Generación de imágenes a partir de texto usando red generativa antagónica

Ronald Nicolas Saenz Chuqui
Universidad Nacional de Ingeniería
Lima, Perú
rsaenzc@uni.pe

Clisman Jesús Adahiton Moreno Marchena
Universidad Nacional de Ingeniería
Lima, Perú
cmorenom@uni.pe

Vilchez Diaz Vicente Jesus
Universidad Nacional de Ingeniería
Lima, Perú
vicente.vilchez.d@uni.pe

***Resumen*—**

Palabras claves— GANs, Redes Generativas Antagónicas

I. Introducción

Uno de los grandes desafíos dentro del campo de la Inteligencia Artificial es sintetizar imágenes a partir de texto y su investigación es tomada desde diferentes perspectivas. La idea central es crear imágenes realistas, sin perder la semántica del texto.

Los motivos de sintetizar imágenes a partir de texto puede ser crear fotos nuevas en base a la inteligencia artificial, sin mencionar el avance de la AI en poder simular no solo nuestra inteligencia sino también nuestra imaginación.

Actualmente el uso de algoritmos de aprendizaje no supervisado, desarrolla sistemas de redes neuronales para la clasificación de imágenes y dentro de ellos podemos encontrar las 'Redes generativas antagónicas' con las siglas GANs. lo cuales son un una clase de algoritmos compuesto por dos redes neuronales convolucionales [1] que compiten mutuamente hasta llegar en una especie de juego de suma cero;el estudio de las redes GANs comenzó en el 2014, cuando Ian J. Goodfellow y coautores introdujeron en el artículo *GenerativeAdversarialNets* [1].

Los GANs consiste en dos modelos de redes neuronales convolucionales,llamados red neuronal generador y red neuronal discriminador, son entrenamos simultáneamente para competir el uno con el otro, por un lado, el generador es entrenado para generar datos falsos lo más parecidos posibles a los ejemplos reales de un determinado conjunto de entrenamiento que se selecciona. Por otro lado, el discriminador es entrenado para ser capaz de discernir los datos falsos producidos por el generador de aquellos que corresponden al conjunto de entrenamiento (los ejemplos reales). Sucesivamente, los dos modelos tratan continuamente de superarse: cuanto mejor es el generador en la creación de datos convincentes, mejor debe ser el discriminador para distinguir los ejemplos reales de los falsos2.

El objetivo de éste proyecto es desarrollar una arquitectura de red GANs simple para sintetizar una imagen a partir de una descripción detallada en texto a partir de un conjunto de imágenes de gatos, nuestro modelo será entrenado en un subconjunto de categorías de entrenamiento por la red neuronal, la primera red neuronal será entrenado con una serie de caracteres de texto y la segunda Red Discriminadora que será entrenado con conjunto de imágenes de gatos llamado Generated using LSUN Cat dataset at 256×256. Además de imágenes de gatos, nuestro modelo será aplicado a otros tipos de imágenes dentro del conjunto de datos llamado Generated using LSUN Car dataset at 512×384.

Para el desarrollo del proyecto utilizamos el lenguaje de programación Python en cuál usamos las siguientes librerías que serán detallados a continuación:

- PyTorch : Es una biblioteca de aprendizaje automático de código abierto basada en la biblioteca de Torch, utilizado para aplicaciones que implementan cosas como visión artificial y procesamiento de lenguajes naturales.
- Theano : Es una biblioteca de Python que nos permite evaluar operaciones matemáticas, incluidas matrices multidimensionales, de manera tan eficiente.
- Scikit-learn : Es una biblioteca para aprendizaje automático de software libre que incluye varios algoritmos de clasificación, regresión y análisis de grupos.
- NLTK : Es un módulo de Python que contiene muchas funciones diseñadas para su uso en el análisis lingüístico de documentos y en el procesamiento de lenguaje natural.

