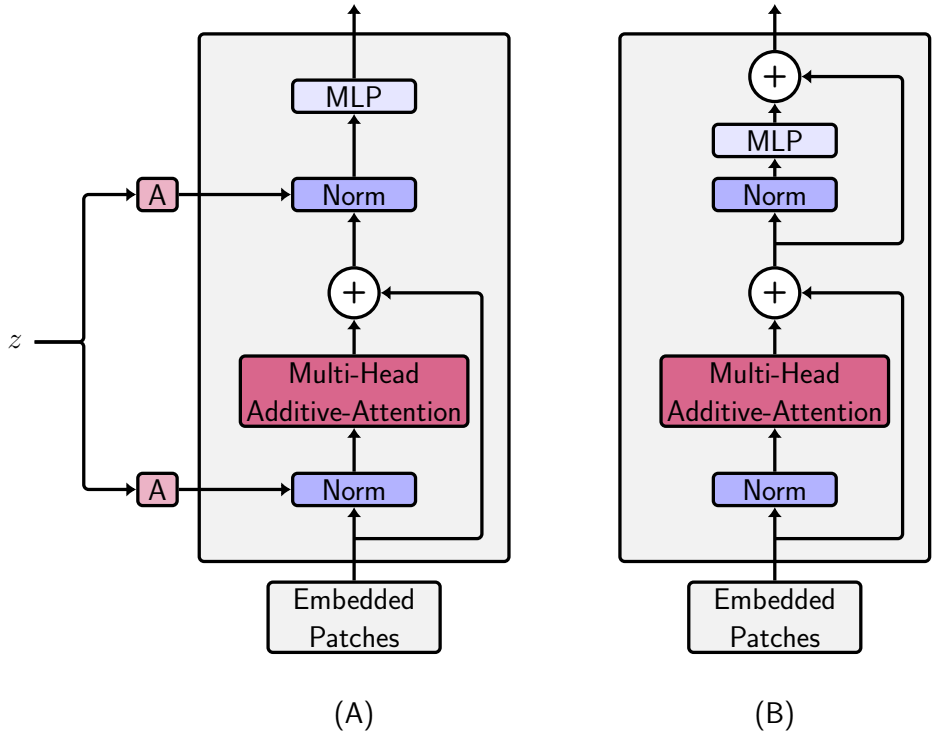


Figura 4.2: Transformadores de atención aditiva de AdatGAN. **(A)** Transformador modulado del generador y **(B)** Transformador del discriminador .

4.3. Atención aditiva y convoluciones

A diferencia de VITGAN (Lee et al., 2021), TransGAN (Jiang et al., 2021), HiT (Zhao et al., 2021a) y StyleSwin (Zhang et al., 2022), que proponen arquitecturas que consisten únicamente de Transformadores, AdatGAN tiene como objetivo ser una arquitectura compatible con convoluciones. Las arquitecturas que utilizan convoluciones con Transformadores han mostrado mejores resultados que las arquitecturas que consisten únicamente de Transformadores. Sin embargo, estas propuestas han sido diseñadas para clasificación de imágenes (Touvron et al., 2021b; Park and Kim, 2021; Wu et al., 2021b). CvT (Wu et al., 2021b) por ejemplo, es una generalización del Transformador y consiste en transformar los *tokens* a un mapa de *tokens* de 2D.

Después, aplica convoluciones separables en profundidad con parámetro *strides* = 2 para las proyecciones de K y V. Esto permite reducir el número total de *tokens* de K y V similar a Linformer (Wang et al., 2020), lo que mejora la eficiencia del modelo. Además, debido al aprendizaje local en las convoluciones, el embbeding de posición original de ViT no es requerido en CvT. De esta manera, CvT utiliza el *Embedding de token convolucional*.

Por otro lado, el generador de AdatGAN tiene como objetivo una tarea generativa, por lo que hace uso de *pixel shuffle* para progresivamente transformar las dimensiones de la variable latente, expandiendo mapas de atención aditiva con Transformadores. Sin embargo, la atención aditiva está diseñada principalmente para el aprendizaje de dependencias largas. Además, la operación de *pixel shuffle* no favorece la localidad en el cambio de la estructura de las representaciones producidas por los Transformadores. Debido a esto, en AdatGAN se emplean convoluciones para reforzar la localidad de *pixel shuffle*, aprovechando la compatibilidad de la atención aditiva con las convoluciones.

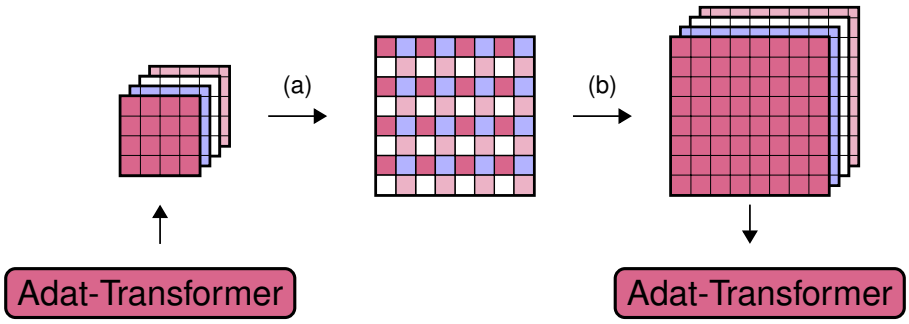


Figura 4.3: Expansión local del *embedding*. La salida del primer Transformador consiste en 4 mapas de características que se expanden con *pixel shuffle* **(a)** para generar un solo mapa de características. Después, el mapa de la salida de *pixel shuffle* pasa por una convolución **(b)** para expandir a 4 canales. Estos nuevos 4 mapas de características son la entrada del segundo Transformador. De esta manera, aunque los Transformadores procesan secuencias de diferente longitud, las dimensiones de los *embeddings* son independientes de la operación *pixel shuffle*.

Expansión local del *embedding* Una dificultad de utilizar *pixel shuffle* en un generador puramente Transformador es que las dimensiones de los *embeddings* en los diferentes Transformadores del modelo dependen del primer Transformador. Por ejemplo, si los mapas de características $X_0 \in \mathbb{R}^{(8\times8)\times1024}$ se utilizan en el primer Transformador del generador, al momento de generar 1024 *tokens* después de realizar dos operaciones de *pixel shuffle*, la forma de los mapas de características está limitada a $X_2 \in \mathbb{R}^{(32\times32)\times64}$. Para poder generar un mapa de características $X_2 \in \mathbb{R}^{(32\times32)\times256}$, se requiere que el primer Transformador tenga la forma de $X_0 \in \mathbb{R}^{(8\times8)\times4096}$, lo que resulta en una dimensión de *embedding* demasiado grande para una secuencia de 64 *tokens*, además de aumentar significativamente el costo computacional. Una manera eficiente de aumentar la flexibilidad del modelo es utilizar la *Expansión local del embedding* (LEE, por las siglas en ingles de *Local embedding expansion*). Esta

operación consiste en una capa de convoluciones después de realizar *pixel shuffle* entre dos Transformadores, denotada como LLE = Conv(PS(X)). Además de reforzar la localidad, la convolución permite expandir o disminuir de manera local el número de canales, que corresponde a la dimensión del *embedding*. Esto consigue que la dimensión de cada Transformador sea independiente, aunque la longitud de secuencia se expanda con *pixel shuffle*, como se muestra en la figura 4.3. Este método se puede utilizar con Transformadores generativos con mecanismos de atención compatibles con convoluciones, como Adat-Transformer.

4.4. Generador

A diferencia de VITGAN, AdatGAN utiliza *pixel shuffle*, lo que permite transformar progresivamente la variable latente z a través de mapas de atención aditiva. TransGAN lo hace de manera similar, sin embargo, utiliza la autoatención producto punto y *grid self-attention*. Otra diferencia es que AdatGAN utiliza la misma arquitectura de tres transformadores para las tres resoluciones 32×32, 64×64 y 128×128, como se muestra en la figura 4.4. Por otro lado, para la resolución 32×32, TransGAN utiliza una pila de once transformadores.

Además, TransGAN genera imágenes a nivel de píxel, lo que hace que sea costoso escalar este modelo a altas resoluciones (1024×1024), ya que la longitud de secuencia se vuelve demasiado larga. Por otro lado, VITGAN genera parches para resolver la limitación de la longitud de secuencia utilizando INRs. Similar a VITGAN, AdatGAN también genera parches, sin embargo, la secuencia de salida de AdatGAN es de mayor longitud (1024 *tokens*) debido a su eficiencia, ya que no requiere autoatención producto punto. Además, el objetivo de generar secuencias largas en AdatGAN es generar un mapa de atención aditiva final de 1024 *tokens*, el cual contiene suficiente información para guiar la generación de píxeles y parches. Finalmente, este modelo no requiere de INRs a diferencia de VITGAN:

$$\mathbf{h}_0 = \mathrm{MLP}(\mathrm{z}), \qquad \mathrm{z} \in \mathbb{R}^{D_z} \tag{4.2}$$

$$\mathbf{h}'_\ell = \mathrm{MAA}(\mathrm{SLN}(\mathbf{h}_{\ell-1} + \mathrm{E}_{\ell-1}, \mathrm{z})) + \mathbf{h}_{\ell-1}, \qquad \ell = 1, \ldots, L, \mathrm{E}_{\ell-1} \in \mathbb{R}^{N_{\ell-1} \times D_{\ell-1}} \tag{4.3}$$

$$\mathbf{h}_\ell = \mathrm{LEE}(\mathrm{MLP}(\mathrm{SLN}(\mathbf{h}'_\ell, \mathrm{z}))), \qquad \ell = 1, \ldots, L \tag{4.4}$$

$$\mathrm{Y} = \mathrm{MAA}(\mathrm{SLN}(\mathbf{h}_L + \mathrm{E}_L, \mathrm{z})) + \mathbf{h}_L \qquad \mathrm{E}_L \in \mathbb{R}^{N_L \times D_L} \tag{4.5}$$

$$\mathrm{X} = \mathrm{Conv}(\mathrm{MLP}(\mathrm{SLN}(\mathrm{Y}, \mathrm{z}))) \qquad \mathrm{X} \in \mathbb{R}^{H \times W \times C} \tag{4.6}$$

Aquí, X denota la imagen de salida, E el *embedding* de posición y MAA la atención aditiva multi cabeza. Cabe señalar que antes de realizar la convolución final en la ecuación 4.6, se realiza un *reshape* para generar el mapa de *tokens* de 2D (1024 *tokens*). Por otro lado, si el número de canales de salida de la convolución de LLE en la ecuación 4.4 es igual el número de canales de entrada, no se realiza expansión en la dimensión del *embedding*. Esto resulta en que la convolución únicamente refuerce la localidad de *pixel shuffle*.

