UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS FÍSICAS

DEPARTAMENTO DE ESTRUCTURA DE LA MATERIA, FÍSICA TÉRMICA Y ELECTRÓNICA

TRABAJO DE FIN DE GRADO

Código de TFG: ETE29

Inteligencia artificial aplicada a física de astropartículas: introducción a las técnicas de aprendizaje profundo

Artificial intelligence applied to astroparticle physics: introduction to deep learning techniques

Supervisor/es: Daniel Nieto Castaño

Sergio García Heredia

Grado en Física

Curso académico 2021-2022

Convocatoria de junio

Índice de contenidos

Índice de algoritmos

Índice de figuras

Resumen

El aprendizaje profundo es un área del aprendizaje automático que está experimentando un enorme desarrollo en los últimos años y cuyas aplicaciones están revolucionando ámbitos muy diversos, entre ellos, el científico. Esta revolución ha llegado también al campo de la física y, entre otras ramas de la misma, a la física de altas energías. Hasta ahora, los modelos de aprendizaje supervisado (de regresión y clasificación) han mostrado una gran eficacia en el análisis masivo de datos. Un ejemplo de ello es el proyecto CTLearn, una librería programada en Python enfocada en el análisis de imágenes de telescopios Cherenkov de imagen atmosférica (IACTs). Sin embargo, el entrenamiento de estos modelos, y en general la caracterización de la respuesta de los IACTs, requiere de un gran volumen de datos etiquetados, algo que hasta ahora se obtiene mediante costosas simulaciones de Monte Carlo. Con el fin de reducir el coste computacional y aumentar el realismo de los datos simulados, en este proyecto se estudia la posibilidad de sustituir dichas simulaciones por lo que se denomina un modelo generativo de aprendizaje profundo. En concreto, se valora el potencial de las denominadas redes generativas adversarias o GANs para este propósito. Empleando un enfoque tipo AC-GANs, es decir, usando un modelo auxiliar para condicionar las GANs, se trata de replicar un conjunto de datos simulados a partir de sus etiquetas (tipo de partícula, energía y dirección de llegada). El nivel de realismo de las imágenes y su grado de correlación con sus etiquetas se evalúa haciendo uso de modelos de CTLearn sobre un conjunto de datos de validación. Como resultado, se obtienen unos tiempos de generación de imágenes varios órdenes de magnitud menores y un grado de realismo prometedor, pero una insuficiente correlación con las etiquetas, algo que podría solucionarse con un mejor ajuste de la configuración del modelo.

Palabras clave: Telescopios Cherenkov de imagen atmosférica, Aprendizaje profundo, Modelado generativo, Redes generativas adversarias.

Abstract

Deep learning is an area of machine learning that is experiencing an enormous development in recent years and whose applications are revolutionizing very diverse domains, including science. This revolution has also reached the field of physics and, among other branches of it, high energy physics. So far, supervised learning models (for regression and classification) have shown great efficiency in massive data analysis. An example of this is the CTLearn project, a library programmed in Python focused on the analysis of images from imaging atmospheric Cherenkov telescopes (IACTs). However, the training of these models, and in general the characterization of the response of IACTs, requires a large volume of labeled data, something that so far is obtained by expensive Monte Carlo simulations. In order to reduce the computational cost and increase the realism of the simulated data, the possibility of replacing such simulations with what is called a generative deep learning model is studied in this project. In particular, the potential of so-called generative adversarial networks or GANs for this purpose is assessed. Using an AC-GANs approach, that is, using an auxiliary model to condition the GANs, the aim is to replicate a set of simulated data from their labels (particle type, energy and arrival direction). The level of realism of the images and their degree of correlation with their labels is evaluated using CTLearn models on a validation data set. As a result, image generation times several orders of magnitude shorter and a promising degree of realism are obtained, but insufficient correlation with the labels, something that could be solved by better tuning of the model configuration.

Keywords: Imaging atmospheric Cherenkov telescopes, Deep learning, Generative modeling, Generative adversarial networks.

1. Introducción

1.1. Contexto del trabajo

En los últimos años, los enormes progresos en capacidad computacional y recolección de datos han permitido materializar la actual revolución del aprendizaje profundo, un área del aprendizaje automático que hace uso de grandes modelos de aprendizaje y masivas cantidades de datos para resolver tareas de todo tipo en una amplísima variedad de campos. Esta revolución ha llegado también al ámbito de la física y, entre otras ramas de la misma, a la física de altas energías[1]. Hasta ahora, el aprendizaje profundo se ha mostrado especialmente útil en tareas de regresión y clasificación de datos, haciendo uso de conjuntos de datos de entrenamiento etiquetados. Ejemplo de ello es el proyecto CTLearn [4], una librería de Python enfocada en el análisis de imágenes de telescopios Cherenkov de imagen atmosférica (IACTs). Los IACTs surgieron en la década de 1980 y su objetivo es el estudio de fuentes de rayos gamma mediante la radiación Cherenkov que emite la cascada de partículas que estos rayos (fotones), al igual que los rayos cósmicos (protones principalmente), desencadenan al llegar a la atmósfera terrestre [16]. Más recientemente, debido a sus impresionantes resultados en otras áreas, se ha empezado ha explorar en física el potencial de los denominados modelos generativos de aprendizaje profundo, aquellos que no realizan predicciones sobre los datos sino que aprenden la distribución de los mismos.

1.2. Motivación

La caracterización de la respuesta de los IACTs y el entrenamiento de los modelos de CTLearn requieren grandes conjuntos de datos etiquetados, que hasta ahora se obtienen mediante simulaciones de Monte Carlo (véase, por ejemplo, [3]). Dos grandes inconvenientes de estas simulaciones son su gran coste computacional y su limitado grado de realismo. En este proyecto, se estudia la posibilidad de emplear un modelo generativo de aprendizaje profundo para resolver alguno de estos problemas o incluso ambos. Este es un enfoque que se ha estudiado ampliamente en el ámbito de la física de altas energías, en particular, en aceleradores de partículas, donde la dependencia de grandes simulaciones es enorme.

En el presente trabajo se propone el uso de las denominadas redes generativas adversarias, más conocidas como GANs (Generative Adversarial Networks), para sustituir o complementar las simulaciones. Las GANs fueron presentadas en [14], y desde entonces han ido perfeccionándose hasta convertirse en el enfoque estándar en la mayoría de tareas de modelado generativo. La elección de las GANs para este proyecto está motivada por su probada eficacia dentro y fuera del ámbito de la física, su facilidad para generar muestras una vez estimada la distribución de los datos de entrenamiento, la gran versatilidad de su diseño, y su teórica robustez frente a la mera copia de los datos de entrenamiento.

1.3. Objetivos

El objetivo de este proyecto es implementar un modelo básico de GANs condicionales, tipo AC-GANs, y evaluar su capacidad para generar imágenes etiquetadas de IACTs.

1.4. Estructura del trabajo

La memoria de este proyecto se estructura de la siguiente forma: en la sección 2 se hace un repaso del trabajo realizado hasta ahora por otros autores en relación con este problema, en la sección 3 se da una detallada explicación de las GANs a modo de marco teórico, en la sección 4 se expone la metodología seguida para la realización del proyecto, en la sección 5 se presentan y analizan los resultados del trabajo y, finalmente, en la sección 6 se recogen las conclusiones y propuestas de trabajo futuro.

[1] Véase `https://github.com/iml-wg/HEPML-LivingReview`.

2. Estado de la cuestión

Si bien ya se había intentado aplicarlas antes en otros ámbitos científicos, las GANs fueron empleadas por primera vez en el campo de la física de altas energías en [25]. El objetivo era acabar con la disyuntiva hasta entonces existente entre simulaciones rápidas pero no suficientemente precisas o precisas pero computacionalmente muy costosas. En otras palabras, el objetivo era reducir el coste en términos de precisión que suponía hasta entonces la aceleración de las simulaciones. En dicho trabajo, se aplicaban por primera vez las GANs a la simulación de la interacción de jets de partículas en calorímetros. Para ello, se hacía uso de redes convolucionales [28] y un modelo auxiliar [24] para condicionar el proceso de generación, un enfoque que serviría como punto de partida para muchos trabajos posteriores. Durante algún tiempo, los autores de este artículo lideraron los esfuerzos por aplicar las GANs al ámbito de los aceleradores de partículas, con trabajos como [27] y [26]. Como resultado, se mostró empíricamente el potencial de las GANs para la simulación eficiente de calorímetros condicionada mediante parámetros físicos, y se abrió el camino a numerosos trabajos académicos posteriores sobre esta cuestión[2]. Más tarde, en [11] se empleó por primera vez la función de pérdida de Wasserstein [1] con penalización de gradiente [15] para la simulación de detectores de partículas. A partir de este trabajo se emplearía frecuentemente en este ámbito esta función de pérdida que ya gozaba de gran popularidad fuera de él. Finalmente, cabe destacar en [7] el uso de un modelo pre-entrenado como modelo auxiliar para condicionar el proceso de generación de muestras.

Por otro lado, los únicos trabajos sobre la aplicación de GANs a la simulación de IACTs que se han podido encontrar son [9] y [10]. Ambos pertenecen a los mismos autores y en ellos se consigue generar satisfactoriamente sucesos de rayos gamma y protones mediante unas GANs no condicionales para cada tipo de partícula, a partir de un conjunto de imágenes de entrenamiento simuladas y sin ruido.

3. Marco teórico

3.1. Aprendizaje automático y profundo

El aprendizaje automático es una rama de la inteligencia artificial[3] cuyo objetivo es hacer que las computadoras aprendan, es decir, que mejoren su desempeño en una tarea dada, a partir de datos. Se suelen distinguir tres tipos de aprendizaje automático: supervisado o predictivo, que hace uso de datos etiquetados para entrenar modelos de regresión y clasificación; no supervisado o descriptivo, que entrena los modelos mediante datos sin etiquetar para descubrir patrones en ellos; y por refuerzo, donde el entrenamiento se basa en un sistema de recompensas por las acciones del modelo. A su vez, el aprendizaje profundo es un área del aprendizaje automático que se caracteriza por hacer uso de grandes modelos de aprendizaje, basados en redes neuronales artificiales, que se entrenan con masivas cantidades de datos.