El resto del documento se ensambla de la siguiente manera. En la sección 2 se desarrolla la parte teórica sobre las redes GANs y su arquitectura, la sección 3 dispensa sobre el estado del arte.A continuación, la sección 4 explica sobre lo métodos a usar para la realización del trabajo, la sección 5 usamos los algoritmos de aprendizaje no supervisado para implementar las redes neuronales además de mostrar los resultados. La sección 6 muestra una discusión de resultados en base

a la experimentación, la sección 7 mencionamos las conclusiones y finalmente en la sección 8 agregamos los posibles futuros trabajos o mejoras a lo desarrollado.

II. Fundamento teórico

II-A. Espacio muestral, variable aleatoria y probabiliad

Se considera un conjunto S, denominado *espacio muestral*, que consiste en un determinado número de elementos. A cada uno de los subconjuntos A de S se le asigna un número real $P(A)$, denominado probabilidad, que se define de acuerdo a los axiomas de Kolmogorov.
Una variable que toma un valor específico para cada elemento del conjunto S se denomina *variable aleatoria*.

II-B. Probabilidad condicional

Si se tienen dos subconjuntos A y B contenidos en el espacio muestral S y en el cual $P(B) \neq 0$, entonces se puede definir la *probabilidad condicional* $P(A|B)$ *como* :

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Además, dos subconjuntos A y B son independientes si

$$P(A \cap B) = P(A)P(B)$$

II-C. Teorema de Bayes

A partir de lo anterior, es posible demostrar que

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

II-D. Reconocimiento de patrones

El objetivo del reconocimiento de patrones es clasificar un determinado es clasificar un determinado patrón x a una clase previamente especificada y.
En el problema de reconocimiento de patrones, el suboconjunto B (o los datos) suele representarse como un vector x y es llamado patrón, vector de características o variable de entrada. De modo equivalente, el subconjunto A (o la hipótesis) suele representarse como una variable y y se denomina variable de salida, *target* o categoría. Es la clase a la cual el patrón x pertenece.

En el enfoque discriminativo se estima la probabilidad de una determinada variable de salida y, o clase, dado el patrón x, mientras que en los enfoques generativos se estima la probabilidad de observar un determinado patrón x, luego de especificada una clase y.

II-E. Redes neuronales profundas

La gran mayoría de sistemas de aprendizajes profundos son redes neuronales y es por este motivo que, en general, el término *Deep Learning* hace alusión a redes neuronales profundas. Sin embargo, cabe destacar que cualquier sistema que emplee múltiples capas (como por ejemplo, las máquinas de Blotzmann) para poder aprender las representaciones complejas de los datos de entrada también se considera *Deep Learning*.

Una red neuronal profunda es una red neuronal con múltiples capas entre las capas de entrada y de salida que son llamadas capas ocultas o *hidden layers*. Cada capa contiene unidades, que están conectadas a las unidades de la capa anteriora través de un conjunto de pesos. El propósito de estas capas ocultas radica

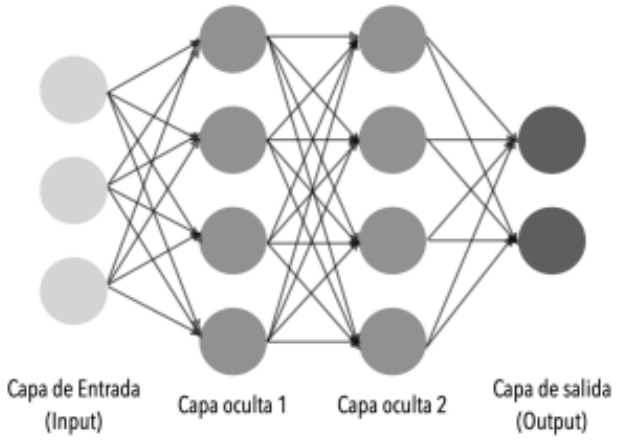

Figura 1. Ejemplo de red neuronal con dos capas ocultas densas.

en que, al combinar las distintas capas, las unidades de cada capa posterior pueden representar aspectos cada vez más sofisticados de la entrada original. El entrenamiento de la red es el proceso mediante el cual se busca cuál es el conjunto de pesos de cada capa que permite establecer predicciones más precisas.