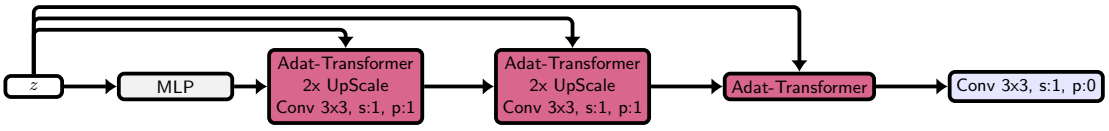


Figura 4.4: Generador de AdatGAN.

4.5. Discriminador

A diferencia de VITGAN (Lee et al., 2021), Styleformer (Park and Kim, 2022), HiT (Zhao et al., 2021a) y StyleSwin (Zhang et al., 2022), la estabilidad de la atención aditiva permite entrenar discriminadores Transformadores $O(N)$ con penalización de gradientes. A diferencia de VITGAN, que utiliza ViT para extraer parches de las imágenes para alimentar a los Transformadores, la compatibilidad de la atención aditiva con las convoluciones permite que el discriminador de AdatGAN tenga como entrada bloques residuales de FastGAN (Liu et al., 2020) para alimentar a un Transformador, como se muestra en la figura 4.5. La idea de utilizar bloques convolucionales y autoatención se muestra en arquitecturas como Alter-ResNet-50 (Park and Kim, 2021), sin embargo, Alter-ResNet-50 está propuesta para la clasificación de imágenes y no utiliza atención aditiva.

Por otro lado, a diferencia del Transformador del generador, el Transformador del discriminador cuenta con la conexión residual del MLP. Esto es debido a que el discriminador de AdatGAN no realiza una tarea generativa, por lo que el MLP no trabaja con *pixel shuffle* para expandir la longitud de secuencia. Además, a diferencia de la salida del discriminador de TransGAN y ViT que utilizan el *embedding* de clase (Devlin et al., 2019), la salida del discriminador de AdatGAN consiste en utilizar convoluciones con *strides* $= 2$. De esta manera, las convoluciones progresivamente reducen la dimensión de la salida del Transformador.

Finalmente, el módulo de normalización por lotes (Ioffe and Szegedy, 2015) en el extractor de características convolucional resulta ser esencial para complementar la estabilidad del discriminador Transformador. Arquitecturas con discriminadores Transformadores como VITGAN y TransGAN no cuentan con normalización por lotes.

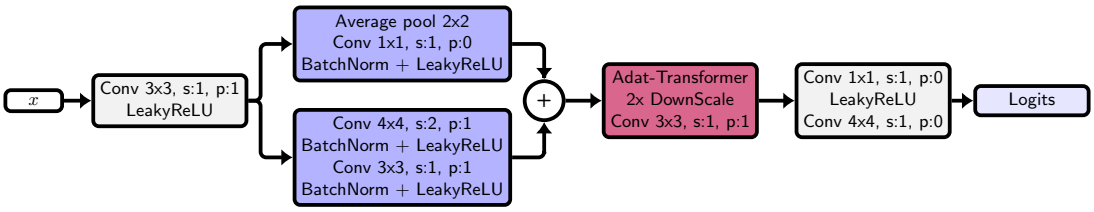


Figura 4.5: Discriminador de AdatGAN.

Capítulo 5

Experimentos

En este capítulo se describen los conjuntos de datos, métricas y los experimentos realizados para la evaluación de los modelos.

5.1. Bases de datos

Para evaluar la arquitectura propuesta se utilizaron 4 conjuntos de datos típicamente empleados en modelos generativos profundos. A continuación se describe de forma general cada uno de ellos.

5.1.1. CIFAR-10

CIFAR-10 (Krizhevsky et al., 2009) consiste de imágenes de resolución 32×32. El conjunto de datos contiene 50k imágenes de entrenamiento y 10k imágenes de prueba, donde hay 10 clases y 6k imágenes por cada una.

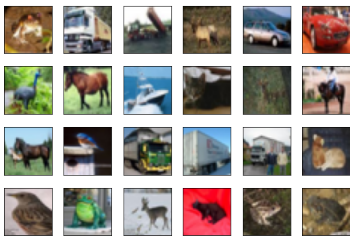


Figura 5.1: Muestras de resolución 32×32 del conjunto de datos CIFAR-10 (Krizhevsky et al., 2009).

5.1.2. CelebA

CelebA (Liu et al., 2015) se utiliza para entrenar en resolución 64×64. Este conjunto de rostros consiste de 162,770 imágenes de entrenamiento y 19,962 de prueba.

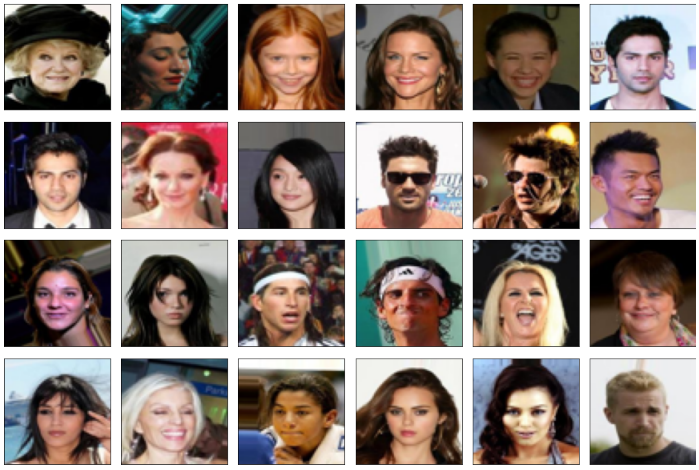


Figura 5.2: Muestras de resolución 64 × 64 del conjunto de datos CelebA (Liu et al., 2015).

5.1.3. LSUN Bedroom

LSUN Bedroom (Yu et al., 2015) se utiliza para entrenar en resolución 128 × 128. Este conjunto consiste de 3 millones de imágenes de dormitorios.

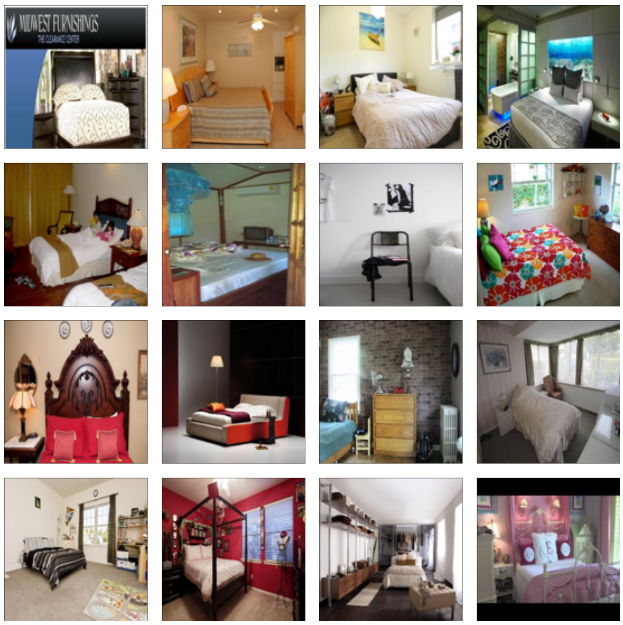


Figura 5.3: Muestras de resolución 128 × 128 del conjunto de datos LSUN Bedroom (Yu et al., 2015).

5.1.4. FFHQ

FFHQ (Karras et al., 2019) se utiliza para entrenar en resolución 128 × 128. Este conjunto consiste en 70k imágenes de rostros con considerables variaciones en términos de edad, etnia y fondos de imagen.

5.2. Métricas

5.2.1. Distancia Fréchet Inception

Para evaluar la efectividad del modelo se utiliza la métrica FID (por las siglas en inglés de *Fréchet Inception Distance*) (Heusel et al., 2017). Esta métrica emplea una red Inception-v3 (Szegedy et al., 2016) pre-entrenada en la tarea de clasificación de imágenes. La evaluación consiste en calcular la divergencia KL entre las imágenes del conjunto de datos reales y las imágenes del conjunto de datos generados. Esto se consigue calculando las activaciones de la red Inception-v3. Entre menor es la evaluación FID, el conjunto de datos generados es más similar al conjunto de datos reales. Esta métrica es un estándar para evaluar GANs.

La evaluación FID de las gráficas se obtiene mediante 10k muestras para todos los conjuntos de datos. Para las tablas, se utilizan los pesos originales de FID (Heusel et al., 2017) en PyTorch (Seitzer, 2020), siguiendo en general el estándar de evaluación en GANs (Karras et al., 2019, 2020; Lee et al., 2021; Zhang et al., 2022; Zhao et al., 2021a).

- En CIFAR-10, la evaluación se realiza en las 50k imágenes de entrenamiento.
- En CelebA, se utilizan las 19,962 imágenes de prueba.
- En LSUN Bedroom, se muestrean aleatoriamente 70k imágenes del conjunto de entrenamiento.
- En FFHQ, se utilizan las 70k imágenes de entrenamiento.

5.3. Detalles de implementación

Todos los modelos se entrenan con regularización R1 (Mescheder et al., 2018) y optimizador Adam (Kingma and Ba, 2014) con los factores de ponderación del primer y segundo momento de los gradientes $\beta_1 = 0.5$ y $\beta_2 = 0.99$, siguiendo la práctica de VITGAN (Lee et al., 2021). Para todos los entrenamientos, la tasa de aprendizaje del generador es de 0.0002. Para los discriminadores convolucionales, se utiliza una tasa de 0.0004, haciendo uso de tasas de aprendizaje desbalanceadas (TTUR, por las siglas en inglés de *two-timescale update rule*) (Heusel et al., 2017). Para los discriminadores Transformadores, se utiliza una tasa de 0.0002, ya que TTUR genera inestabilidad. El Transformador inicial de AdatGAN genera mapas de 8×8 de dimensión 1024, seguido de un Transformador de 16×16 de dimensión 256. La única diferencia para cada conjunto de datos es la dimensión del Transformador final de 32×32. Para CIFAR-10, se utiliza dimensión de 64 y 128, y la generación es a nivel píxel. Para CelebA, la dimensión es de 128 y se generan parches de 2×2. Por último, para LSUN Bedroom y FFHQ, la dimensión es de 512 y se generan parches de 4×4. El número de cabezas es 4 y la dimensión del MLP es 512 en todos los Transformadores. Todos los entrenamientos en CIFAR-10 se realizan con aumento de datos de Traslación, Color y Cutout (Zhao et al., 2020), empleando bCR (Zhao et al., 2021a) con $\lambda_{real} = \lambda_{fake} = 1.0$ y tamaño de lote 128.