Las redes neuronales, en su forma más simple, son modelos de aprendizaje supervisado que constan de L "capas", en cada una de las cuales se aplica una operación lineal, dada por una de matriz $W^{(l)}$ y un vector $\mathbf{b}^{(l)}$, que constituyen los parámetros del modelo, seguida de una función de activación no lineal $g^{(l)}$. Es decir, la salida $\mathbf{a}^{(l)}$ de cada capa l está dada por:

$$\mathbf{a}^{(l)} = g^{(l)}\left(W^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \qquad l = 1, \ldots, L$$

donde $\mathbf{a}^{(0)} \equiv \mathbf{x}$ denota el vector de entrada (datos). Su entrenamiento se lleva a cabo optimizando una función de coste J que mide la discrepancia entre las predicciones del modelo, $\hat{y}$, y los valores esperados,

[2] Un listado muy completo de ellos se puede encontrar también en `https://github.com/iml-wg/HEPML-LivingReview`.

[3] La inteligencia artificial es aquella disciplina que busca resolver problemas de forma autónoma simulando la inteligencia humana.

y, dados por las etiquetas de los datos de entrenamiento. La actualización de los parámetros del modelo para optimizar J se realiza mediante algoritmos basados en el denominado descenso del gradiente de dicha función de coste. Para el cálculo de dicho gradiente, se aplica la llamada propagación hacia atrás.

3.2. Modelado generativo

En el ámbito del aprendizaje automático, el modelado generativo es una tarea de aprendizaje no supervisado que consiste en tomar un conjunto de datos de entrenamiento, $\{\mathbf{x}^{(i)}\}_{i=1}^{m}$, formado por muestras tomadas de una distribución[4] de función de densidad p_{datos} desconocida, y aprender a representar una estimación, p_{modelo}, de dicha distribución[5]. En algunos casos, el modelo generativo estima p_{modelo} explícitamente; en otros, el modelo solo es capaz de hacerlo implícitamente, aprendiendo a generar muestras $\tilde{\mathbf{x}}$ de p_{modelo}. Algunos modelos incluso son capaces de realizar ambas tareas. [12]

A menudo, las diferentes componentes x_i de las variables visibles u observadas $\mathbf{x} = (x_1, \ldots, x_n)$ son altamente dependientes entre sí. En el contexto del aprendizaje profundo, el enfoque más comúnmente usado para modelar estas dependencias es introducir varias variables latentes u ocultas, $\mathbf{z}$. El modelo trata así de aprender las dependencias entre cualquier par de variables x_i y x_j indirectamente, por medio de las dependencias directas entre x_i y $\mathbf{z}$, y las dependencias directas entre x_j y $\mathbf{z}$. Una ventaja adicional de esta estrategia es que, más allá de su papel en la estimación eficiente de $p_{\text{datos}}(\mathbf{x})$, las variables latentes $\mathbf{z}$ también proporcionan una representación alternativa de $\mathbf{x}$ que puede resultar útil para ciertos propósitos. [13]

3.3. Redes generativas adversarias

Las redes generativas adversarias, comúnmente conocidas como GANs, fueron presentadas por primera vez en [14], como una arquitectura o estrategia para entrenar un modelo generativo. Lo que define esta técnica es que enfoca esta tarea de aprendizaje no supervisado como una de aprendizaje supervisado haciendo uso de dos submodelos adversarios:

- **El generador** G: es el modelo generativo que se quiere entrenar para generar nuevas muestras realistas. El generador toma un vector latente aleatorio $\mathbf{z}$, a modo de semilla, a partir del cual se genera una nueva muestra $\tilde{\mathbf{x}}$. Después[6] del entrenamiento, los puntos del espacio latente forman una representación comprimida o codificación de la distribución de datos del problema, y el generador actúa como decodificador.
- **El discriminador** D: es un modelo auxiliar que se introduce para dar un enfoque de aprendizaje supervisado al problema. En origen, se trata de un clasificador binario que intenta discernir entre muestras reales (del conjunto de datos de entrenamiento) y generadas. Lo bien o lo mal que hayan conseguido engañar las muestras del generador al discriminador en cada paso del entrenamiento de las GANs sirve de retroalimentación para entrenar el generador. Después del entrenamiento, este modelo normalmente se desecha, pues el interés se centra en el generador[7].

Los dos modelos son entrenados de forma conjunta en lo que, en teoría de juegos, se denomina un juego no cooperativo: D trata de detectar si una muestra es generada o real, mientras que G trata de maximizar la probabilidad de que D se equivoque. En el límite, y si ambos modelos tienen capacidad suficiente, el discriminador es incapaz de distinguir las muestras generadas de las reales en la mitad

[4] Normalmente, se hará referencia a las distribuciones de probabilidad a través de sus respectivas funciones de densidad p.

[5] En el caso de imágenes, por ejemplo, esto significa que se aprende a estimar la densidad de probabilidad $p_{\text{datos}}(\mathbf{x})$ de cada posible combinación $\mathbf{x}$ de los valores de los píxeles.

[6] El espacio latente no tiene significado hasta que el generador lo dota de él a medida que aprende [6].

[7] El discriminador también puede ser posteriormente empleado para aplicaciones de aprendizaje transferido [6].

de las ocasiones, lo que significa que el generador ha aprendido a crear muestras realistas, es decir, ha estimado correctamente la distribución de los datos p_{datos}.

Las GANs realizan una estimación implícita[8] de la distribución de los datos mediante la generación de muestras de p_{modelo}, por lo que no es necesario predefinir una familia paramétrica de funciones de densidad que ajustar a los datos de entrenamiento.

Normalmente, el generador y el discriminador son dos redes neuronales profundas. De hecho, debido a sus populares aplicaciones en la generación de imágenes, ambos modelos suelen estar basados en las llamadas redes neuronales convolucionales o CNNs (Convolutional Neural Networks). Este tipo de GANs, denominadas redes convolucionales profundas generativas adversarias o DCGANs (Deep Convolutional Generative Adversarial Networks), fueron presentadas en [28] y han demostrado ser más estables y potentes, hasta el punto de convertirse en el punto de partida para la mayoría de modelos.

3.3.1. Formalismo

Desde una perspectiva más formal, el generador, G, es una función diferenciable de parámetros $\boldsymbol{\theta}^{(G)}$ que toma como entrada un vector aleatorio $\mathbf{z}$ de una distribución $p_{\mathbf{z}}$ (gaussiana, normalmente) y que da como resultado un elemento $G(\mathbf{z};\boldsymbol{\theta}^{(G)})$ de una distribución p_G que intenta replicar la distribución p_{datos} de los datos reales $\mathbf{x}$. Clásicamente, el discriminador, D, es una función diferenciable escalar de parámetros $\boldsymbol{\theta}^{(D)}$ que da como resultado una estimación de la probabilidad $P_D(Y = 1 \mid \mathbf{x}) = D(\mathbf{x};\boldsymbol{\theta}^{(D)})$ de que $\mathbf{x}$ provenga de la distribución real de datos ($y = 1$), p_{datos}, en lugar de la generada ($y = 0$), p_G.

Las funciones de coste, $J^{(G)}$ y $J^{(D)}$, de los modelos G y D, respectivamente, están definidas en términos de los parámetros de ambos modelos: el discriminador busca minimizar $J^{(D)}(\boldsymbol{\theta}^{(G)},\boldsymbol{\theta}^{(D)})$ controlando únicamente $\boldsymbol{\theta}^{(D)}$ y el generador busca minimizar $J^{(G)}(\boldsymbol{\theta}^{(G)},\boldsymbol{\theta}^{(D)})$ controlando únicamente $\boldsymbol{\theta}^{(G)}$. Dado que la función de coste de cada jugador depende, no solo de sus propios parámetros, sino también de los del otro jugador, los cuales no puede controlar, resulta más natural describir este escenario como un juego (no cooperativo) que como un problema de optimización clásico. Mientras que la solución a un problema clásico de optimización es un mínimo (local) en el espacio de parámetros, la solución a un juego no cooperativo es lo que se denomina un equilibrio de Nash. En este contexto, un equilibrio de Nash es una tupla $(\boldsymbol{\theta}^{(G)},\boldsymbol{\theta}^{(D)})$ que constituye un mínimo local de $J^{(D)}$ respecto de $\boldsymbol{\theta}^{(D)}$ y un mínimo local de $J^{(G)}$ respecto de $\boldsymbol{\theta}^{(G)}$. [12]

Clásicamente, la función de coste para el discriminador se define como:

$$J^{(D)}(\boldsymbol{\theta}^{(G)},\boldsymbol{\theta}^{(D)}) = -\mathbb{E}_{\mathbf{x}\sim p_{\text{datos}}}[\log \underbrace{(D(\mathbf{x}))}_{P_D(Y=1|\mathbf{x})}] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log \underbrace{(1 - D(G(\mathbf{z})))}_{P_D(Y=0|\mathbf{x})}] \tag{1}$$

donde $\mathbb{E}_{\mathbf{x}\sim p}[f(\mathbf{x})]$ denota el valor esperado[9] de $f(\mathbf{x})$ cuando $\mathbf{x}$ es tomado de la distribución p. Esta es la función de coste empleada en [14] y no es más que la denominada entropía cruzada, que es comúnmente empleada en el entrenamiento de clasificadores binarios con una salida sigmoide[10].

