

Un CGAN interesante: de texto a imagen

Por los alumnos de matemáticas aplicadas: Gerardo Pastrana,
Jorge Rotter, Maximiliano Medina

4 de junio de 2020

Abstract

...She let the balloon float up into the air with her hopes and dreams

La creación de datos sintéticos ha ganado popularidad e importancia en los últimos años. Uno de los retos actuales es la producción de imágenes sintéticas. En este artículo se abordará la creación de imágenes a partir de descripciones breves de texto utilizando el dataset *Flickr8k*. Al igual que las soluciones más prometedoras de problemas similares, se utilizará una *CGAN* para atacar el problema. El fin de nuestra red no fue producir las imágenes más nítidas posibles, sino crear un modelo que fuera asequible para personas que incursionan por vez primera en el tema con fines didácticos. Debido a la gran complejidad de las imágenes y al objetivo establecido los resultados obtenidos son poco satisfactorios visualmente.

1. Introducción

El reto del proyecto es generar imágenes sintéticas a partir de texto, de manera que texto e imagen estén relacionadas. La complejidad del problema es elevada y actualmente no existen soluciones mas que prototípicas, por lo que podríamos considerarlo un problema abierto. Los modelos generativos profundos, y en particular las redes generativas antagónicas (*GAN* por sus siglas en inglés) han mostrado en los últimos años los resultados más prometedores[1]. En la forma más básica de estas redes, un generador de imágenes y un discriminador de imágenes se entrenan con objetivos enfrentados: el generador debe engañar al discriminador y el discriminador debe ser lo mejor posible [2]. Sobre esta idea se han construido arquitecturas especializadas como *StackGAN*[3, 4] y *AttnGAN*[5]. Cada una de esas redes convolucionales profundas incorpora mecanismos para mejorar la calidad y coherencia de la imagen sintética, que llega a ser de calidad fotográfica. Entre ellos están

apilar una *GAN* que bosqueje una imagen de baja calidad seguida de otra que corrija y añada detalles de alta resolución e incorporar mecanismos de atención para asociar regiones de la imagen con las palabras más relevantes en la oración. No obstante novedosas, las propuestas que existen son específicas, creadas a la medida de las bases de datos y sus características. Las implementaciones de estas arquitectura disponibles en línea están descuidadas y a veces no pueden ni reproducir los resultados originales.

Este trabajo presenta un *WCGAN* como generador de imágenes condicionado a texto, que utiliza a su vez un *embedding* de texto preentrenado. El generador es de capas de *LSTM* bidireccionales, densas y convolucionales, y el discriminador está formado por capas de densas esencialmente. Escogimos este modelo apegándonos al principio de parsimonia, dando preferencia a modelos más simples. Asimismo, nos sujetamos a restricciones de tiempo, recursos computacionales y de conocimiento. Las elecciones anteriores, aunque fortalecieron la base teórica de nuestro trabajo, debilitaron nuestros resultados finales. La calidad de la salida fue baja (de 64×64) y el contenido del mismo es difuso, si bien variado. Sin embargo, enfatizamos que de antemano preferimos favorecer el aprendizaje a tener resultados obtenidos ciegamente.

La estructura del trabajo es la siguiente: detallamos el problema, explicamos la propuesta de solución, presentamos los resultados experimentales y sintetizamos en la conclusión. Al final se incluye la arquitectura en anexo.

2. Problema

El problema general consiste crear un modelo capaz de producir imágenes sintéticas a partir de entradas de texto de tal manera que las imágenes producidas reflejen el texto. Para entrenar dicho modelo, así como para probarlo, se utiliza la base de datos *Flick8k*, la cual consiste de 8,091 imágenes, cada una con cinco descripciones o textos de la imagen [6]. A diferencia de los datos utilizados usualmente en generación de imágenes como *Birds* o *COCO*, las imágenes son muy variadas en contenido y forma. Tratan de temas muy diversos y la mayoría de las imágenes no focalizan el objeto de la imagen; es decir, el tema de la imagen se encuentra rodeado de ruido que el modelo debe aprender a ignorar.

Todo esto conlleva problemas mayores en la creación de la arquitectura del modelo. El primer, y mayor problema, es que las sofisticadas soluciones del estado del arte son poco accesibles para fines de este problema. La mayoría de las implementaciones no tienen mantenimiento, no están comentadas, no es directo adaptarlas para utilizar una base de datos distinta a su propósito original, además de basarse en lenguajes de programación poco utilizados. Los *benchmarks* actuales, en los cuales se basan las soluciones más prometedoras, utilizan bases de datos menos complejas que

el dataset *Flickr8k*. Todas ellas mantienen un tema homogéneo en todas las imágenes, el objeto que se desea reproducir está constantemente centrado dentro de la imagen, además de contar con un tamaño uniforme. No es de esperar que dichas implementaciones sean capaces de reproducir la misma calidad con otro tipo de imágenes más complejas. El segundo problema es que dichos modelos, por su enorme rebuscamiento, requieren de capacidades computacionales prohibitivas. Un tercer problema es que, debido a la enorme complejidad del fenómeno, crear una arquitectura propia para resolver el problema difícilmente obtendrá resultados satisfactorios.