5.4. Autoatención $O(N)$ y explosión de gradientes

En los experimentos de control de explosión de gradientes y autoatención $O(N)$, se evalúa la efectividad del mecanismo de atención aditiva para controlar los picos de las magnitudes (Lin et al., 2021) con regularización R1 (Mescheder et al., 2018). En este experimento, a diferencia de VITGAN (Lee et al., 2021) que utiliza la normalización espectral mejorada, se controla la estabilidad de los gradientes del discriminador mediante un generador Transformador con mecanismo de atención aditiva $O(N)$. Con el objetivo de realizar entrenamientos eficientes y estables, se utiliza el discriminador convolucional de FastGAN (Liu et al., 2020) sin normalización por lotes (Ioffe and Szegedy, 2015) ni normalización espectral (Lin et al., 2021). Esto debido a que los discriminadores convolucionales son más estables que los discriminadores Transformadores (Lee et al., 2021; Zhang et al., 2022). Además, para evaluar la efectividad de la atención aditiva propuesta, la arquitectura se compara con la autoatención de bajo rango de Linformer (Wang et al., 2020) y de reducción de muestreo de Swin-Transformer (Liu et al., 2021), y la atención aditiva original de Fastformer (Wu et al., 2021a). Todos los experimentos con Linformer (Wang et al., 2020) se realizan con $k = 64$. Los experimentos de Swin Transformer (Liu et al., 2021) con ventanas de tamaño 8×8.

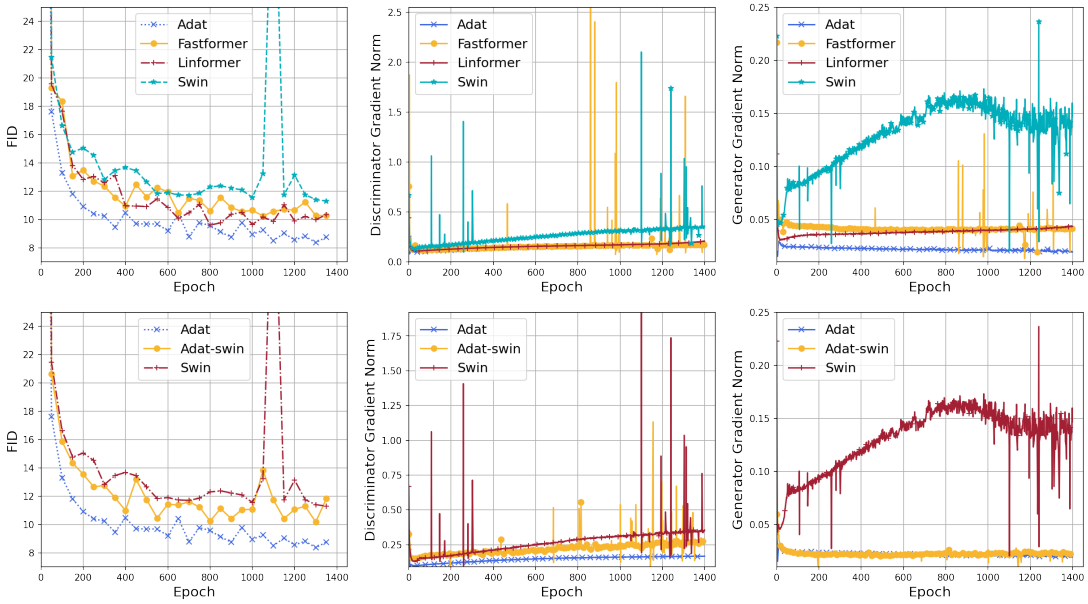


Figura 5.4: Magnitudes de gradientes sobre todos los parámetros y evaluación FID de Linformer, Swin Transformer, Fastformer y Adat en el generador.

Como se muestra en la tabla 5.1, los FLOPs de los mecanismos de atención resultan similares, en comparación con los FLOPs de StyleGAN2 y VITGAN de la tabla 5.12. En la figura 5.4 se observa cómo Swin Transformer es el modelo que muestra más picos en los gradientes del discriminador, resultando en un FID alto. Por otro lado, las arquitecturas con Linformer, Fastformer y Adat consiguen controlar los gradientes con regularización R1. Debido a que Linformer es autoatención producto punto, y Fastformer y Adat son atención aditiva, los resultados indican que son

Tabla 5.1: FLOPS y evaluación FID de Linformer, Swin Transformer, Fastformer y Adat en el generador sobre 50k imágenes de CIFAR-10.

Arquitectura	**Conv**	**FLOPs**	**FID** ↓
Swin (Liu et al., 2021)	✗	0.5B	8.05
Linformer (Wang et al., 2020)	✗	0.6B	6.03
Fastformer (Wu et al., 2021a)	✗	0.4B	6.66
AdatSwin	✗	0.4B	6.53
Adat	✓	0.7B	**4.88**

las ventanas de Swin Transformer las que ocasionan los picos de las magnitudes de los gradientes, lo que resulta en un FID alto. Con el objetivo de corroborar esta hipótesis se desarrolló AdatSwin. Este modelo trabaja con ventanas de atención aditiva de tamaño 8×8. Como se muestra en la figura 5.4, utilizar ventanas de atención aditiva igualmente resulta en picos en los gradientes del discriminador. Sin embargo, los gradientes del generador son más constantes. Esto indica que la atención de Adat en el generador tiene un mejor comportamiento de gradientes que la autoatención producto. Lo anterior permite entrenar Transformadores con penalización de gradientes consiguiendo el mejor FID.

5.5. Convoluciones y Transformadores

Tabla 5.2: FLOPS y evaluación FID de Linformer y Adat trabajando con convoluciones en el generador sobre 50k imágenes de CIFAR-10.

Arquitectura	**Conv**	**FLOPs**	**FID** ↓
Linformer (Wang et al., 2020)	✗	0.6B	6.06
Linformer (Wang et al., 2020)	✓	0.8B	6.63
Fastformer (Wu et al., 2021a)	✗	0.4B	6.66
Fastformer (Wu et al., 2021a)	✓	0.6B	6.68
Adat	✗	0.5B	5.91
Adat	✓	0.7B	**4.88**

Como se muestra en la figura 5.5, a diferencia de Linformer y Fastformer, la atención de AdatGAN es compatible con las convoluciones. Agregar convoluciones empeora la aproximación de bajo rango de Linformer (Wang et al., 2020) y no muestra un cambio significativo en Fastformer (Wu et al., 2021a). Por otro lado, Adat muestra una mejora significativa al agregar convoluciones después de las etapas de *pixel shuffle* y reforzar el aprendizaje local. Por último, no se nota que el uso de convoluciones en Transformadores provoque inestabilidad en los gradientes. Los FLOPs de la tabla 5.2 resultan solo en un ligero incremento al usar convoluciones.

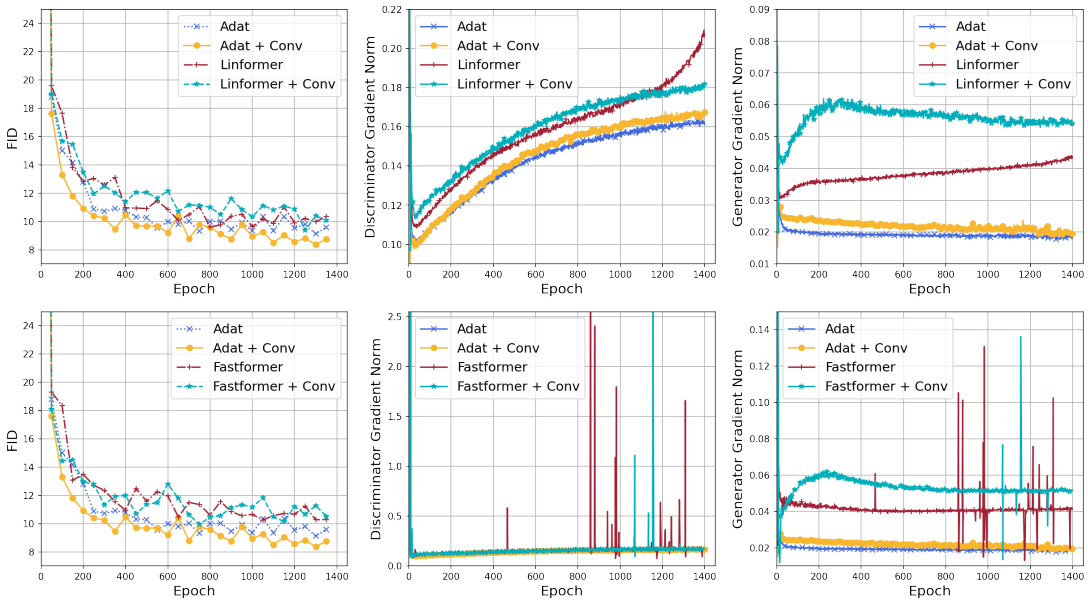


Figura 5.5: Magnitudes de gradientes sobre todos los parámetros y evaluación FID de Linformer, Fastformer y Adat utilizando convoluciones en el generador.

5.6. Conexiones residuales y autoatención $O(N)$

En la figura 5.6 se puede observar que el Transformador original (Vaswani et al., 2017) con Linformer disminuye su aprendizaje al eliminar la conexión residual del MLP. Además, genera inestabilidad en el discriminador. Lo anterior ocasiona que no se cumpla la condición de Lipchitz, lo que resulta en el FID más alto como se muestra en la tabla 5.3. Por otro lado, el Transformador de atención aditiva muestra ser más robusto. Al eliminar o preservar la conexión residual del MLP, las normas permanecen estables y con una evaluación FID similar. Sin embargo, AdatGAN mejora aún más la evaluación FID si no se agrega la conexión residual del MLP. Esto se debe a que la entrada de *pixel shuffle* depende únicamente del MLP, como se menciona en la sección 4.2. Además, el vector global crea una representación de los canales utilizando los pesos de atención aditiva, lo que resulta útil al trabajar con canales en *pixel shuffle*.