La versión más sencilla del juego entre generador y discriminador es lo que en teoría de juegos se denomina un juego de suma cero, es decir, en el que la suma de las funciones de coste de ambos jugadores es cero, de modo que la ganancia de uno supone la pérdida del otro en la misma cuantía:

$$J^{(G)} = -J^{(D)} \tag{2}$$

[8] Es posible diseñarlas para que realicen también una estimación explícita de p_{datos} [12].

[9] En el caso discreto, $\mathbb{E}_{\mathbf{x}\sim p}[f(\mathbf{x})] = \sum_{\mathbf{x}} P(\mathbf{x}) f(\mathbf{x})$.

[10] La función logística sigmoide o simplemente sigmoide es $s(x) = 1/(1 + \exp(-x))$.

Esta relación directa entre ambas funciones de coste permite resumir el juego mediante una única función objetivo, definida como

$$V(\boldsymbol{\theta}^{(G)}, \boldsymbol{\theta}^{(D)}) = J^{(G)} = -J^{(D)} = \mathbb{E}_{\mathbf{x}\sim p_{\text{datos}}}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \tag{3}$$

de modo que los parámetros del generador óptimo son

$$\boldsymbol{\theta}^{(G)*} = \arg\min_{\boldsymbol{\theta}^{(G)}}\left[\max_{\boldsymbol{\theta}^{(D)}} V(\boldsymbol{\theta}^{(G)}, \boldsymbol{\theta}^{(D)})\right] \tag{4}$$

Es decir, para aprender a clasificar correctamente los ejemplos como reales o generados, D es entrenado para maximizar el logaritmo de $P_D(Y = 1 \mid \mathbf{x}) = D(\mathbf{x})$ para las muestras reales y el logaritmo de $P_D(Y = 0 \mid \mathbf{x}) = 1 - D(\mathbf{x})$ para las generadas ($\mathbf{x} = G(\mathbf{z})$), de modo que, idealmente, la probabilidad estimada para las muestras reales sea 1 y para las generadas, 0. Por el contrario, G trata de conseguir justo el objetivo opuesto, es decir, que la probabilidad predicha por el discriminador para las muestras generadas sea próxima a 1, haciendo "creer" al discriminador que las muestras generadas son reales. Debido a que su solución involucra un proceso de minimización en un bucle externo y uno de maximización en otro bucle interno, los juegos de suma cero también reciben el nombre de juegos minimax.

Haciendo uso de esta formulación de las GANs, en [14] se demuestra que:

1. El juego tiene un óptimo global en el que $G(\mathbf{z})$ sigue la misma distribución que los datos de entrenamiento, es decir $p_G = p_{\text{datos}}$, y $D(\mathbf{x}) = 1/2$ para todo $\mathbf{x}$.
2. El aprendizaje en este juego equivale a minimizar la llamada divergencia de Jensen-Shannon entre las distribuciones p_{datos} y p_G.
3. El siguiente algoritmo asegura la convergencia de p_G hacia p_{datos} si G y D pueden actualizarse directamente en el espacio de funciones[11]:

Algoritmo 1: Entrenamiento de GANs mediante descenso del gradiente de mini-lote. [14]

for n *iteraciones de entrenamiento* **do**
  for k *pasos* **do**
    Extraer un lote de m muestras, $\{\mathbf{z}^{(i)}\}_{i=1}^{m}$, de la distribución $p_{\mathbf{z}}$.
    Extraer un lote de m muestras, $\{\mathbf{x}^{(i)}\}_{i=1}^{m}$, de la distribución p_{datos}.
    Actualizar el discriminador mediante el descenso[a] del gradiente de mini-lote dado por

$$\nabla_{\boldsymbol{\theta}^{(D)}} J^{(D)} = -\nabla_{\boldsymbol{\theta}^{(D)}} \frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\mathbf{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\mathbf{z}^{(i)}\right)\right)\right)\right]$$

  end
  Extraer un lote de m muestras, $\{\mathbf{z}^{(i)}\}_{i=1}^{m}$, de la distribución $p_{\mathbf{z}}$.
  Actualizar el generador mediante el descenso del gradiente de mini-lote dado por[b]

$$\nabla_{\boldsymbol{\theta}^{(G)}} J^{(G)} = \nabla_{\boldsymbol{\theta}^{(G)}} \frac{1}{m}\sum_{i=1}^{m}\log\left(1 - D\left(G\left(\mathbf{z}^{(i)}\right)\right)\right)$$

end

[a] En [14] se propone el ascenso del mismo gradiente cambiado de signo, lo cual es totalmente equivalente.

[b] Nótese que $\nabla_{\boldsymbol{\theta}^{(G)}} \frac{1}{m}\sum_{i=1}^{m}\log D\left(\mathbf{x}^{(i)}\right) = 0$.

[11] En la práctica, G y D están representados por redes neuronales profundas, y las actualizaciones se realizan en el espacio de parámetros, de modo que este resultado no aplica, aunque no por ello resulta irrelevante.

Como último comentario de esta sección, cabe añadir que, si bien se suele decir que las GANs dan un enfoque de aprendizaje supervisado a la tarea de aprendizaje no supervisado que es el modelado generativo, también se puede pensar que el generador lleva a cabo una especie de aprendizaje por refuerzo. En efecto, en lugar de indicarle una salida $\mathbf{x}$ concreta que debe asociar con cada $\mathbf{z}$, se recompensa (o castiga) al generador por las acciones que realiza. Nótese que $J^{(G)}$ (su gradiente) no hace referencia a los datos de entrenamiento directamente en absoluto; toda la información sobre ellos proviene únicamente de lo que el discriminador ha aprendido, a través de sus predicciones sobre los datos generados. Esto tiene la ventaja de hacer que las GANs sean resistentes al sobreajuste, ya que el generador no tiene la oportunidad de copiar directamente ejemplos de entrenamiento. Debido a que resulta natural analizar las GANs con las herramientas de la teoría de juegos, en concreto no cooperativos, esta arquitectura recibe el calificativo de "adversarias", pero según esta interpretación, es posible entenderla desde un punto de vista cooperativo: el discriminador, más que como un adversario, actúa más bien como un instructor que enseña al generador cómo mejorar en su tarea de estimar la distribución de los datos. Sin embargo, como se comenta en [12], el proceso de entrenamiento difiere en ciertos aspectos del aprendizaje por refuerzo tradicional, por lo que esta imagen no es totalmente exacta.

3.3.2. Funciones de pérdida alternativas

La definición de $J^{(G)}$ que da lugar al anterior juego de suma cero resulta útil para el análisis teórico, pero no tiene un desempeño especialmente bueno en la práctica. Minimizar la entropía cruzada entre una distribución objetivo y la distribución estimada por un clasificador es muy efectivo para su entrenamiento, ya que la función de coste nunca se "satura" (aplana) cuando el clasificador tiene una salida incorrecta. Esta función de coste solo se satura finalmente, aproximándose asintóticamente a cero, cuando el clasificador ya ha aprendido la distribución correctamente y predice de forma correcta con un alto grado de confianza (probabilidades próximas a 0 o a 1, según la clase). En el juego minimax anterior, el discriminador minimiza la entropía cruzada, pero el generador maximiza esa misma entropía cruzada. Esto resulta problemático, ya que cuando el discriminador identifica correctamente las muestras del generador como falsas con un alto grado de confianza, el gradiente del generador se vuelve muy débil. Esta situación es, de hecho, bastante habitual, ya que la tarea del discriminador es mucho más fácil que la del generador, especialmente en etapas tempranas del entrenamiento, cuando los resultados de G son muy pobres, muy diferentes de los reales, y fáciles de identificar por D.

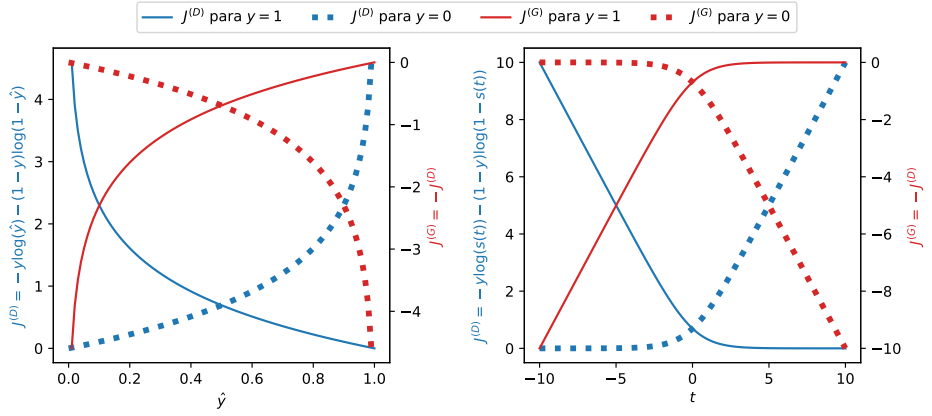


Figura 1: Funciones de pérdida en la versión de juego de suma de cero frente a la salida $\hat{y}$ de D (izda.) y frente a su salida t antes de aplicar la función de activación sigmoide s (dcha.).