II-F. Funciones de activación

La salida de una determinada unidad y es la suma pesada de las entradas x_i que recibe de capas anteriores. Esta suma es la variable de entrada de una función de activación f_{act} no lineal antes de ser enviada a la siguiente capa. Esto permite que la red aprenda patrones complejos, evitando que la salida sea una sencilla combinación lineal de las entradas que recibe.

$$y = f_{act}(\sum_i (w_i x_i + b)),$$

donde w_i representan los pesos de ls conexiones y b representa un valor de sesgo o *bias*.
Las redes neuronales profundas pueden ser formuladas a través de una función $f_\theta(x)$ parametrizada por θ, que contiene todos los parámetros de todas las unidades de la red.

II-G. Entrenamiento de la red

La red es entrenada con el fin de encontrar el conjunto de parámetros θ óptimo. El problema de aprendizaje

de la red nuronal se formula en términos de la minimización de una función de pérdida L o $loss\ function$ que compara la salida predicha por la red con los datos reales.
El proceso de entrenamiento consta de una secuencia iterativa de pasos que podrían resumirse de la siguiente forma:

- *Propagación hacia adelante*: La red recibe muestras de datos del conjunto de entrenamiento y, luego de ser procesados por cada unidad de la red, los resultados se propagan hacia adelante hasta llegar a un $output$ o valor(es) final(es).
- *Medida del error*: En este paso se calcula la función de pérdida L. Esta función retorna un valor promedio por cada ejemplo de entrenamiento y cuanto mayor sea, indica que peor se ha comportado la red para esa observación. Tres de las funciones de pérdida más comunes son el error cuadrático medio, la entropía cruzada categórica y la entropía cruzada binaria. La elección de cada una depende de cada caso en particular.
- *Retropropagación*: El algoritmo de retropropagación permite calcular el gradiente de la función de pérdida media respecto a los parámetros de la red, permitiendo encontrar los valores de los parámetros que minimicen la función de pérdida.
- *Actualización de los valores de los parámetros*: Una vez se pasan todos los ejemplos de entrenamiento a la red y se obtienen los gradientes de las funciones de pérdida respecto de cada parámetro de la red, éstos son promediados a través de todos los ejemplos de entrenamiento.
  Finalmente, se actualizan los valores de los parámetros de tal manera que intentamos minimizar la función de pérdida. Así, la actualización de los parámetros se establece como:

$$\theta \leftarrow \theta - \eta g,$$

  donde η es el parámetro *learning rate* que denota en qué cantidad debe moverse el valor del parámetro a ser actualizado en la dirección de minimización de la función de pérdida y g es el promedio de los gradientes respecto al parámetro a actualizar a lo largo de todo el conjunto de entrenamiento:

$$g = \frac{1}{n} \sum_{i}^{n} \nabla_\theta L_i.$$

  Como deben ajustarse muchos parámetros resulta importante poder entrenar los modelos rápidamente. Para ello existen los $optimizadores$, que son algoritmos que se utilizan para actualizar los pesos de la red neuronal basado en el gradiente de la función pérdida. Existen muchos optimizadores, entre ellos: Gradiente descendiente estocástico, RMSprop, Adam, Adagrad, Adamax, etc2 .

III. Estado del Arte

III-A. TAC-GAN – Text Conditioned Auxiliary Classifier Generative Adversarial Network [3]

En éste trabajo presentan una TAC-GAN que es una red GAN de texto a imágenes para sintetizar imágenes a partir de de sus descripciones de texto, el modelo TAC-GAN es desarrollado en el conjunto de datos de flores Oxford102 y evaluan la discriminabilidad de las imágenes generadas con Inception-Score, así como su diversidad utilizando el índice de similitud estructural de múltiples escalas (MS-SSIM).