Tabla 5.3: Evaluación FID de la conexión residual del MLP en Linformer y Adat sobre 50k imágenes de CIFAR-10.

Arquitectura	**Residual-MLP**	**FID** $\downarrow$
Linformer (Wang et al., 2020)	✓	6.63
Linformer (Wang et al., 2020)	✗	10.68
Fastformer (Wu et al., 2021a)	✓	6.68
Fastformer (Wu et al., 2021a)	✗	5.32
Adat	✓	6.36
Adat	✗	**4.88**

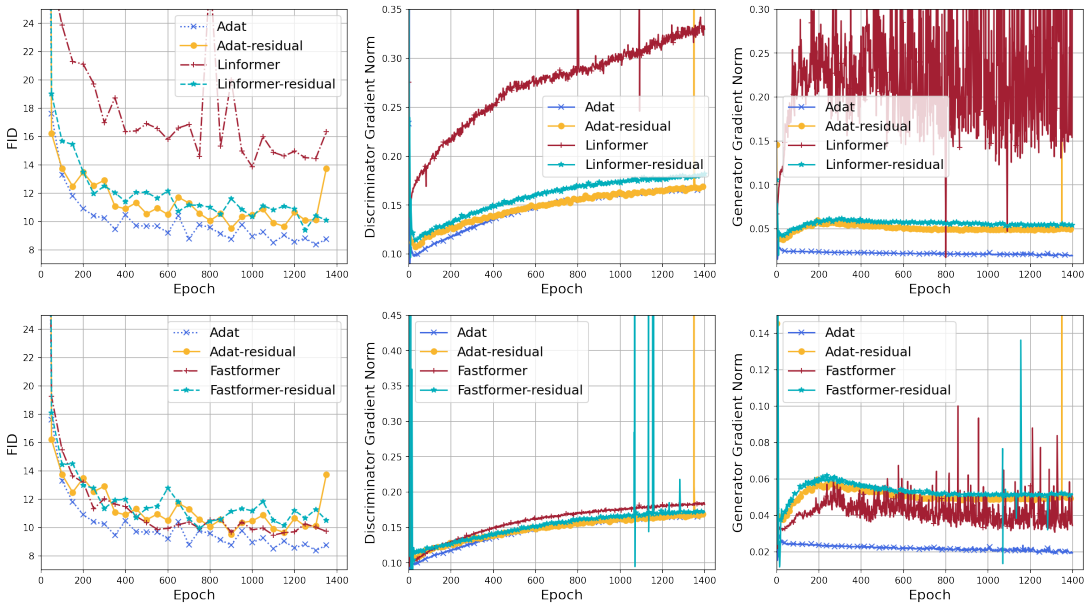


Figura 5.6: Magnitudes de gradientes sobre todos los parámetros y evaluación FID de la conexión residual del MLP en Linformer y Adat.

5.7. Conexiones residuales y modulación

Debido a que el efecto de las conexiones residuales en el generador de AdatGAN puede resultar en una diferencia significativa en la evaluación FID, en estos entrenamientos se evalúa el efecto de la modulación con la conexión residual del MLP. Como se muestra en la figura 5.7, agregar o eliminar la modulación del generador Transformador no resulta en inestabilidad en las magnitudes de los gradientes del discriminador. Sin embargo, la evaluación FID mejora sorprendentemente con la modulación si el Transformador de atención aditiva no cuenta con la conexión residual del MLP, como se meustra en la tabla 5.4.

Tabla 5.4: Evaluación FID de la conexión residual del MLP y la modulación en el generador sobre 50k imágenes de CIFAR-10.

Modulación	**Residual-MLP**	**FID ↓**
✓	✓	6.36
✗	✓	6.31
✗	✗	5.95
✓	✗	**4.88**

5.8. Discriminador de atención aditiva

En la figura 5.8 y en la tabla 5.5 se muestra una comparación del discriminador convolucional con el discriminador de AdatGAN. Como se observa en la figura 5.8, la atención aditiva permite entrenar un discriminador Transformador con regularización

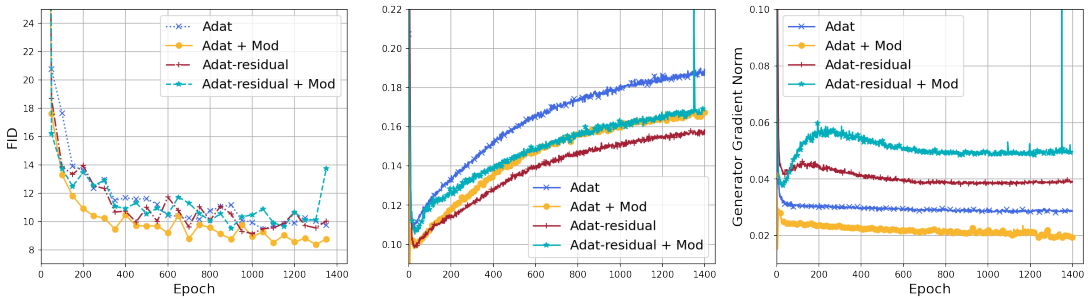


Figura 5.7: Magnitudes de gradientes sobre todos los parámetros y evaluación FID de la conexión residual del MLP y la modulación.

R1 y gradientes estables. Para este experimento, se agregó normalización por lotes (Ioffe and Szegedy, 2015) al discriminador de FastGAN (Liu et al., 2020). Aunque la normalización por lotes no mejora la evaluación FID de los discriminadores convolucionales, este método es esencial para complementar la estabilidad de la atención aditiva. Como se muestra en la tabla 5.5, el discriminador de Adat descrito en la sección 4.5 obtiene los mejores resultados. Además, el pequeño aumento de FLOPs en el discriminador permite procesar secuencias de longitud 256 manteniendo bajo el requerimiento de cómputo.

Tabla 5.5: FLOPs y evaluación FID del discriminador de AdatGAN en 50k imágenes de CIFAR-10.

Discriminador	**BN**	**FLOPs**	**FID** ↓
Conv-D	✗	0.5B	4.88
Conv-D-BN	✓	0.5B	4.97
Adat-D	✓	0.7B	**4.36**

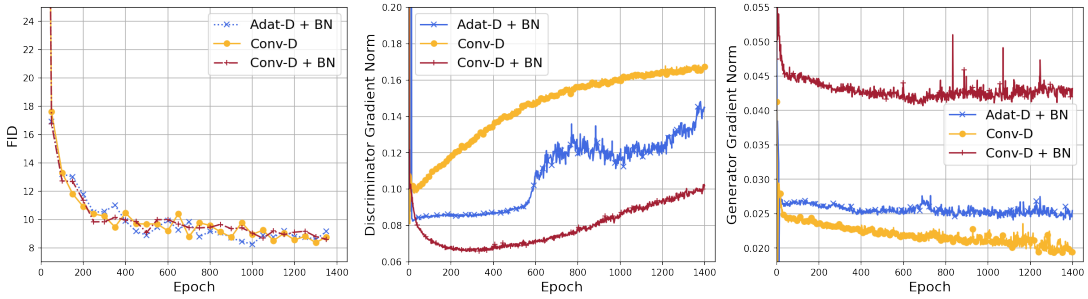


Figura 5.8: Magnitudes de gradientes sobre todos los parámetros y evaluación FID del discriminador de AdatGAN.

Expansión local del *embedding* En la tabla 5.6 se muestra el efecto de LEE en el generador y en el discriminador. Al aumentar la dimensión del último Transformador del generador de 64 a 128, los FLOPs aumentan de 0.7B a 0.9B. Esto provoca que la evaluación FID mejore, manteniendo el mismo comportamiento en los gradientes como

Tabla 5.6: FLOPs y evaluación FID de la expansión local del *embedding* en AdatGAN en 50k imágenes de CIFAR-10.

LEE	G-FLOPs	D-FLOPs	D-STD	bCR	FID ↓
✗	0.7B	0.7B	✓	✗	6.95
✗	0.7B	0.7B	✓	✓	4.33
G	0.9B	0.7B	✓	✗	6.21
G	0.9B	0.7B	✓	✓	**4.04**
D	0.7B	1.2B	✗	✗	6.75

se observa en la figura 5.9. Por otro lado, al aumentar la dimensión del *embedding* del discriminador a 512, es necesario eliminar la operación STD (por las siglas en inglés, *space-to-depth*) a la salida del Transformador. Esto evita generar demasiados canales y también resulta en una mejora en la evaluación FID. Este experimento también permite cuantificar la compatibilidad de las convoluciones con los Transformadores de atención aditiva y la operación STD en el discriminador. Los experimentos se realizan con tamaño de lote 64.

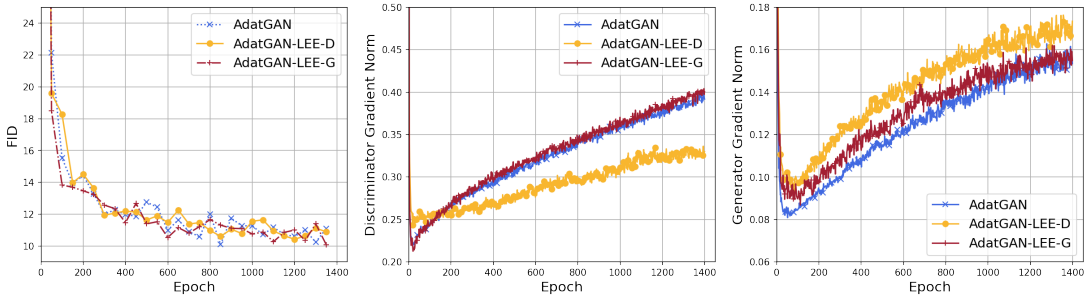


Figura 5.9: Magnitudes de gradientes sobre todos los parámetros y evaluación FID de la expansión local del *embedding* sin bCR.

Reducción del tamaño de lote Entrenar GANs con bajos requerimientos de cómputo requiere reducir el tamaño del lote. Las arquitecturas que mantienen su evaluación FID al reducir el tamaño del lote pueden aprovecharse al momento de escalar a altas resoluciones (imágenes de 1024×1024). En la tabla 5.7 se muestra que al reducir el tamaño del lote de 128 a 64 no hay cambios significativos en la evaluación FID. En la figura 5.10 se observa un ligero cambio en el comportamiento de los gradientes del discriminador, sin embargo, estos continúan siendo estables.