Una posible solución a este problema es, en lugar de cambiar el signo de la función de coste del

discriminador para obtener la función de coste del generador ($J^{(G)} = -J^{(D)}$), intercambiar el objetivo[12] usado para construir la función de coste de entropía cruzada:

$$J^{(G)} = -\mathbb{E}_{\mathbf{x}\sim p_{\text{datos}}}[\log(1 - D(\mathbf{x}))] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(D(G(\mathbf{z})))] \tag{5}$$

En realidad, como en el descenso del gradiente de $J^{(G)}$ solo intervienen datos generados, se puede escribir

$$J^{(G)} = -\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(D(G(\mathbf{z})))] \tag{6}$$

De este modo, $J^{(G)}$ es también la entropía cruzada, pero no la misma[13] que la de $J^{(D)}$: $J^{(D)}$ es la entropía cruzada de la distribución $P_{\text{real}}(Y = 1 \mid \mathbf{x})$ (que asigna 1 a los datos reales y 0 a los generados) y la estimación que el discriminador hace de ella, $P_D(Y = 1 \mid \mathbf{x})$, mientras que $J^{(G)}$ es la entropía cruzada de $P_{\text{real}}(Y = 1 \mid \mathbf{x})$ y $1 - P_D(Y = 1 \mid \mathbf{x}) = P_D(Y = 0 \mid \mathbf{x})$ (distribución complementaria). Así, D busca aprender $P_{\text{real}}(Y = 1 \mid \mathbf{x})$, pero G le induce a aprender $P_{\text{real}}(Y = 0 \mid \mathbf{x})$. En el juego minimax, el generador minimiza $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]$, es decir, el grado de acierto del discriminador. En este juego, el generador maximiza $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(D(G(\mathbf{z})))]$, es decir, el grado de error del discriminador. El cambio es sutil y, de hecho, según [14] no parece tener implicaciones más allá de hacer que cada jugador tenga un gradiente suficiente cuando esté "perdiendo" el juego.

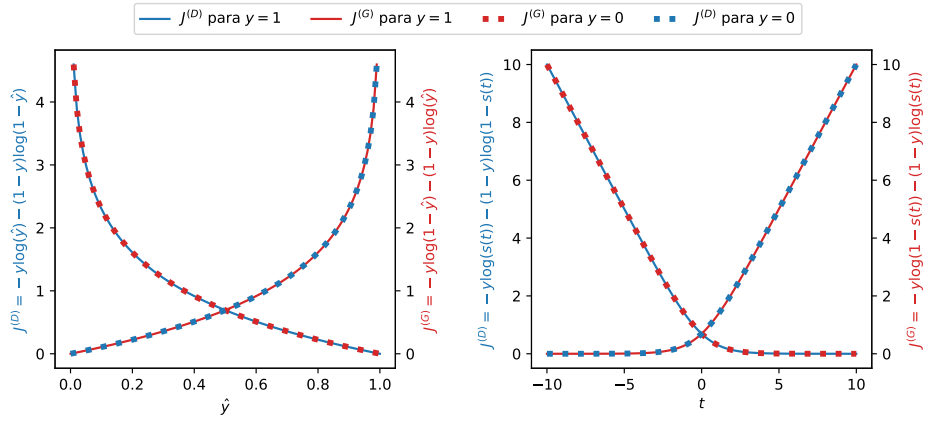


Figura 2: Funciones de pérdida en la versión modificada del juego frente a la salida $\hat{y}$ de D (izda.) y frente a su salida t antes de aplicar la función de activación sigmoide s (dcha.).

Sin embargo, esta nueva función de pérdida también presenta un inconveniente. Ahora, en el caso de que las imágenes del generador sean predichas como reales, apenas contribuyen a la función de coste y a sus gradientes, por lo que casi no se corrigen. Esto puede ser un problema cuando las imágenes del generador son predichas como reales por el discriminador, a pesar de ser muy distintas de ellas. Es decir, el discriminador apenas corrige imágenes que caen en el lado real de su frontera de decisión, incluso si se encuentran muy alejadas de la distribución real.

Ante los problemas que presenta la entropía cruzada como función de pérdida, muchos de los esfuerzos de la comunidad investigadora en el ámbito de las GANs se han centrado en buscar una función de pérdida alternativa, con el objetivo de proporcionar unos gradientes suficientes durante la mayor parte posible del entrenamiento, y así estabilizar el aprendizaje y asegurar la convergencia del mismo.

[12] En la práctica, esto se traduce en intercambiar las etiquetas real-generado para el entrenamiento de G.

[13] Esta versión del juego ya no es, por tanto, de suma cero, por lo que no puede ser descrita con una única función objetivo.

Least-Squares-GANs

Una de las alternativas más simples fue presentada en [20], y consiste en sustituir la salida sigmoide del discriminador por una lineal y hacer que tanto el generador como el discriminador minimicen el error cuadrático en lugar de la entropía cruzada:

$$J^{(D)} = \sum_i \left[\left(D\left(\mathbf{x}^{(i)}\right) - 1 \right)^2 + \left(D\left(G\left(\mathbf{z}^{(i)}\right)\right) - 0 \right)^2 \right] \tag{7a}$$

$$J^{(G)} = \sum_i \left[\left(D\left(\mathbf{x}^{(i)}\right) - 0 \right)^2 + \left(D\left(G\left(\mathbf{z}^{(i)}\right)\right) - 1 \right)^2 \right] \rightarrow \sum_i \left(D\left(G\left(\mathbf{z}^{(i)}\right)\right) - 1 \right)^2 \tag{7b}$$

La ventaja de esta función de pérdida es que su gradiente solo se anula cuando las etiquetas coinciden exactamente con las predicciones y aplica una mayor penalización (el gradiente es mayor) cuanto mayor es el error, tanto si es por exceso como por defecto. La función de pérdida de mínimos cuadrados es capaz, por consiguiente, de mover los ejemplos generados hacia el límite de decisión[14], ya que penaliza tanto más cuanto más alejados se encuentren, incluso si se hallan en el lado correcto de la frontera de decisión. Todo esto da lugar a un entrenamiento más estable y con resultados de mayor calidad.

Otra consecuencia reseñable de este cambio de función de pérdida es que, en lugar de minimizar la divergencia de Jensen-Shannon entre p_{datos} y p_G, las LSGANs (Least-Squares-GANs) minimizan la divergencia χ^2 de Pearson.

Wasserstein-GANs

Una propuesta más sofisticada y popular es la denominada función de pérdida de Wasserstein, presentada en [1]. En este artículo, se centra la atención en las distintas formas de medir cómo de próximas son la distribución real de los datos y la estimación que el generador hace de ella, o equivalentemente, en las distintas formas de definir una distancia o divergencia, $\rho(p_G, p_{\text{datos}})$, entre ambas. Este punto resulta clave para los autores, ya que cada definición de distancia o divergencia tiene un impacto distinto en la convergencia de secuencias de distribuciones de probabilidad[15]. Es decir, la noción de convergencia de las distribuciones p_{G_t} depende de la forma en que se calcule la distancia entre dos distribuciones.

Como resultado de un análisis teórico, los autores concluyen que la denominada distancia de excavadora o distancia EM (Earth-Mover distance), también conocida como distancia Wasserstein-1, presenta ventajas interesantes en comparación con la divergencia de Jensen-Shannon y otras distancias y divergencias popularmente empleadas en el contexto del aprendizaje de distribuciones. En consecuencia, se propone una nueva forma de GANs que tiene por objetivo minimizar la distancia EM. Para ello, se define una nueva función de pérdida que aproxime dicha distancia[16]:

$$J^{(D)} = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))] - \mathbb{E}_{\mathbf{x}\sim p_{\text{datos}}}[D(\mathbf{x})] \tag{8a}$$

$$J^{(G)} = -J^{(D)} = \mathbb{E}_{\mathbf{x}\sim p_{\text{datos}}}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))] \rightarrow -\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))] \tag{8b}$$

Por este motivo, este tipo de GANs se denominan Wasserstein-GANs o WGANs. El uso de esta función de pérdida implica sustituir la salida sigmoide del discriminador por una lineal. Ahora la salida del discriminador es un número real no acotado, en lugar de una estimación de probabilidad (entre 0 y 1).

[14] Recuérdese que, en el equilibrio óptimo, el discriminador predice correctamente solo en la mitad de los casos, por lo que su frontera de decisión divide por la mitad la distribución de datos reales, de ahí que el objetivo sea acercar los datos generados a la frontera de decisión.

[15] Una secuencia de distribuciones $\{p_t\}_{t\in\mathbb{N}}$ converge si y solo si hay una distribución p_∞ tal que $\rho(p_t, p_\infty)$ tiende a cero, algo que depende de cómo se defina exactamente ρ [1].

[16] También es posible definir las funciones de pérdida añadiendo un signo negativo global en ambos casos.

Este número real se interpreta como una calificación del grado de realismo de una imagen. Por ello, en este caso el discriminador suele recibir el nombre de crítico. El objetivo del crítico es maximizar la distancia entre las calificaciones de los datos reales y los generados.

Además del análisis teórico, en este trabajo, también se muestra empíricamente algunas de las ventajas que presentan las WGANs. En primer lugar, al proporcionar unos gradientes útiles en casi todos los contextos, la función de pérdida de Wasserstein permite un entrenamiento continuado y estable de los modelos. Por otro lado, al estabilizar el proceso de entrenamiento, las WGANs no requieren un constante equilibrio entre las capacidades del discriminador y el generador, ni tampoco un cuidado diseño de la arquitectura de los modelos y su configuración de hiperparámetros. De hecho, los autores recomiendan entrenar al crítico hasta el óptimo: cuanto más se entrena el crítico, mejor es la estimación de la distancia EM y más confiable es el gradiente que se obtiene de la función de pérdida de Wasserstein. Otra importante virtud de las WGANs es que reducen drásticamente el frecuente problema del colapso modal, es decir, la situación en la que el generador produce siempre la misma imagen o un conjunto muy poco variado de ellas. Es más, en el mismo artículo se afirma que entrenar el crítico hasta el óptimo hace imposible el colapso modal. Por último, al estimar de forma continua la distancia EM mediante el entrenamiento del discriminador hasta el óptimo en cada paso de entrenamiento, el valor de la función de pérdida parece estar significativamente bien correlacionado con la calidad observada de las muestras generadas: un valor de la función de coste del discriminador más próximo a cero indicaría una mejor calidad de imágenes generadas. Hasta donde alcanza el conocimiento de los autores, esta es la primera vez en la literatura sobre GANs que una función de pérdida presenta dicha propiedad. Ahora, en lugar de buscar un equilibrio entre dos modelos, se busca explícitamente minimizar una función de pérdida. Sin embargo, los autores también advierten de que, por sí sola, la función de Wasserstein no es un método cuantitativo de evaluación de GANs. En concreto, es incorrecto comparar así GANs con distintos modelos de crítico.