III-A1. Método: Para implementar una TAC-GAN usan una implementación en TensorFlow de una Red Adversarial Generativa Convolucional Profunda (DC-GAN), en la que G se modela como una Red Neural Deconvolucional (convoluciones fraccionadas) y D se modela como una red neuronal convolucional (CNN). Para la función de incrustación de texto ψ, usan vectores de omisión de pensamiento para generar un vector de incrustación de tamaño Nt.
La red de generadores está compuesta por tres capas convolucionales transpuestas con 256, 128 y 64 mapas de filtro, respectivamente.
La salida de cada capa tiene un tamaño dos veces mayor que el de las imágenes que reciben como entrada.
D se compone de tres capas convolucionales con 128, 256 y 384 mapas de filtro, respectivamente.
La salida de la última capa es MD. El tamaño del grano de todas las capas convolucionales hasta la generación de MD es 55, después de la concatenación entre MD y el lr replicado espacialmente, la última capa convolucional se compone de 512 mapas de filtro de tamaño 11 y paso 1.
En el entrenamiento, LD_s denota la pérdida de entrenamiento relacionada con la fuente de la entrada (real, falsa o incorrecta) que se calcula como una suma de de la entropía cruzada binaria, entre la salida del discriminador y el valor deseado para cada una de las imágenes y el LD_C, denota la pérdida de entrenamiento relacionada con la clase a la que se supone que pertenece la imagen de entrada. De es forma el discriminador minimiza el objetivo de entrenamiento $LD_C + LD_S$.

III-A2. Conclusión: Comparan los resultados de su enfoque con los del modelo StackGAN, para demostrar que el modelo produce muestras muy diversas, utilizan el MS-SSIM. Indicando que su modelo produce imágenes más diversas que las de los datos de entrenamiento, lo que corrobora su hipótesis.
Además, muestran que el modelo confirma los hallazgos en otros enfoques, es decir, aprende representaciones separadas para el estilo y el contenido de las imágenes generadas produciendo imágenes interpoladas entre dos vectores de ruido diferentes, mientras se mantiene el mismo texto de entrada.
Finalmente indican que el modelo es fácilmente exten-

sible: al punto que es posible condicionar las redes no solo en texto, sino en cualquier otro tipo de información potencialmente útil.

III-B. Síntesis generativa de texto a imagen antagónica (Generative Adversarial Text to Image Synthesis) [4]

III-B1. Método: entrenar una profunda red convolucional generativa adversaria (DC-GAN) condicionada por características de texto codificadas por una red neuronal recurrente convolucional híbrida a nivel de carácter. Tanto la red de generadores G como la red de discriminadores D realizan inferencias de retroalimentación condicionadas a la característica de texto.
La forma más sencilla de entrenar una GAN condicional es ver los pares (texto, imagen) como observaciones conjuntas y entrenar al discriminador para juzgar los pares como reales o falsos. Este tipo de condicionamiento es ingenuo en el sentido de que el discriminador no tiene una noción explícita de si las imágenes de entrenamiento reales coinciden con el contexto de inserción del texto.
Sin embargo, la dinámica del aprendizaje puede ser diferente del caso no condicional. Al comienzo del entrenamiento, el discriminador ignora la información de condicionamiento y fácilmente rechaza las muestras de G porque no parecen plausibles. Una vez que G ha aprendido a generar imágenes plausibles, también debe aprender a alinearlas con la información de condicionamiento y, de la misma forma, D debe aprender a evaluar si las muestras de G cumplen con esta restricción de condicionamiento.
En GAN ingenuo, el discriminador observa dos tipos de entradas: imágenes reales con texto coincidente e imágenes sintéticas con texto arbitrario. Por lo tanto, debe separar implícitamente dos fuentes de error: imágenes poco realistas (para cualquier texto) e imágenes realistas de la clase incorrecta que no coinciden con la información condicionante. Basándonos en la intuición de que esto puede complicar la dinámica de aprendizaje, modificamos el algoritmo de entrenamiento GAN para separar estas fuentes de error. Además de las entradas reales / falsas al discriminador durante el entrenamiento, agregamos un tercer tipo de entrada que consiste en imágenes reales con texto no coincidente, que el discriminador debe aprender a puntuar como falso. Al aprender a optimizar la coincidencia de imagen / texto además del realismo de la imagen, el discriminador puede proporcionar una señal adicional al generador.