Tabla 5.7: Evaluación FID del tamaño de lote de AdatGAN en 50k imágenes de CIFAR-10.

Tamaño de lote	FID ↓
128	4.36
64	**4.33**

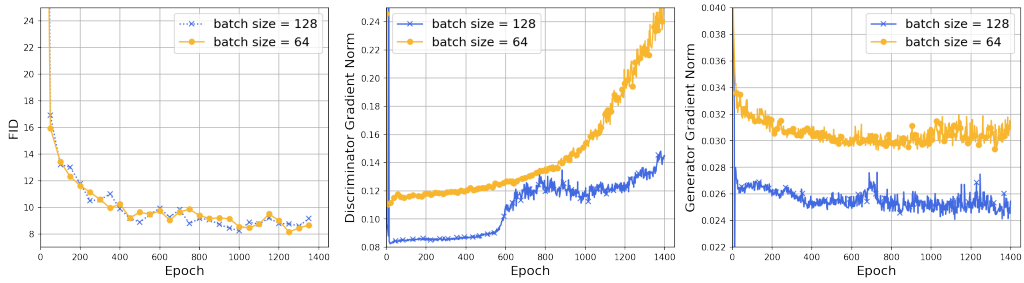


Figura 5.10: Magnitudes de gradientes sobre todos los parámetros y evaluación FID del tamaño de lote de AdatGAN.

5.9. Regularización

En esta sección se muestra el efecto de bCR (Zhao et al., 2021b) en los discriminadores convolucionales y los discriminadores Transformadores. Al igual que en VITGAN y HiT, la regularización en Transformadores mejora la evaluación FID más que en los modelos convolucionales, como se muestra en la tabla 5.8. En la figura 5.11 no se observa que la regularización altere la estabilidad de los gradientes.

Tabla 5.8: Evaluación FID de regularización en 50k imágenes de CIFAR-10. *Resultados del artículo de Lee et al. (2021).

Generador	**Discriminador**	**bCR**	**FID ↓**
VITGAN	VITGAN	✗	8.84*
VITGAN	VITGAN	✓	6.66*
AdatGAN	Conv	✗	5.97
AdatGAN	Conv	✓	4.88
AdatGAN-LEE-G	Adat	✗	6.21
AdatGAN-LEE-G	Adat	✓	**4.04**

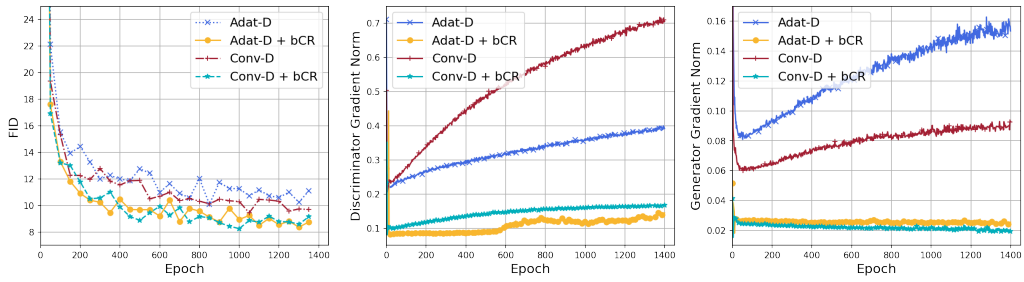


Figura 5.11: Magnitudes de gradientes sobre todos los parámetros y evaluación FID de bCR.

En la tabla 5.9 se muestra una comparación de la regularización y los componentes de las arquitecturas de VITGAN, Vanilla-ViT y AdatGAN. Vanilla-ViT es similar a AdatGAN, ya que ambos se entrenan con la regularización R1. Por otro lado, la

principal diferencia con VITGAN es el uso de INRs. Aunque AdatGAN no utiliza INRs, los resultados de los mapas de atención aditiva consiguen una evaluación similar a las INRs.

Tabla 5.9: Comparación de regularización, arquitectura y FID de VITGAN, Vanilla-ViT y AdatGAN en CIFAR-10. *Resultados del artículo de Lee et al. (2021).

Método	**VITGAN**	**Vanilla-ViT**	**AdatGAN**
ISN (Lee et al., 2021)	✓	✗	✗
R_1 (Mescheder et al., 2018)	✗	✓	✓
bCR (Zhao et al., 2021b)	✓	✓	✓
L_2	✓	✗	✗
INRs	✓	✗	✗
Convoluciones	✓	✗	✓
Complejidad de atención	$O(N^2 \cdot d)$	$O(N^2 \cdot d)$	$O(N \cdot d)$
Longitud de secuencia	64	64	1024
FID ↓	4.92*	12.70*	**4.04**

5.10. Aumento de datos

En la tabla 5.10 y en la figura 5.12, se muestra una comparación con StyleGAN2 (Karras et al., 2020) y el aumento diferenciable (Zhao et al., 2020). Estos experimentos se realizan con un tamaño de lote de 64 y sin LEE. Un aspecto importante a considerar es que a diferencia de AdatGAN, el discriminador de StyleGAN2 es convolucional. Debido a esto, en conjuntos de datos pequeños, StyleGAN2 consigue mejores resultados. Sin embargo, a medida que aumenta la cantidad de datos, AdatGAN tiende a superar la evaluación FID de StyleGAN2. Esto se debe a que, al igual que en HiT (Zhao et al., 2021a) y VITGAN (Lee et al., 2021), los Transformadores funcionan mejor que los modelos convolucionales cuando se tienen grandes cantidades de datos. Por lo tanto, para estos escenarios, AdatGAN es la mejor opción en términos de capacidad y eficiencia.

Tabla 5.10: Evaluación FID de aumento diferenciable en 50k imágenes de entrenamiento y 10k imágenes de prueba de CIFAR-10. *Resultados del artículo de Zhao et al. (2020).

Arquitectura	100 % data		20 % data		10 % data	
	50k	10k	50k	10k	50k	10k
StyleGAN2 (Karras et al., 2020)	5.79*	9.89*	-	**12.15***	-	**14.50***
AdatGAN	**4.33**	8.60	9.07	12.99	12.13	16.39

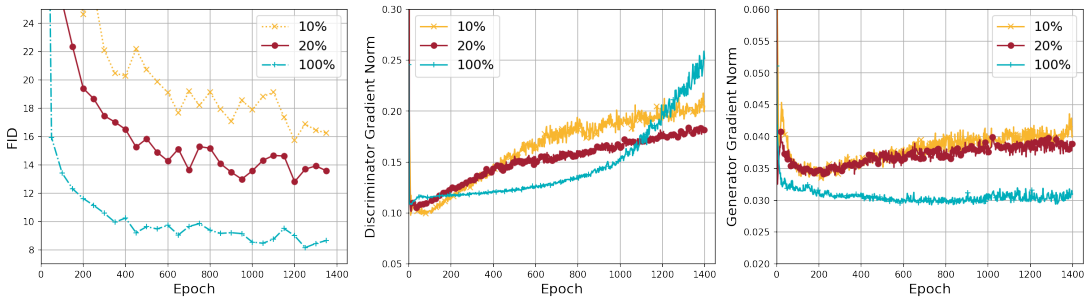


Figura 5.12: Magnitudes de gradientes sobre todos los parámetros y evaluación FID del aumento diferenciable en AdatGAN.

5.11. Resultados principales

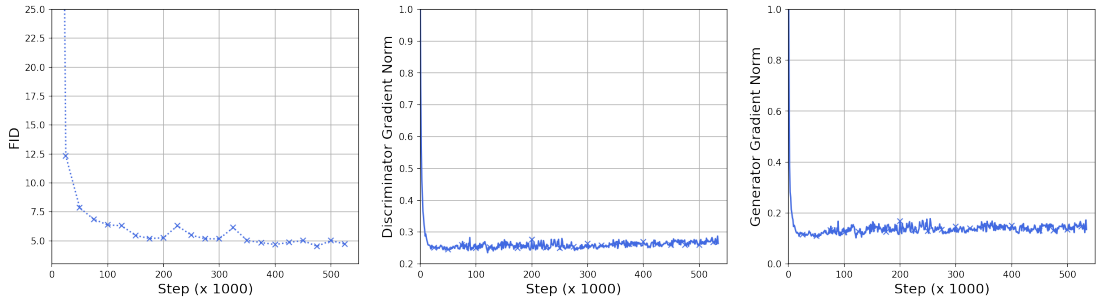
Tabla 5.11: Evaluación FID en CIFAR-10 (32×32) y CelebA (64×64). *Resultados de los artículos originales.

Arquitectura	**CIFAR-10**	**CelebA**
StyleGAN2 + DiffAugment (Zhao et al., 2020)	5.79*	-
TransGAN (Jiang et al., 2021)	9.02*	-
Vanilla-ViT (Lee et al., 2021)	12.70*	20.20*
VITGAN (Lee et al., 2021)	4.92*	3.74*
BigGAN + DiffAugment (Zhao et al., 2020)	4.61*	-
StyleGAN2-D+ViTGAN-G (Lee et al., 2021)	4.57*	-
AdatGAN	**4.04**	**3.60**

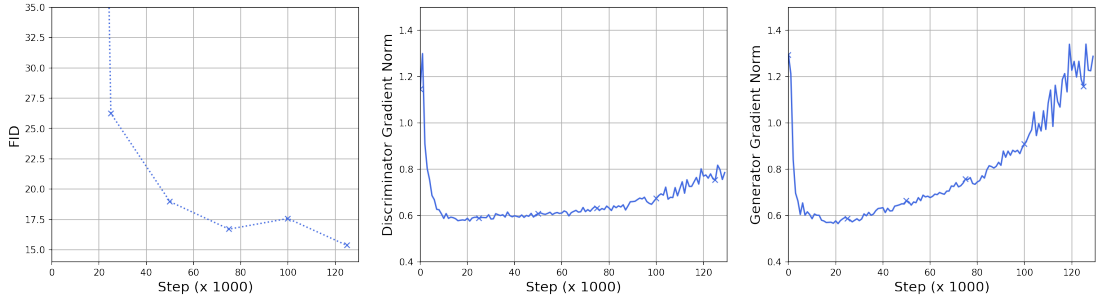
En la tabla 5.11 se muestran los resultados principales. AdatGAN se compara con Transformadores GANs como VITGAN (Lee et al., 2021) y TransGAN (Jiang et al., 2021). Para el conjunto de datos de CelebA, se generan parches de 2×2 y se incrementa la dimensión del último Transformador del generador de 64 a 128 mediante el uso de LEE, resultando en el mismo generador que se utiliza en CIFAR-10. En el discriminador, se agrega un bloque residual de FastGAN para reducir los mapas de características a 256 *tokens*, igual que en CIFAR-10, y se utiliza aumento de datos de Traslación y Color. Además, no se utiliza bCR. (Zhao et al., 2021b).