Una particularidad de las WGANs de especial relevancia es que requieren que el discriminador (crítico) pertenezca al denominado espacio de funciones 1-Lipschitz. Se dice que una función $f : \mathbb{R}^n \to \mathbb{R}^m$ es k-Lipschitz si $\|f(\mathbf{x}) - f(\mathbf{x}')\|/\|\mathbf{x} - \mathbf{x}'\| \leq k,\ \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, donde (en el contexto de este trabajo) la norma es la distancia euclídea. En un principio, en [1] se impone esta condición mediante el recorte de los pesos del crítico, forzándolos a estar contenidos en un intervalo compacto $[-c, c]$. El conjunto de funciones que satisfacen esta restricción es un subconjunto de las funciones k-Lipschitz para algún k que depende de c y de la arquitectura del crítico. Sin embargo, esta estrategia ya era criticada desde un inicio por los propios autores: un intervalo pequeño restringe demasiado el espacio de parámetros en el que llevar a cabo la optimización, mientras que uno grande puede no garantizar la condición deseada.

Una alternativa mucho más conveniente y popular es la denominada penalización del gradiente, presentada en [15]. La base de esta propuesta es que una función diferenciable es 1-Lipschitz si y solo si tiene gradiente de norma 1 en todas partes. Por este motivo, se intenta restringir la norma del gradiente del crítico con respecto a su entrada añadiendo un término de regularización a la función de pérdida que penalice que dicha norma sea distinta de 1. En la práctica, esta penalización se calcula sobre muestras de la forma $\hat{\mathbf{x}} := \epsilon\tilde{\mathbf{x}} + (1 - \epsilon)\mathbf{x}$, creadas mediante la interpolación de muestras generadas, $\tilde{\mathbf{x}}$, y reales, $\mathbf{x}$, donde ϵ es un número aleatorio de una distribución uniforme $U(0, 1)$:

$$J^{(D)} = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))] - \mathbb{E}_{\mathbf{x}\sim p_{\text{datos}}}[D(\mathbf{x})] + \lambda\mathbb{E}_{\hat{\mathbf{x}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2] \tag{9a}$$

$$J^{(G)} = -J^{(D)} = \mathbb{E}_{\mathbf{x}\sim p_{\text{datos}}}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))] \to -\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))] \tag{9b}$$

Por este motivo, se suele hacer referencia a este tipo de GANs como WGAN-GP (Wasserstein-GANs with Gradient Penalty). Es posible también, como se muestra en [18], obtener otra función de pérdida si se

combinan la entropía cruzada con la penalización del gradiente.

Otra alternativa al recorte de pesos que también es digna de mención es la denominada normalización espectral. Fue presentada en [23] y se trata de una técnica de normalización de pesos que se aplica sobre el crítico[17]. La idea es que la condición de que el crítico sea una función 1-Lipschitz es equivalente a que la norma espectral de las matrices de pesos que configuran sus capas sea igual a 1, por lo que se propone normalizarlas. Además se presenta una implementación computacionalmente eficiente y fácil de incorporar a modelos ya existentes. Una posible ventaja de este enfoque frente a la penalización del gradiente es su menor coste computacional. Los autores plantean incluso la posibilidad de emplearse en combinación con la penalización del gradiente.

Elección de la función de pérdida

Es importante mencionar que existe también evidencia empírica, como la recogida en [19], que apunta a que la elección de la función de pérdida no es en realidad tan relevante en todos los casos. Entre las conclusiones de dicho trabajo, se incluye que las diferencias entre las distintas funciones de pérdida se vuelven menos relevantes a medida que el presupuesto computacional aumenta y que, dado un presupuesto limitado, un algoritmo teóricamente superior puede ofrecer peores resultados que otro considerado inferior. Sin embargo, sus autores advierten de que no se puede excluir la posibilidad de que algunos modelos superen significativamente a otros bajo condiciones que no han sido exploradas en su estudio. De hecho, la experiencia en la realización de este proyecto apunta muy claramente a que la elección de la función de pérdida sí es relevante, en la línea de la opinión más extendida en la literatura sobre GANs.

Como último apunte, solo decir que todas las propuestas de GANs incluidas en esta sección se implementan siguiendo también Algoritmo 1 sin más que modificar las funciones de coste $J^{(D)}$ y $J^{(G)}$.

3.3.3. GANs condicionales

Un potencial inconveniente de las GANs vistas hasta ahora es que, si bien son capaces de generar nuevas muestras realistas, no se tiene ningún control sobre los datos concretos que se generan. A lo sumo, se puede intentar explorar el espacio latente una vez finalizado el entrenamiento para encontrar algunas relaciones entre sus puntos y los datos generados. Sin embargo, en ausencia de ningún condicionamiento sobre la forma en que el generador debe aprender a transformar los puntos latentes $\mathbf{z}$, es posible que la estructura que el generador acabe dando al espacio latente sea muy intrincada, sin que ninguna dimensión individual corresponda a aspectos de los datos con un significado evidente.

No obstante, existen varias formas de extender las GANs a un formato condicional. Esta idea ya fue introducida desde un comienzo en [14], aunque su primera implementación, comúnmente conocida como CGANs (Conditional GANs), fue presentada en [22]. La estrategia que se expone en estos trabajos consiste en suministrar como datos de entrada alguna información adicional tanto al generador como al discriminador para condicionar sus salidas. Esta información adicional puede ser etiquetas de clase, pero también puede ser, por ejemplo, imágenes[18]. El problema de este enfoque es que esa información extra no se tiene en cuenta en absoluto a la hora de calcular las funciones de pérdida, es decir, no se valida de ninguna forma que el generador esté efectivamente haciendo un uso correcto de ella para condicionar su salida.

[17] Algunos trabajos posteriores como [30] y [5] recomiendan aplicarla también sobre el generador.

[18] Un ejemplo de esto es la denominada traducción imagen a imagen, de la que algunas aplicaciones son la transferencia de estilo o el coloreado de imágenes en blanco y negro.

Sin embargo, el enfoque de [22] no es el único. Por ejemplo, es también posible condicionar la generación de datos mediante una tarea de aprendizaje no supervisado, como se propone en [8]. Esta clase de GANs, denominadas InfoGANs (Information Maximizing GANs), introduce variables de control en el mismo vector latente, que más tarde son validadas por un modelo auxiliar, lo que permite que el generador dote de significado a dichas variables de control (a sus correspondientes dimensiones en el espacio latente) y obtener así una representación con la que resulte fácil operar, y todo ello sin necesidad de datos etiquetados, en un proceso de aprendizaje no supervisado. El modelo auxiliar, que se puede entender bien como un modelo independiente o bien integrado en el discriminador, predice si los datos generados se corresponden con los valores dados en las variables de control de los vectores latentes con los que fueron generados. De esta forma, el modelo auxiliar busca darle algún significado a las variables de control y, mediante una contribución adicional en la función de pérdida del generador, empuja a este a dotar también de significado a dichas variables de control durante el proceso de generación de muestras.

Dentro del condicionamiento supervisado, es especialmente relevante, tanto por su versatilidad como por sus buenos resultados, la propuesta de [24], denominada AC-GANs (Auxiliary Classifier GANs). En cierto modo, el enfoque de este trabajo se puede ver como una fusión de las ideas de [22] y [8]: se proporcionan, junto con el vector latente, las etiquetas de clase como entrada al generador, pero no al discriminador, y se hace uso de un modelo auxiliar que valide que las imágenes generadas se corresponden con dichas etiquetas. Una vez más, el modelo auxiliar se puede ver como una parte integrada en el discriminador o como un modelo independiente. La ventaja ahora es que dicho modelo auxiliar puede estar pre-entrenado o no. Nótese además que, en el caso de ser un modelo pre-entrenado, no se necesitan datos de entrenamiento etiquetados para aplicar este enfoque, pues ya no se necesitan las etiquetas de los datos reales para entrenar el modelo auxiliar, solamente las etiquetas empleadas para crear los datos generados, que bien pueden ser inventadas.

El algoritmo de entrenamiento de las AC-GANs es Algoritmo 1 sin más que sumar una contribución a $J^{(G)}$ procedente de las predicciones del modelo auxiliar sobre los datos generados. De este modo, el generador busca crear muestras realistas que engañen al discriminador y que, al mismo tiempo, se correspondan con las etiquetas adecuadas según el modelo auxiliar. En el caso de que el modelo auxiliar no esté pre-entrenado, se entrena en el mismo paso que el discriminador. La función de pérdida del discriminador seguirá siendo la misma si está separado del modelo auxiliar y, en caso contrario, también recibirá una contribución adicional en forma de un término aditivo por la tarea de reconstruir las etiquetas de los datos. Las AC-GANs tienen la ventaja adicional de permitir el uso de cualquier función de activación en la salida de los modelos y cualquier función de pérdida de forma trivial.

Para terminar la exposición sobre el condicionamiento de las GANs, es importante decir que, además de permitir cierto control sobre el proceso de generación de muestras, esta extensión de las GANs originales presenta ventajas adicionales. Al hacer uso de información adicional, proporcionada normalmente por las etiquetas de los datos de entrenamiento, y forzar a los modelos a la multitarea, se consigue mejorar los resultados de las GANs, ya sea en forma de una reducción del colapso modal, un entrenamiento más estable, un aprendizaje más rápido o imágenes de mayor calidad. Además, las GANs condicionales permiten la generación de nuevos datos etiquetados, lo que tiene aplicaciones, por ejemplo, en el aumento de conjuntos de datos de entrenamiento para otros modelos.