III-B2. Conclusiones: En este trabajo desarrollamos un modelo simple y efectivo para generar imágenes a partir de descripciones visuales detalladas. Demostramos que el modelo puede sintetizar muchas interpretaciones visuales plausibles de una leyenda de texto determinada. Nuestro regularizador de interpolación múltiple mejoró sustancialmente la síntesis de texto a imagen en CUB. Mostramos cómo desenredar el estilo y el contenido, y la postura del pájaro y la transferencia de fondo de las imágenes de consulta a las descripciones de texto. Finalmente, demostramos la posibilidad de generalizar nuestro enfoque para generar imágenes con múltiples objetos y fondos variables con nuestros resultados en el conjunto de datos MS-COCO. En el trabajo futuro, nuestro objetivo es ampliar aún más el modelo a imágenes de mayor resolución y agregar más tipos de texto.

III-C. StackGan: síntesis de texto a imagen fotorrealista con Redes Adversarias Generativas Apiladas [5]

III-C1. Método: En el artículo se proponen Redes Adversarial Generativas Apiladas (StackGAN) para generar 256×256 imágenes fotorrealistas condicionadas a descripciones de texto. Descomponemos el problema difícil en subproblemas más manejables a través de un proceso de refinamiento del bosquejo. La GAN de la etapa I esboza la forma y los colores primitivos del objeto en función de la descripción de texto dada, lo que produce imágenes de baja resolución de la etapa I. La GAN Stage-II toma los resultados de la Stage-I y las descripciones de texto como entradas, y genera imágenes de alta resolución con detalles fotorrealistas. Es capaz de rectificar defectos en los resultados de la Etapa I y agregar detalles convincentes con el proceso de refinamiento. Para mejorar la diversidad de las imágenes sintetizadas y estabilizar el entrenamiento de la GAN condicional, presentamos una novedosa técnica de Acondicionamiento Aumento que fomenta la suavidad en la variedad de acondicionamiento latente.

III-C2. Conclusiones: Proponen una novedosa Redes Adversarial Generativas Apiladas para sintetizar fotorrealista imágenes de descripciones de texto. Descompone el difícil problema de generar imágenes de alta resolución en subproblemas más manejables y mejora significativamente el estado de la técnica. StackGAN genera por primera vez imágenes de 256× Resolución de 256 con detalles fotorrealistas de descripciones de texto. Se propone una nueva técnica de Acondicionamiento Aumento para estabilizar el entrenamiento condicional GAN y también mejora la diversidad de las muestras generadas. Amplios experimentos cualitativos y cuantitativos demuestran la efectividad del diseño general del modelo, así como los efectos de los componentes individuales, que proporcionan información útil para diseñar futuros modelos de GAN condicionales.

IV. Metodología

En esta sección se describen las herramientas y metodología que se usarán en el presente trabajo:

IV-A. Pytorch-GANs [6]

PyTorch es una biblioteca de aprendizaje automático de código abierto basada en la biblioteca de Torch, utilizado para aplicaciones que implementan cosas como visión artificial y procesamiento de lenguajes naturales.

Si bien PyTorch no proporciona una implementación incorporada de una red GAN, proporciona primitivas que le permiten construir redes GAN, incluidas capas de redes neuronales completamente conectadas, capas convolucionales y funciones de entrenamiento.

IV-B. Apex

Es una extensión de Pytoch que proporciona métodos convenientes para precisión mixta, donde algunas operaciones usan el tipo de datos torch.float32(float) y otras operaciones usan torch.float16(half). Algunas operaciones, como las capas lineales y las convoluciones, son mucho más rápidas float16. Otras operaciones, como las reducciones, a menudo requieren el rango dinámico de float32. La precisión mixta intenta hacer coincidir cada operación con su tipo de datos apropiado, lo que puede reducir el tiempo de ejecución y la huella de memoria de su red.

IV-C. Torchvision

Torchvision es una biblioteca para Computer Vision que va de la mano con PyTorch. Tiene utilidades para transformaciones eficientes de imagen y video, algunos modelos pre-entrenados de uso común y algunos conjuntos de datos. Es muy usado en aplicaciones de redes GANs dado que nos proporciona modelos de entrenamiento más asequibles al momento de implementar una red GAN y por ende comprender su funcionamiento.