Aunque no se utiliza bCR, AdatGAN consigue una buena evaluación FID en CelebA debido al tamaño de este conjunto de datos en comparación con CIFAR-10. Como se muestra en los experimentos de aumento de datos, AdatGAN funciona mejor con conjuntos de datos grandes. En la figura 5.13, se observa un comportamiento estable de los gradientes al generar parches. AdatGAN es un modelo que consigue una buena evaluación FID tanto en la generación de parches como en la de píxeles.

En las figuras 5.14, 5.15, 5.16 y 5.17 se muestran ejemplos de imágenes generadas por AdatGAN, así como los mapas de atención aditiva de un solo cabezal en las resoluciones 32×32, 16×16 y 8×8. En la resolución del mapa de atención aditiva más cercana a la variable latente, que es 8×8, el mapa de atención aprende a utilizar un solo *token* como un acceso de memoria al vector latente. Por otro lado, en la



(a) CelebA



(b) LSUN Bedroom

Figura 5.13: Magnitudes de gradientes sobre todos los parámetros y evaluación FID de CelebA (64×64) y LSUN Bedroom (128×128).

resolución 16×16, el mapa de atención aditiva genera una estructura global para luego complementar la información con el mapa de atención aditiva de 32×32.

5.11.1. Muestras de alta resolución

En las figuras 5.16 y 5.17, se muestran los mapas de atención aditiva y las imágenes generadas por AdatGAN entrenado en LSUN Bedroom y FFHQ en 128×128. Para esta resolución, en el generador se incrementa la dimensión de los Transformadores en 16×16 y 32×32 a 512 con LEE. Para el discriminador en LSUN Bedroom, solo se agrega un bloque convolucional para procesar 256 *tokens*. Este modelo consigue un FID de 10.64 con tamaño de lote 64 y sin bCR. Como se muestra en la figura 5.18, al realizar interpolación entre dos muestras con AdatGAN, se produce un cambio suave de estilo. Para el discriminador en FFHQ, se encontró que al incrementar la dimensión del Transformador a 512 y utilizar un solo bloque de FastGAN con tamaño de $kernel = 8$, se consigue una mejor evaluación FID. Este modelo consigue un FID de 6.10.

5.11.2. Análisis de costo computacional

En las tablas 5.12 y 5.13 se muestra la eficiencia de AdatGAN en FLOPs y en número de parámetros en comparación con VITGAN y StyleGAN2. AdatGAN es el modelo más eficiente en FLOPs requiriendo solo el 11.53 % de StyleGAN2 y el 34.61 % de VITGAN para la resolución 64×64, y $\sim$27.8 % para la resolución 128×128.

Tabla 5.12: Costo computacional de los generadores para la resolución 64×64. *Resultados del artículo de Lee et al. (2021).

Arquitectura	FLOPs@64^2	#Params@64^2
StyleGAN2 (Karras et al., 2020)	7.8B*	24M*
VITGAN (Lee et al., 2021)	2.6B*	38M*
AdatGAN	**0.9B**	**20M**

Tabla 5.13: Costo computacional de los generadores para la resolución 128×128. *Resultados del artículo de Lee et al. (2021).

Arquitectura	FLOPs@128^2	#Params@128^2
StyleGAN2 (Karras et al., 2020)	11.5B*	-
VITGAN (Lee et al., 2021)	11.8B*	-
AdatGAN	**3.2B**	**26M**

(a) (b) (c) (d)

Figura 5.14: (a) Imágenes generadas de CIFAR-10 (32×32). Mapas de atención aditiva (b) 32×32, (c) 16×16 y (d) 8×8.

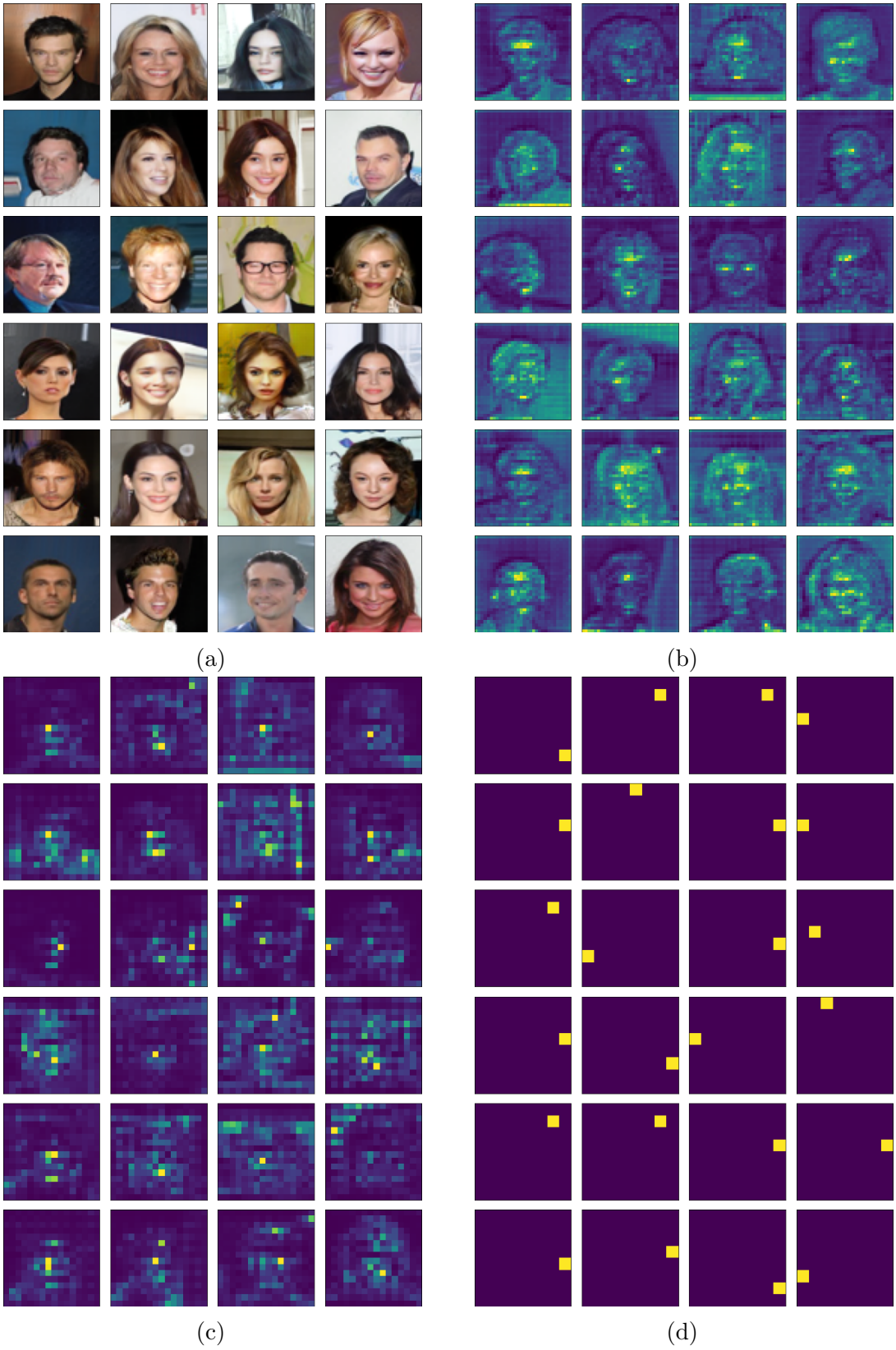


Figura 5.15: (a) Imágenes generadas de CelebA (64×64). Mapas de atención aditiva (b) 32×32, (c) 16×16 y (d) 8×8.