3.3.4. Evaluación de GANs

Hasta ahora, se ha hablado sobre las funciones de pérdida empleadas en el entrenamiento de las GANs, pero nada acerca de las posibles métricas para evaluar sus resultados. En el caso de las GANs, el valor de la función de pérdida no está necesariamente correlacionado con unos mejores resultados, y

cuando parece estarlo, como en el caso de las WGANs, la función de pérdida no permite comparar el desempeño de distintos modelos. El problema es que las GANs carecen de una función objetivo evidente que optimizar, pues resulta complicado definir de forma cuantitativa una función que evalúe la calidad de los resultados, su realismo, su variedad, etc. Si bien existen muchas propuestas de métricas, tanto cualitativas como cuantitativas, para la evaluación de GANs, no existe un consenso sobre cuál es la medida que mejor captura las fortalezas y limitaciones de los modelos y que, por tanto, debe ser empleada para un comparación justa y objetiva de los mismos en una amplia variedad de contextos. Es por esto que el problema de la evaluación objetiva y cuantitativa de GANs permanece abierto, y a menudo se definen métricas específicas para el dominio particular de cada problema.

Una práctica habitual es tomar como punto de partida la evaluación visual de los resultados. Esta estrategia suele resultar bastante útil en las primeras etapas de la experimentación, sobre todo para descartar fácilmente aquellas arquitecturas o configuraciones con un desempeño deficiente. Sin embargo, se trata de un método muy limitado: es difícil comparar resultados por encima de cierto umbral de calidad, depende de sesgos del examinador y de sus conocimientos sobre el dominio del problema, está limitado por el número de imágenes que pueden ser revisadas en un tiempo razonable, etc. Por todo ello, la evaluación visual no debería emplearse como un método de selección final del modelo, especialmente en proyectos no triviales.

Por otro lado, dos propuestas de métrica cuantitativa para GANs bastante populares son las denominadas “inception score” y “Fréchet inception distance”. Ambas se basan en las predicciones que un clasificador hace sobre los datos generados. En concreto, se emplea el modelo denominado Inception v3, del cual ambas métricas reciben su nombre, y que es entrenado con el conjunto de datos llamado ImageNet. Si bien la estrategia de estas propuestas puede ser extrapolable a distintos ámbitos, su dependencia del modelo Inception v3 y el conjunto de entrenamiento ImageNet hace que no sean directamente aplicables a muchos problemas.

4. Metodología

Tras una fase previa de formación teórica y práctica en aprendizaje profundo, así como un proceso de indagación sobre la posible aplicación de modelos generativos al ámbito de la física de partículas, el primer paso en la realización del proyecto consistió en implementar la arquitectura propuesta, AC-GANs, a pequeña escala y entrenarla con un conjunto de datos pequeño, sencillo y fácil de evaluar visualmente, como lo es MNIST. El objetivo era hacer una primera validación rápida de la idea que se quería aplicar al problema real. Los resultados fueron altamente satisfactorios bajo evaluación visual y pueden consultarse en el siguiente repositorio en GitHub: `https://github.com/sgh14/MNIST_GANs`.

El siguiente paso consistió en extrapolar esta implementación al problema de los telescopios Cherenkov (IACTs), adaptando el código al nuevo conjunto de datos e integrando los modelos de CTLearn como modelos auxiliares. El modelo se ha programado de tal forma que se pueda definir fácilmente una configuración de su arquitectura e hiperparámetros, entre otros aspectos, mediante un archivo yaml externo, lo que permite probar diferentes configuraciones y su fácil integración con otros conjuntos de datos y modelos predefinidos.

La implementación de los modelos se ha realizado haciendo uso de Python 3 como lenguaje de programación y Tensorflow 2 como librería especializada en el aprendizaje profundo. Además de por ser herramientas de código abierto, con una sintaxis sencilla y potente, y con una amplia documentación, la principal motivación de esta elección era poder integrar este trabajo con CTLearn, que ya hacía uso de

ellas. Para la realización del proyecto, y con objeto de facilitar el seguimiento del mismo, se ha creado el siguiente repositorio en GitHub: `https://github.com/sgh14/CTGANs`.

Una vez programado el modelo en sí, el siguiente paso consistió en buscar una configuración satisfactoria del mismo. Para ello, se escogió un conjunto de datos de entrenamiento simulados de los telescopios MAGIC[19]. Existen varios motivos para esta elección. En primer lugar, los telescopios MAGIC llevan funcionando bastante tiempo, y por ello su funcionamiento está bien caracterizado y se dispone de datos reales de ellos. Por otro lado, cuentan también con simuladores de sus detectores y ya se han aplicado los modelos de CTLearn previamente con sus datos. Merece la pena resaltar la importancia de contar tanto con datos reales como simulados. Por un lado, los datos simulados son los únicos datos etiquetados disponibles, por lo que son necesarios para entrenar los modelos auxiliares (de CTLearn) y para una primera evaluación de los resultados. Por otra parte, los datos reales son necesarios para, en un futuro, entrenar con ellos las GANs y evaluar su capacidad de generar datos realistas frente a las simulaciones de Monte Carlo. Para reducir la complejidad del problema, se optó por usar un subconjunto de los datos de entrenamiento con: un único telescopio, un solo canal, un ángulo cenital entre $5°$ y $35°$ y un umbral de energía alto.

Para llevar a cabo la búsqueda de configuraciones, se definieron varios archivos yaml de configuración del modelo con distintas arquitecturas, funciones de pérdida y conjuntos de hiperparámetros. Sin embargo, debido a la ingente cantidad de posibles combinaciones de parámetros, la falta de una métrica que optimizar, lo costoso que resulta el entrenamiento de los modelos, y el reducido tiempo disponible para la realización del proyecto, un filtrado sistemático de todas las configuraciones relevantes resultaba inviable. En cambio, se ha llevado a cabo una búsqueda (semi)aleatoria, una estrategia muy empleada en este campo del aprendizaje profundo. En un proceso de prueba y error, extrapolando inicialmente la configuración empleada en el proyecto piloto, y guiado por los consejos y experiencia recogidos en los distintos trabajos consultados[20], se fue afinando la búsqueda de configuraciones hasta dar con la que se presenta en la sección de resultados. En un primer momento, se entrenó el modelo durante un número reducido de épocas en una batería de configuraciones para realizar una primera criba de ellas. Posteriormente, se fue incrementando el número de épocas de entrenamiento a medida que se afinaba la búsqueda. El método de selección consistía fundamentalmente en la evaluación periódica[21] visual, que en la inmensa mayoría de los casos era suficiente para descartar las peores configuraciones.

Finalmente, tras haber seleccionado una configuración del modelo, es necesaria una evaluación del mismo. El método de evaluación escogido consiste en lo siguiente. Primero se escoge un único conjunto de datos simulados (etiquetados) y se divide, aproximadamente, en un $80\,\%$ para entrenamiento y un $20\,\%$ para validación. Posteriormente, se usa el conjunto de entrenamiento para entrenar:

- Unas GANs (y sus respectivos modelos auxiliares de CTLearn) para cascadas inducidas por rayos gamma y otras para las inducidas por protones, en ambos casos condicionadas en energía y dirección de llegada de la partícula que genera la cascada.
- Un clasificador binario de CTLearn que distinga entre rayos gamma y protones.
- Un modelo de CTLearn de regresión de energía y otro de dirección de llegada por cada tipo de partícula (rayos gamma y protones).

[19] `https://magic.mpp.mpg.de/`

[20] La mayoría de ellos están recogidos en [6], y proceden de artículos como [28], [29] o [5].

[21] Dado que no hay una medida objetiva del desempeño del modelo, no se puede saber cuándo se debe detener el entrenamiento. Por consiguiente, es una práctica común el usar el generador para crear un cierto número de muestras periódicamente durante el entrenamiento, al mismo tiempo que se salva el estado del generador en ese momento. Esto permite una evaluación posterior del modelo y escoger la versión que produzca los resultados más satisfactorios. [6]

Por último, se usan las GANs entrenadas para generar datos correspondientes a las mismas etiquetas que el conjunto de datos de validación, y se evalúa la calidad de los resultados atendiendo al grado de realismo de las imágenes y su correlación con sus etiquetas. Para valorar el nivel de realismo, se estudian las predicciones del discriminador sobre los datos de validación, tanto simulados como generados. Por otro lado, para evaluar el grado de correlación de las imágenes con sus etiquetas, se comparan las predicciones del clasificador y los dos modelos de regresión sobre los datos de validación con las que hacen sobre sus análogos generados. Esto constituye un primer método para comprobar la capacidad de las GANs de realizar el mismo papel que las simulaciones. Para comprobar si además suponen una ganancia en términos de coste computacional, se mide el tiempo empleado en la generación de los datos de validación a partir de sus etiquetas. Como último apunte, nótese que el método de validación no debe emplearse también como método de búsqueda y selección de configuraciones de los modelos (a menos que se usen distintos conjuntos de datos en cada caso) con el fin de no sobreajustar la búsqueda al conjunto de datos de validación.