IV-D. Argparse [7]

Argparse es una biblioteca completa de procesamiento de argumentos, el cual incluye herramientas para construir procesadores de argumentos y opciones de línea de comandos.
El módulo argparse será utilizado para ejecutar el modelo en base a hiperparámetros de red como: el conjunto de datos a usar, Tasa de aprendizaje, Número de filtros de conv en la primera capa del generador, Número de filtros de conversión en la primera capa del discriminador, tamaño de imagen.

IV-E. Métricas

IV-E1. Puntuación Inicial [8]: La puntuación inicial o *Inception Score* mide cómo de reales"son los datos generados:

$$\mathbf{IS}(G) = \mathbf{exp}(\mathbb{E}_{x\sim p_g} D_{KL}(p(y|x)||p(y))$$

La ecuación tiene dos componentes $p(y|x)$ y $p(y)$. Aquí, x es la imagen que produce el Generador; $p(y|x)$ es la distribución de probabilidad que se obtiene cuando se paa la imagen "x" a tra´ves de una red de inicio preentrenada. También, $p(y)$ es la distribución de probabilidad marginal, que puede ser calculada promediando $p(y|x)$ sobre unas pocas muestras distintas de imágenes generadas. Estos dos términos representan dos cualidades diferentes que son deseables en imágenes reales:

- Que la imagen generada tenga objetos que sean significativos (objetos son claros y no borrosos). Esto significa que $p(y|x)$ debe tener "baja entropía". En otras palabras, nuestra inicial debe estar firmemente convencida de que la imagen generada pertenece a una clase particular.
- Las imágenes generadas deben ser diversas. Esto significa que $p(y)$ debe tener .ªlta entropía". En otras palabras, el generador debe produvir imágenes de manera que cada imagen represente a una etiqueta diferente (idealmente)

IV-E2. Distancia de Fréchet (FID) [9]: Un inconveniente del IS es que las estadísticas de los datos reales no se comparan con las estadísticas de los datos generados. La distancia de Fréchet (FID) resuelve ese inconveniente comparando la media y la covarianza de las imágenes reales y generadas. El FID realiza el mismo análisis que el IS, pero en los mapas de características producidos al pasar las imágenes reales y generadas a través de una red Inception-v3 previamente entrenada. La ecuación se describe a continuación:

$$\mathbf{FID} = ||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$$

El FID comapra la media y la covarianzade las distribuciones de datos reales y generados. "Tr"hace referencia a la traza.

IV-F. Metodología del trabajo

Para nuestro trabajo emplearemos principalmente las librerías Pytorch y Torchvision. Además, trabajaremos en:

1. Adaptar el modelo para dos redes neuronales y generar una competencia entre ambas como parte del proceso para llegar a resultados óptimos.
2. La red discriminadora tendrá que diferenciar las imágenes reales de las falsas.
3. La red generadora deberá ser capaz de engañar de manera consistente a la red discriminadora.
4. Evaluar y comparar los resultados con los datos reales del conjunto de datos usados.

V. Experimentación y resultados

VI. Discusión de resultados

VII. Conclusiones

VIII. Trabajos Futuros

Referencias

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[2] Laura Randa Calcagni. *Redes Generativas Antagónicas y sus aplicaciones*. PhD thesis, Universidad Nacional de La Plata, 2020.

[3] Ayushman Dash, John Cristian Borges Gamboa, Sheraz Ahmed, Marcus Liwicki, and Muhammad Zeshan Afzal. Tac-gan-text conditioned auxiliary classifier generative adversarial network. *arXiv preprint arXiv:1703.06412*, 2017.

[4] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069. PMLR, 2016.

[5] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[6] Minguk Kang, Woohyeon Shim, Minsu Cho, and Jaesik Park. Rebooting acgan: Auxiliary classifier gans with stable training. *arXiv preprint arXiv:2111.01118*, 2021.

[7] Maintainer Trevor L Davis. Package 'argparse'. 2015.

[8] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wassertein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[9] Johannes Kepler. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2018.