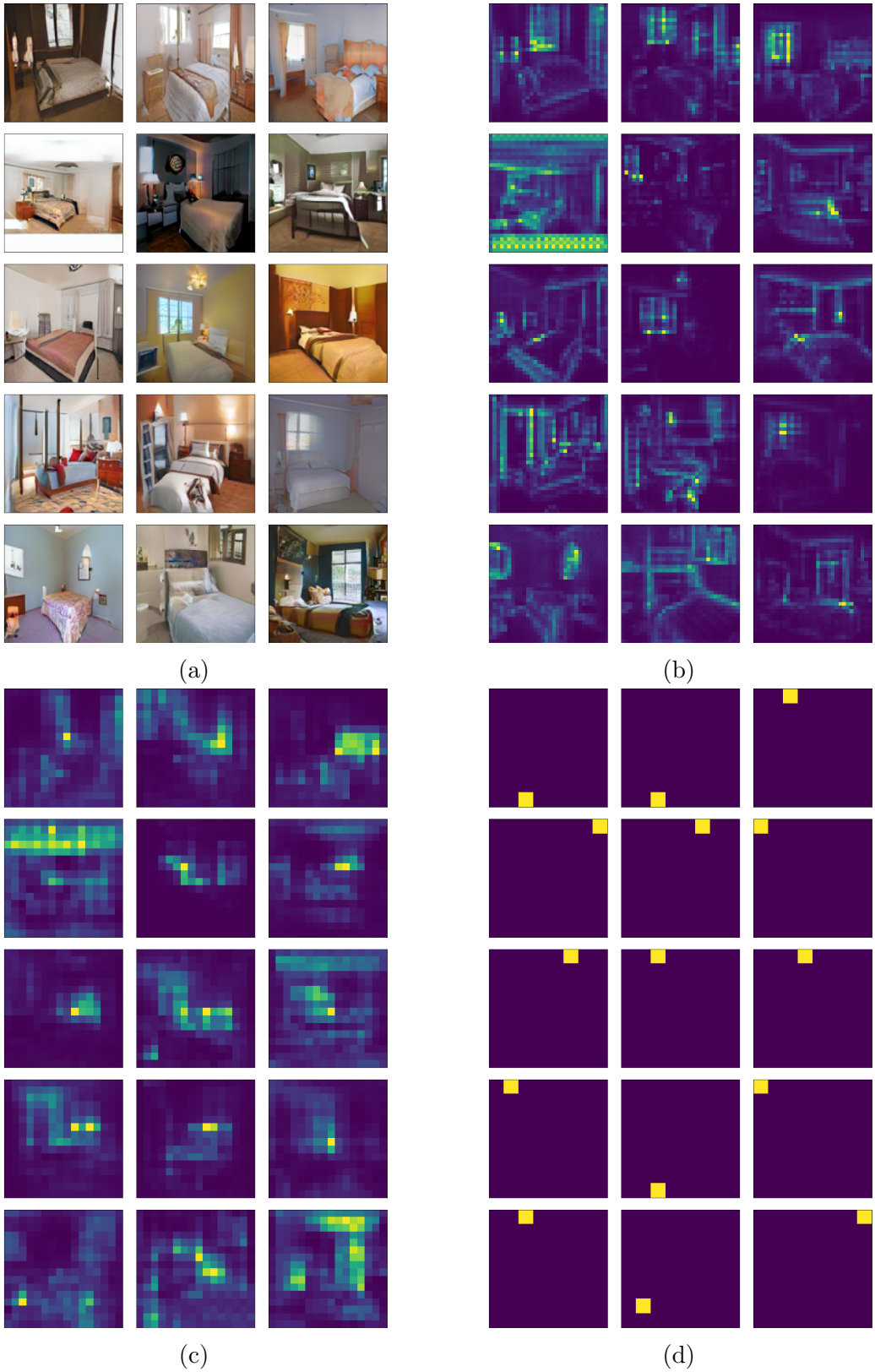


Figura 5.16: (a) Imágenes generadas de LSUN Bedroom (128 × 128). Mapas de atención aditiva (b) 32 × 32, (c) 16 × 16 y (d) 8 × 8.

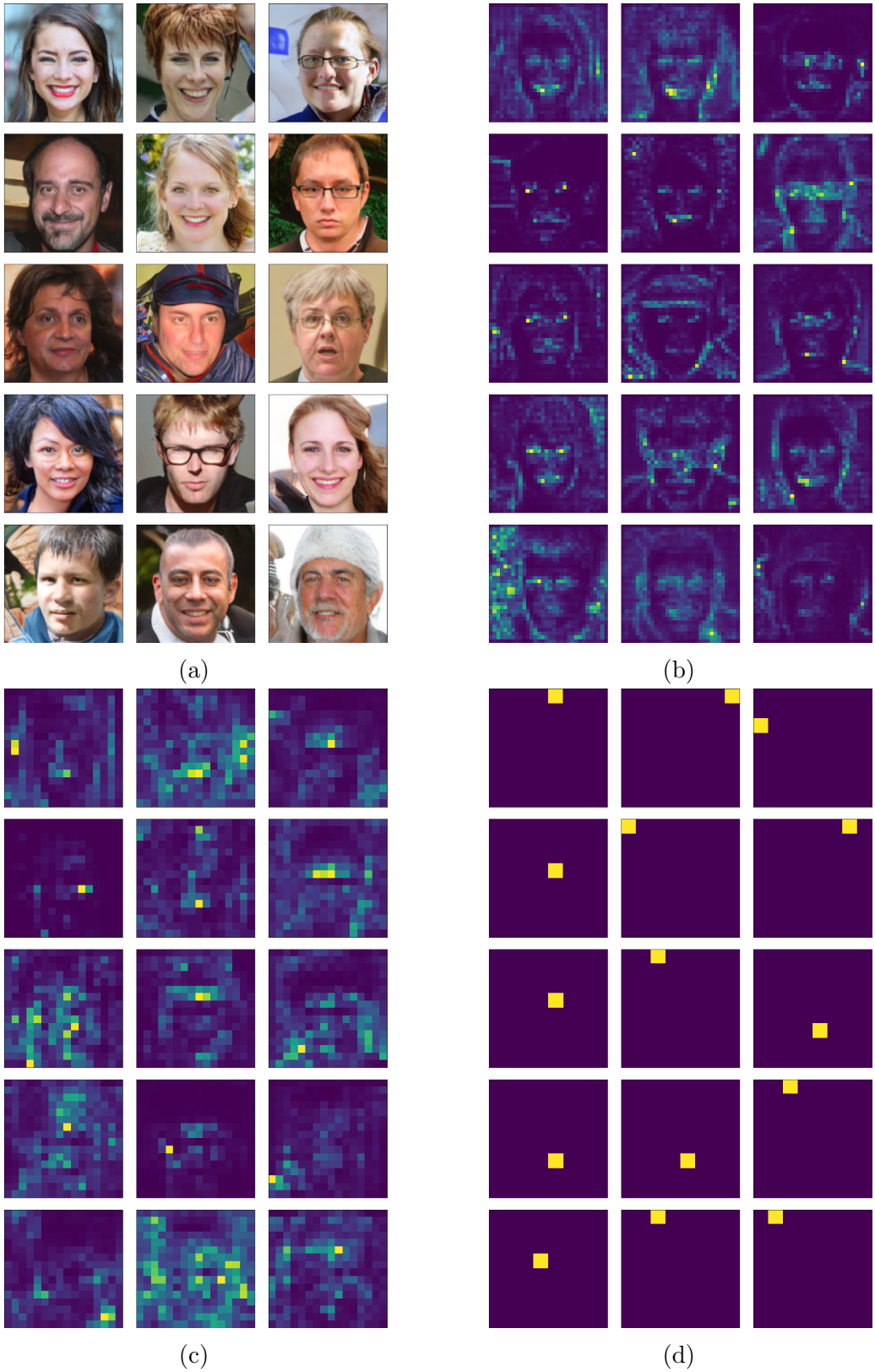


Figura 5.17: (a) Imágenes generadas de FFHQ (128×128). Mapas de atención aditiva (b) 32×32, (c) 16×16 y (d) 8×8.

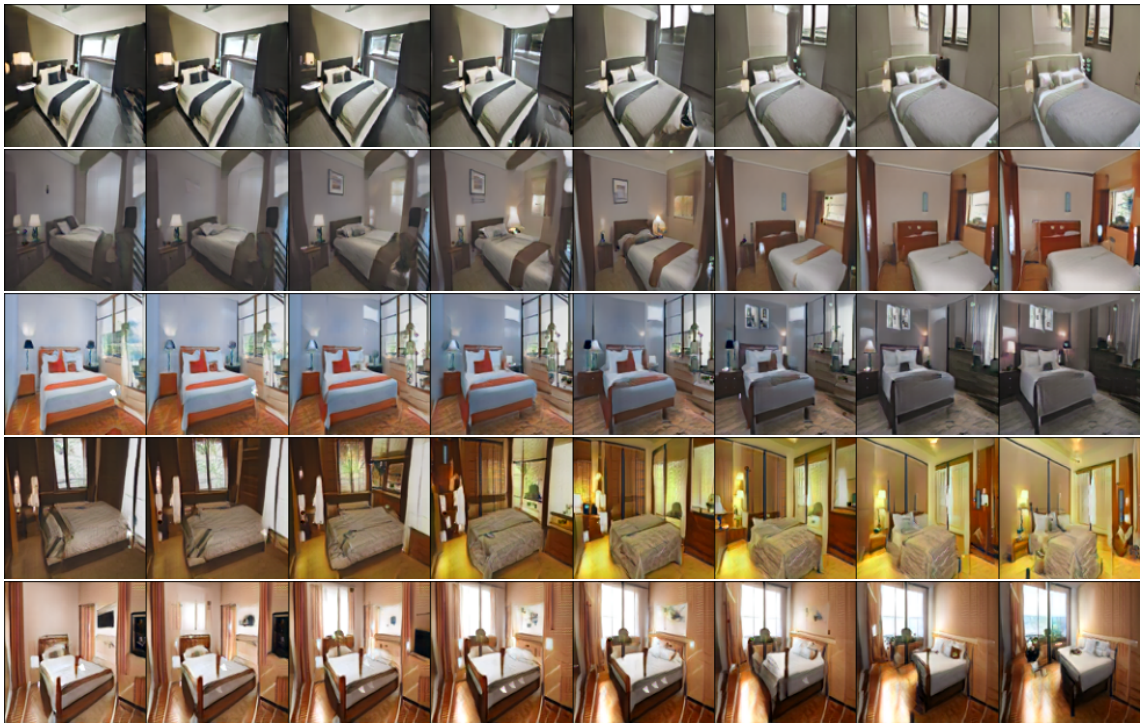


Figura 5.18: Interpolación en el espacio latente en LSUN Bedroom.

Figura 5.19: Interpolación en el espacio latente en FFHQ.

Capítulo 6

Conclusiones

En este trabajo se presentan los Transformadores antagónicos generativos de atención aditiva (AdatGAN). La exploración de diferentes conexiones residuales, la modulación y los mecanismos de atención permitieron desarrollar Transformadores generativos eficientes que requieren en términos de FLOPs el 34.61 % de VITGAN para la resolución de 64×64 y $\sim$27.8 % para la resolución de 128×128. Los Transformadores y la atención aditiva propuesta muestran un mejor desempeño en tareas generativas que los Transformadores de autoatención producto punto como Linformer y Swin Transformer, y de atención aditiva como Fastformer.

AdatGAN consigue una evaluación FID similar a VITGAN y StyleGAN de manera más eficiente. La generación de secuencias largas con atención aditiva obtiene resultados competitivos en comparación con las INRs de VITGAN, sin embargo, no se descarta la posibilidad de utilizar ambas en conjunto.

El uso de convoluciones con Transformadores en los generadores muestra cambios significativos en la evaluación FID dependiendo del mecanismo de atención utilizado en la arquitectura. En este trabajo, se demuestra la efectividad de utilizar Transformadores de atención aditiva con convoluciones en tareas generativas. Además, el método LLE permite aprovechar de manera más eficiente la operación de *pixel shuffle*, lo que también mejora la evaluación FID.

Para estabilizar los entrenamientos, se propone utilizar discriminadores Transformadores eficientes compatibles con penalización de gradientes, a diferencia de VITGAN, HiT, StyleSwin y Styleformer, que utilizan un discriminador convolucional para conseguir resultados de estado del arte. Debido a la mayor capacidad de los discriminadores Transformadores comparados con los discriminadores convolucionales, AdatGAN consigue una evaluación FID competitiva y resulta en generadores más eficientes en FLOPs que los modelos de estado del arte.