5. Resultados

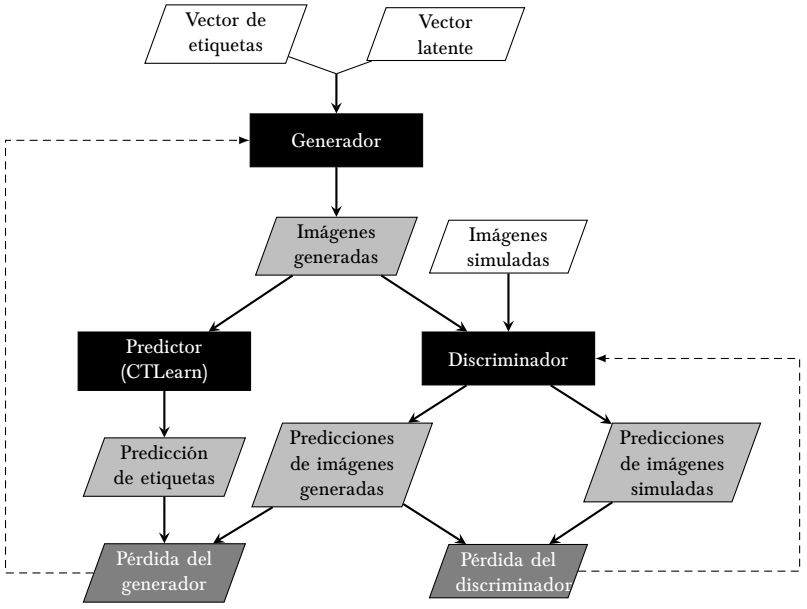


Figura 3: Arquitectura de CTGANs

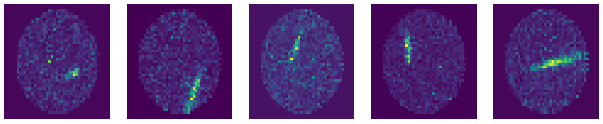
El primer resultado de este proyecto es la creación del repositorio CTGANs[22]. Este consta básicamente de un conjunto de scripts de Python que implementan las ideas expuestas en la sección 3 (arquitecturas, funciones de pérdida, algoritmo de entrenamiento, etc.), entre otras funcionalidades, y las integra con la librería CTLearn para permitir el condicionamiento de las GANs. El código está pensado de tal forma que los modelos, sus hiperparámetros y su conjunto de datos de entrenamiento se puedan especificar externamente mediante un archivo yaml. El objetivo de esto es facilitar la experimentación, el ajuste de hiperparámetros y la integración con modelos predefinidos. El enfoque adoptado es fundamentalmente el de AC-GANs, por sus buenos resultados y su flexibilidad a la hora de integrar distintos modelos y funciones de pérdida para condicionar las GANs. El esquema del modelo es el que se muestra en Figura 3. En primer lugar, el generador crea muestras a partir de la concatenación de un vector latente y un vector de etiquetas (una selección de entre: tipo de partícula, energía y dirección

[22] `https://github.com/sgh14/CTGANs`

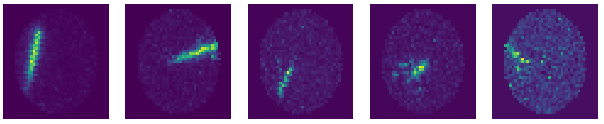
de llegada). Posteriormente, las imágenes generadas son pasadas tanto al discriminador como al modelo auxiliar, que en el repositorio recibe el nombre de predictor. El discriminador predice el grado de realismo tanto de las imágenes generadas como de las simuladas, mientras que el predictor, que es un modelo pre-entrenado[23] de CTLearn, predice el valor de las etiquetas con las que fueron generadas las salidas del generador. La salida del predictor junto con las del discriminador sobre las imágenes generadas son empleadas para calcular la función de pérdida del generador, mientras que la función de pérdida del discriminador se construye únicamente a partir de sus salidas sobre las imágenes simuladas y las generadas.

La configuración particular del modelo que finalmente se ha encontrado más satisfactoria consiste en un generador y un discriminador en forma de CNNs, y un modelo auxiliar (predictor) dado por lo que se denomina una "residual network" (ResNet), de CTLearn [21]. Debido a que un mismo modelo de CTLearn no puede, por el momento, predecir energía y dirección de llegada para varios tipos de partículas distintos, se han definido unas GANs independientes para rayos gamma y otras para protones. Las funciones de pérdida empleadas son la de Wasserstein con penalización de gradiente para las salidas del discriminador, y el error cuadrático medio para las salidas (energía y dirección de llegada) del predictor. El entrenamiento ha consistido en 200 épocas para el conjunto de datos de rayos gamma, que superaba las 700.000 imágenes, con un tamaño de mini-lote de 512 imágenes y Adam como algoritmo de optimización. La misma configuración se ha empleado para protones, aunque en este caso el número de imágenes no llegaba a las 80.000, por lo que el número de épocas de entrenamiento se incrementó hasta 2000. En ambos casos, las imágenes tenían dimensiones de $43 \times 43 \times 1$. Aunque el código es capaz de trabajar con varios canales en las imágenes, solo se ha usado uno para simplificar el problema. El entrenamiento de las GANs se ha llevado a cabo en una GPU Nvidia RTX 3090 durante aproximadamente 130 horas para rayos gamma y 120 para protones. Todos los detalles sobre los modelos y su entrenamiento se pueden consultar en los archivos yaml de configuración incluidos en el repositorio del proyecto.

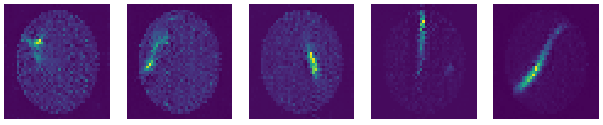
En la siguiente figura se muestran algunos ejemplos de imágenes simuladas y las imágenes generadas correspondientes a las mismas etiquetas:



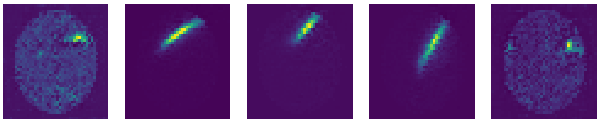
(a) Imágenes simuladas de rayos gamma.



(b) Imágenes simuladas de protones.



(c) Imágenes generadas de rayos gamma.



(d) Imágenes generadas de protones.

Figura 4: Imágenes simuladas y generadas para las mismas etiquetas, de rayos gamma y protones.

A simple vista, aunque mejorable, el grado de realismo parece aceptable. Esta idea es apoyada por la estimación que el discriminador hace de la distancia de Wasserstein-1 entre las imágenes simuladas y las generadas para los datos de validación:

[23] En caso de no existir previamente, el modelo se entrena automáticamente antes de comenzar el aprendizaje de las GANs.

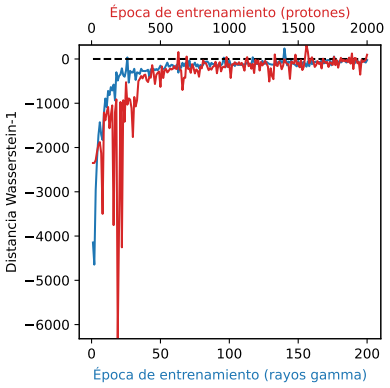
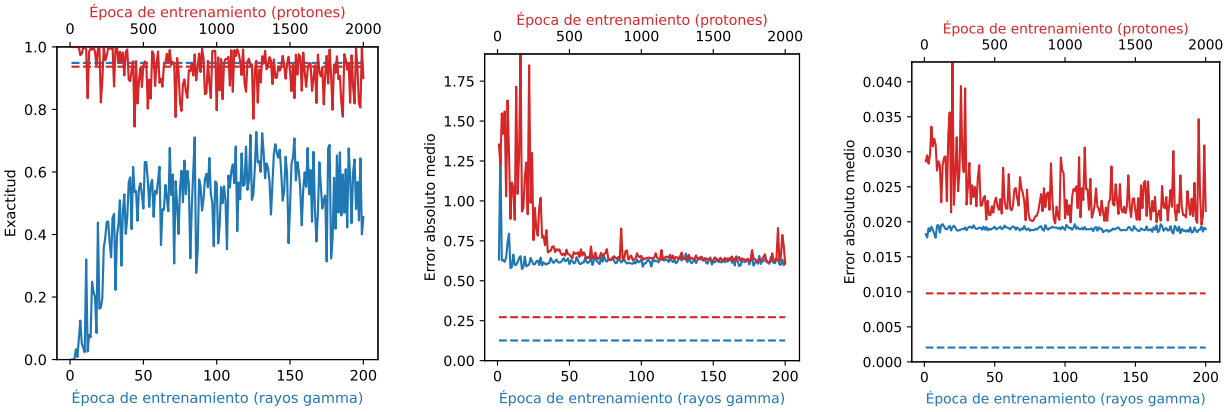


Figura 5: Estimación del discriminador de la distancia de Wasserstein-1 entre las imágenes simuladas y las generadas para los datos de validación.

Además del nivel de realismo, es necesario evaluar el grado de correlación de las imágenes generadas con sus respectivas etiquetas. Para ello, se ha recurrido a un total de tres métricas evaluadas sobre las predicciones que modelos de CTLearn hacen de los datos de validación. En concreto, se estudia la fracción de imágenes que son correctamente identificadas con el tipo de partícula que especifica su etiqueta (exactitud) y el error absoluto medio de las predicciones de los modelos de CTLearn sobre la energía y dirección de llegada de las imágenes respecto de sus etiquetas:



(a) Clasificación de partícula.

(b) Regresión de energía.

(c) Regresión de dirección de llegada.

Figura 6: Métricas de evaluación de la correlación de las imágenes (línea continua para generadas y discontinua para simuladas) con sus etiquetas.

Como se puede ver, la correlación es bastante mejorable. La única excepción es la exactitud para las imágenes de protones, pero esto podría deberse a que el modelo de CTLearn tiende a clasificar como protones las imágenes de peor calidad, algo que estaría incrementando la exactitud para protones y disminuyéndola para rayos gamma. El problema de la baja correlación de las imágenes con sus etiquetas puede deberse a la insuficiente búsqueda de configuraciones y ajuste de hiperparámetros. Quizás se pueda solucionar o mitigar este problema incrementando el peso de las contribuciones del modelo auxiliar en la función de pérdida del generador.

