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:

- Tensorflow: es una biblioteca de código abierto que se basa en un sistema de redes neuronales. Esto significa que puede relacionar varios datos en red simultáneamente, de la misma forma que lo hace el cerebro humano.
- 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

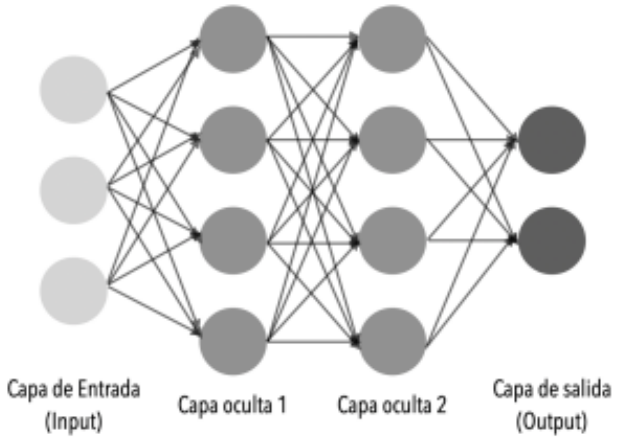

Figura 1. Ejemplo de red neuronal con dos capas ocultas densas.

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 .

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.