6.1. Trabajo futuro

Una vez conseguidos y explorados módulos y arquitecturas de generadores eficientes, se puede proceder a explorar con arquitecturas Transformadores estrategias autosupervisadas como FastGAN (Chen et al., 2019b; Liu et al., 2020) y sus efectos con penalizaciones de gradientes. Asimismo, el uso de INRs como en CIPS (Anokhin et al., 2021) y VITGAN (Lee et al., 2021), estrategias de preentrenamiento como en

VQ-GAN (Esser et al., 2021) y Stable Diffusion (Rombach et al., 2022), y transferencia de aprendizaje como en Projected GAN (Sauer et al., 2021), ahora se pueden explorar de manera viable.

Además, la capacidad de los Transformadores generativos de atención aditiva junto con las convoluciones queda por ser explorada fuera del marco de trabajo de GANs, como en la clasificación de imágenes (Dosovitskiy et al., 2021) y en modelos de difusión (Ho et al., 2020; Song et al., 2020; Dhariwal and Nichol, 2021; Rombach et al., 2022). Sin embargo, es importante mencionar que en estos últimos, la inferencia es un proceso muy lento en comparación con GANs, por lo que si se busca una arquitectura eficiente tanto en entrenamiento como en inferencia, puede ser una mejor opción optar por alternativas híbridas (Wang et al., 2022), aprovechando la eficiencia en inferencia de GANs y la estabilidad de los entrenamientos de los modelos de difusión, o utilizar un enfoque completamente diferente como el modelado de imágenes enmascaradas (Chang et al., 2023).

Por último, debido a que la arquitectura muestra eficiencia y capacidad para entrenarse en grandes conjuntos de datos, el siguiente paso es realizar entrenamientos en altas resoluciones y en datos masivos en tareas como transformación de texto a imagen (Sauer et al., 2022; Schuhmann et al., 2021; Rombach et al., 2022; Chang et al., 2023), síntesis de voz (Wang et al., 2017; Kumar et al., 2019; Bińkowski et al., 2019; Kong et al., 2020), y generación de música (Dhariwal et al., 2020), empleando y aprovechando los métodos mencionados anteriormente.

Referencias

Anokhin, I., Demochkin, K., Khakhulin, T., Sterkin, G., Lempitsky, V., and Korzhenkov, D. (2021). Image generators with conditionally-independent pixel synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14278–14287. 20, 32, 59

Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*. 15, 23

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *ICML*. 20, 23

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*. 30

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. 15, 16, 28, 35

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*. 19

Bińkowski, M., Donahue, J., Dieleman, S., Clark, A., Elsen, E., Casagrande, N., Cobo, L. C., and Simonyan, K. (2019). High fidelity speech synthesis with adversarial networks. In *International Conference on Learning Representations*. 15, 60

Brock, A., Donahue, J., and Simonyan, K. (2019). Large scale gan training for high fidelity natural image synthesis. In *ICLR*. 16, 19, 28, 34

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901. 15, 16, 28, 31

Chang, H., Zhang, H., Barber, J., Maschinot, A., Lezama, J., Jiang, L., Yang, M.-H., Murphy, K., Freeman, W. T., Rubinstein, M., et al. (2023). Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*. 60

Chen, T., Lucic, M., Houlsby, N., and Gelly, S. (2019a). On self modulation for generative adversarial networks. In *ICLR*. 28, 36

Chen, T., Zhai, X., Ritter, M., Lucic, M., and Houlsby, N. (2019b). Self-supervised gans via auxiliary rotation loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12154–12163. 59

Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*. 16, 19

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL*. 30, 40

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. (2020). Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*. 60

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794. 60

Dong, Y., Cordonnier, J.-B., and Loukas, A. (2021). Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pages 2793–2803. PMLR. 32

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*. 15, 16, 28, 29, 36, 60

Esser, P., Rombach, R., and Ommer, B. (2021). Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883. 60

Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*. 28

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680. 15, 19, 21, 22, 24

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*. 15

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777. 20, 24, 25, 29

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30. 43

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851. 60

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. 28

Huang, X. and Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510. 27, 28

Hudson, D. A. and Zitnick, L. (2021). Generative adversarial transformers. In *International conference on machine learning*, pages 4487–4499. PMLR. 20

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR. 27, 40, 44, 48

Jiang, Y., Chang, S., and Wang, Z. (2021). Transgan: Two transformers can make one strong gan. *arXiv preprint arXiv:2102.07074*. 16, 19, 29, 34, 37, 52

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer. 28

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410. 16, 19, 25, 26, 28, 31, 42, 43

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119. 16, 19, 20, 25, 28, 34, 43, 51, 54

Kim, H., Papamakarios, G., and Mnih, A. (2021). The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pages 5562–5571. PMLR. 32, 35

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 43

Kitaev, N., Kaiser, L., and Levskaya, A. (2020). Reformer: The efficient transformer. In *International Conference on Learning Representations*. 16, 19

Kong, J., Kim, J., and Bae, J. (2020). Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033. 15, 60

Krizhevsky, A. et al. (2009). Learning multiple layers of features from tiny images. 11, 41

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brébisson, A., Bengio, Y., and Courville, A. C. (2019). Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32. 15, 60

Lee, K., Chang, H., Jiang, L., Zhang, H., Tu, Z., and Liu, C. (2021). Vitgan: Training gans with vision transformers. In *International Conference on Learning Representations*. 9, 15, 16, 19, 26, 30, 31, 32, 34, 35, 36, 37, 40, 43, 44, 50, 51, 52, 54, 59

Lim, J. H. and Ye, J. C. (2017). Geometric gan. *arXiv preprint arXiv:1705.02894*. 26

Lin, Z., Sekar, V., and Fanti, G. (2021). Why spectral normalization stabilizes gans: Analysis and improvements. *Advances in neural information processing systems*, 34:9625–9638. 26, 44

Liu, B., Zhu, Y., Song, K., and Elgammal, A. (2020). Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. In *International Conference on Learning Representations*. 26, 40, 44, 48, 59

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022. 11, 16, 19, 31, 32, 35, 44, 45

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*. 11, 41, 42

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022). A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986. 15

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*. 28

Mescheder, L., Geiger, A., and Nowozin, S. (2018). Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR. 25, 26, 43, 44, 51

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer. 32

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *ICLR*. 20, 26

Mokady, R., Tov, O., Yarom, M., Lang, O., Mosseri, I., Dekel, T., Cohen-Or, D., and Irani, M. (2022). Self-distilled stylegan: Towards generation from internet photos. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9. 15

Park, J. and Kim, Y. (2022). Styleformer: Transformer based generative adversarial networks with style vector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8983–8992. 15, 31, 32, 35, 40

Park, N. and Kim, S. (2021). How do vision transformers work? In *International Conference on Learning Representations*. 20, 37, 40

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR. 28

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*. 26, 28

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. 15, 16, 28, 31

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67. 15, 28

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695. 60

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242. 23

Sauer, A., Chitta, K., Müller, J., and Geiger, A. (2021). Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492. 60

Sauer, A., Schwarz, K., and Geiger, A. (2022). Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10. 15, 60

Schuhmann, C., Vencu, R., Beaumont, R., Kaczmarczyk, R., Mullis, C., Katta, A., Coombes, T., Jitsev, J., and Komatsuzaki, A. (2021). Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*. 60

Seitzer, M. (2020). pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid. Version 0.2.1. 43

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423. 21

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883. 29, 37

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473. 32

Song, J., Meng, C., and Ermon, S. (2020). Denoising diffusion implicit models. In *International Conference on Learning Representations*. 60

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27. 15, 28

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826. 43

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547. 32

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2020). Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*. 31, 32

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2021a). Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR. 15

Touvron, H., Cord, M., El-Nouby, A., Bojanowski, P., Joulin, A., Synnaeve, G., and Jégou, H. (2021b). Augmenting convolutional networks with attention-based aggregation. *arXiv preprint arXiv:2112.13692*. 20, 37

Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., and Li, Y. (2022). Maxvit: Multi-axis vision transformer. *arXiv preprint arXiv:2204.01697*. 15

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*. 27

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30. 15, 28, 36, 46

Villani, C. (2009). *Optimal transport: old and new*, volume 338. Springer. 24

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*. 16, 19, 31, 38, 44, 45, 46

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, Z. Y., Xiao, Y., Chen, Z., Bengio, S., Le, Q., et al. (2017). Tacotron: Towards end-to-end speech synthesis. 60

Wang, Z., Zheng, H., He, P., Chen, W., and Zhou, M. (2022). Diffusion-gan: Training gans with diffusion. *arXiv preprint arXiv:2206.02262*. 60

Wu, C., Wu, F., Qi, T., Huang, Y., and Xie, X. (2021a). Fastformer: Additive attention can be all you need. *arXiv preprint arXiv:2108.09084*. 35, 36, 44, 45, 46

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L. (2021b). Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31. 19, 20, 37

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. (2021). Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148. 20

Yu, F., Zhang, Y., Song, S., Seff, A., and Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*. 11, 42

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.-H., Tay, F. E., Feng, J., and Yan, S. (2021). Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567. 15

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. (2020). Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297. 16, 19

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2022). Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113. 15

Zhang, B., Gu, S., Zhang, B., Bao, J., Chen, D., Wen, F., Wang, Y., and Guo, B. (2022). Styleswin: Transformer-based gan for high-resolution image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11304–11314. 16, 20, 25, 31, 32, 34, 35, 37, 40, 43, 44

Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2019). Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR. 15, 29

Zhao, L., Zhang, Z., Chen, T., Metaxas, D., and Zhang, H. (2021a). Improved transformer for high-resolution gans. *Advances in Neural Information Processing Systems*, 34. 16, 20, 25, 32, 34, 35, 37, 40, 43, 50, 51

Zhao, S., Liu, Z., Lin, J., Zhu, J.-Y., and Han, S. (2020). Differentiable augmentation for data-efficient gan training. In *NeurIPS*. 9, 33, 34, 43, 51, 52

Zhao, Z., Singh, S., Lee, H., Zhang, Z., Odena, A., and Zhang, H. (2021b). Improved consistency regularization for gans. In *AAAI*. 34, 50, 51, 52

Zhu, C., Ping, W., Xiao, C., Shoeybi, M., Goldstein, T., Anandkumar, A., and Catanzaro, B. (2021). Long-short transformer: Efficient transformers for language and vision. *Advances in Neural Information Processing Systems*, 34. 19