Por último, el tiempo de generación por imagen, haciendo uso de una GPU Nvidia RTX 3090, tanto en el caso de rayos gamma como de protones, es del orden de los microsegundos. En cambio, el tiempo

medio de simulación de un suceso es de aproximadamente 1 segundo, dependiendo de los telescopios que se simulen, del tipo de partícula y su energía, o de la potencia de cómputo disponible[24]. La diferencia entre los tiempos de generación en ambos casos es de entre 5 y 6 órdenes de magnitud. Pese a que la comparativa no es totalmente justa, debido a que cada suceso simulado consta de una imagen por cada detector, mientras que, en este proyecto, las GANs generan una imagen para un solo detector por cada suceso, la diferencia de tiempo es tan grande que se puede afirmar con seguridad que las GANs suponen una importante ganancia en términos de coste computacional en cualquier caso.

6. Conclusiones y trabajo futuro

Los resultados de este trabajo son insuficientes para afirmar que las GANs pueden sustituir las actuales simulaciones de Monte Carlo para la creación de imágenes de telescopios Cherenkov de imagen atmosférica. El principal motivo es la falta de correlación entre las imágenes generadas y sus etiquetas. Sin embargo, el grado de realismo de las imágenes generadas es al mismo tiempo prometedor, más aun si se tiene en cuenta el poco tiempo del que se ha dispuesto para experimentar con las configuraciones del modelo y llevar a cabo un buen ajuste de hiperparámetros. Es importante también decir que este trabajo presentaba una dificultad añadida frente a la inmensa mayoría de los trabajos citados en la sección 2, pues las dimensiones de las imágenes en este caso eran significativamente mayores, lo que incrementa el coste computacional y la dificultad del problema. Por todo ello, los resultados de este proyecto se deben entender más como una prueba de concepto abierta a una mayor experimentación posterior que como una demostración de eficacia.

Existen varios incentivos importantes para seguir apostando por las GANs como un potencial sustituto de las simulaciones de Monte Carlo: disminuyen drásticamente los costes computacionales y podrían proporcionar imágenes etiquetadas más realistas y mediante un proceso de generación más selectivo. Merece la pena hacer hincapié en este último punto, pues las GANs cambian totalmente el enfoque en la generación de datos: ahora es posible generar a demanda imágenes de un tipo de partícula concreto con unos parámetros físicos dados, en lugar tener que generar toda una distribución de ellas, cuando quizás solo se está interesado en una pequeña parte de la misma. Esto puede permitir reforzar fácilmente el entrenamiento de modelos de aprendizaje supervisado con datos adicionales de aquellas regiones de la distribución de datos en las que su desempeño sea peor. En conclusión, las GANs no evitarían tener que programar un simulador, pero sí podrían permitir prescindir posteriormente de él para tratar de superar sus limitaciones y obtener datos más realistas, en menos tiempo, y de forma más selectiva.

En cuanto a las líneas de trabajo futuro, se propone: mejorar el método de evaluación, emplear datos de entrenamiento reales y perfeccionar y generalizar el proceso de generación. En primer lugar, es prioritario mejorar y sistematizar el método de evaluación de resultados, pues el actual resulta computacionalmente costoso y no permite entender bien el funcionamiento del generador. Es necesario evaluar el grado de realismo de las imágenes, su correlación con sus etiquetas y la diversidad de los conjuntos generados. Sería además conveniente poder comparar la distribución de ciertos parámetros físicos de interés en los datos reales y los generados. Una vez esto sea posible, sería deseable probar a entrenar los modelos con datos reales, en lugar de simulados, para así comparar el grado de realismo de los datos generados frente a los simulados. Esto, a su vez, podría permitir incluir el modelo auxiliar de CTLearn en el ciclo de entrenamiento: una vez se hayan entrenado el generador y el discriminador lo suficiente, se puede volver a entrenar el modelo auxiliar a partir de su estado en ese momento con las nuevas imágenes del generador (idealmente más realistas que las simuladas). Incluir así el modelo auxiliar también en el ciclo de entrenamiento podría tener efectos positivos en un entrenamiento posterior de los tres modelos.

[24] Véase por ejemplo [2].

Sería también interesante tratar de generalizar la generación de eventos a "arrays" de telescopios y telescopios de mayor tamaño. Para este propósito, y en general para mejorar la calidad de los resultados, puede ser interesante explorar los mecanismos de atención empleados en [30] y el escalado progresivo de las imágenes generadas presentado en [17].[25]

Referencias

[1] M. Arjovsky, S. Chintala, and L. Bottou, Wasserstein gan, (2017). https://doi.org/10.48550/ARXIV.1701.07875.

[2] L. Arrabito, K. Bernlöhr, J. Bregeon, M. Carrère, A. Khattabi, P. Langlois, D. Parello, and G. Revy, Optimizing cherenkov photons generation and propagation in corsika for cta monte-carlo simulations, (2020). https://doi.org/10.48550/ARXIV.2006.14927.

[3] K. Bernlöhr et al., Monte carlo design studies for the cherenkov telescope array, *Astroparticle Physics* **43** (2013), 171–188. https://doi.org/https://doi.org/10.1016/j.astropartphys.2012.10.002.

[4] A. Brill, B. Kim, D. Nieto, T. Miener, and Q. Feng, CTLearn: Deep learning for imaging atmospheric Cherenkov telescopes event reconstruction, 2022. https://doi.org/10.5281/zenodo.5947837.

[5] A. Brock, J. Donahue, and K. Simonyan, Large scale gan training for high fidelity natural image synthesis, (2018). https://doi.org/10.48550/ARXIV.1809.11096.

[6] J. Brownlee, *Generative Adversarial Networks with Python: Deep Learning Generative Models for Image Synthesis and Image Translation*, Machine Learning Mastery, 2019.

[7] V. Chekalina, E. Orlova, F. Ratnikov, D. Ulyanov, A. Ustyuzhanin, and E. Zakharov, Generative models for fast calorimeter simulation: the lhcb case, *EPJ Web of Conferences* **214** (2019), 02034. https://doi.org/10.1051/epjconf/201921402034.

[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, (2016). https://doi.org/10.48550/ARXIV.1606.03657.

[9] J. Dubenskaya, A. Kryukov, and A. Demichev, Fast simulation of gamma/proton event images for the TAIGA-IACT experiment using generative adversarial networks, *PoS* **ICRC2021** (2021), 874. https://doi.org/10.22323/1.395.0874.

[10] J. Dubenskaya, A. Kryukov, and A. Demichev, Modeling Images of Proton Events for the TAIGA Project Using a Generative Adversaria Network: Features of the Network Architecture and the Learning Process, *PoS* **DLCP2021** (2021), 011. https://doi.org/10.22323/1.410.0011.

[11] M. Erdmann, L. Geiger, J. Glombitza, and D. Schmidt, Generating and refining particle detector simulations using the wasserstein distance in adversarial networks, (2018). https://doi.org/10.48550/ARXIV.1802.03325.

[12] I. Goodfellow, Nips 2016 tutorial: Generative adversarial networks, (2016). https://doi.org/10.48550/ARXIV.1701.00160.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[25] Estos trabajos, entre otros igualmente interesantes como [5], se explican en los últimos capítulos de [6].

[14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial networks, (2014). https://doi.org/10.48550/ARXIV.1406.2661.

[15] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, Improved training of wasserstein gans, (2017). https://doi.org/10.48550/ARXIV.1704.00028.

[16] A. Hillas, Evolution of ground-based gamma-ray astronomy from the early days to the cherenkov telescope arrays, *Astroparticle Physics* **43** (2013), 19–43. https://doi.org/10.1016/j.astropartphys.2012.06.002.

[17] T. Karras, T. Aila, S. Laine, and J. Lehtinen, Progressive growing of gans for improved quality, stability, and variation, (2017). https://doi.org/10.48550/ARXIV.1710.10196.

[18] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, How to train your dragan, (2017). https://doi.org/10.48550/ARXIV.1705.07215.

[19] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, Are gans created equal? a large-scale study, (2017). https://doi.org/10.48550/ARXIV.1711.10337.

[20] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, Least squares generative adversarial networks, (2016). https://doi.org/10.48550/ARXIV.1611.04076.

[21] T. Miener, D. Nieto, A. Brill, S. Spencer, and J. L. Contreras, Reconstruction of stereoscopic cta events using deep learning with ctlearn, 2021. https://doi.org/10.48550/ARXIV.2109.05809.

[22] M. Mirza and S. Osindero, Conditional generative adversarial nets, (2014). https://doi.org/10.48550/ARXIV.1411.1784.

[23] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, Spectral normalization for generative adversarial networks, (2018). https://doi.org/10.48550/ARXIV.1802.05957.

[24] A. Odena, C. Olah, and J. Shlens, Conditional image synthesis with auxiliary classifier gans, (2016). https://doi.org/10.48550/ARXIV.1610.09585.

[25] L. de Oliveira, M. Paganini, and B. Nachman, Learning particle physics by example: Location-aware generative adversarial networks for physics synthesis, *Computing and Software for Big Science* **1** no. 1 (2017). https://doi.org/10.1007/s41781-017-0004-6.

[26] L. de Oliveira, M. Paganini, and B. Nachman, Controlling physical attributes in GAN-accelerated simulation of electromagnetic calorimeters, *Journal of Physics: Conference Series* **1085** (2018), 042017. https://doi.org/10.1088/1742-6596/1085/4/042017.

[27] M. Paganini, L. de Oliveira, and B. Nachman, Accelerating science with generative adversarial networks: An application to 3d particle showers in multilayer calorimeters, *Physical Review Letters* **120** no. 4 (2017). https://doi.org/10.1103/physrevlett.120.042003.

[28] A. Radford, L. Metz, and S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, (2015). https://doi.org/10.48550/ARXIV.1511.06434.

[29] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, Improved techniques for training gans, (2016). https://doi.org/10.48550/ARXIV.1606.03498.

[30] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, Self-attention generative adversarial networks, (2018). https://doi.org/10.48550/ARXIV.1805.08318.