Una particularidad de la generación de imágenes es su dificultad de evaluación. No es directo utilizar una métrica como error cuadrático medio o precisión, y necesita evaluarse tanto la calidad de la imagen como su relación con el texto. En [3] y [5] se introducen algunas medidas de evaluación para el problema, pero no las utilizamos en este trabajo.

3. Solución

Elegimos una *GAN* porque han mostrado mejores resultados que los autoencoders variacionales en generación de imágenes. En gran parte se debe a que ambos modelos realizan inferencia variacional en direcciones opuestas, que permiten a la *GAN* capturar modas específicas de la distribución a diferencia de la muestra de varias modas que aprenden los *autoencoders*. [7]. Después de dificultades usando implementaciones en línea de las arquitecturas más recientes y prometedoras, decidimos crear nuestra propia *GAN*. Es un modelo condicional [8] con pérdida de Wasserstein. Siguiendo a [9], usamos penalización de gradiente en vez de recorte de gradientes para asegurar la condición de Lipschitz con un entrenamiento más estable.

Antes de alimentar las imágenes, se hizo una interpolación bilineal a 64×64 píxeles, misma dimensión que las imágenes de salida. Se escogió esta representación porque la calidad de las imágenes de salida no daba para una mayor resolución. Para el texto se optó por cortar los textos a un máximo de 25 palabras, pues el 99.9% de los textos no contenían más de 25 palabras. Para aquellos textos que tuvieran menos de 25 palabras se realizó un *padding* con ceros al final. A continuación presentamos una breve explicación de nuestra arquitectura. Las imágenes de la misma se encuentran en el anexo.

Nuestra arquitectura fue construida con la mayor parte de los elementos aprendidos en clase. Para el *text embedding* utilizamos RoBERTa [10], usa de los ajustes de BERT. Ambos modelos utilizan *transformers* para entrenar *embeddings* contextualizados: cada vector representa no a la palabra, sino a la palabra en ese contexto particular. La librería de Python **transformers** tiene implementaciones utilizables de muchos modelos con *transformers*. Para el generador nos apoyamos primordialmente en el *embedding* de texto RoBERTa. Seguido de ello se utilizó un LSTM bidirec-

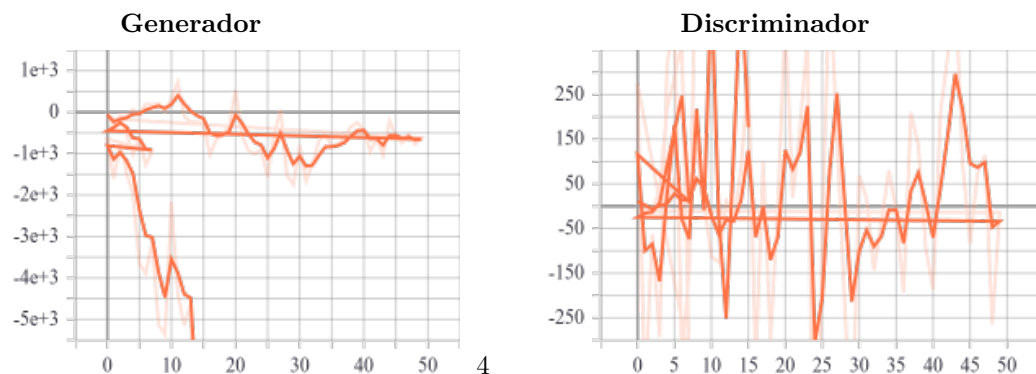
cional para capturar la información de las palabras acorde a su posición. El resultado se multiplicó matricialmente con ruido gaussiano para crear una interacción. Después de ello se utilizó una red *fully connected* y *upsampling* para la generación de imágenes. A lo largo de la red se usó *batch normalization*, como método de regularización. Por su parte, para el discriminador se utilizaron redes convolucionales *fully connected* para el tratamiento de las imágenes. En esta parte si se utilizó la concatenación para introducir la interacción entre la información del texto, previamente procesado con una capa *fully connected* y RoBERTa, y de la imagen. Finalmente, dicha concatenación se sumerge a una red *fully connected* para obtener el *score* final del discriminador. Para esta parte se escogió el *dropout* como método de regularización.

Esta implementación resultó en cerca de 86 millones de parámetros, de los cuales casi 82 eran RoBERTa y los marcamos como no-entrenables.

4. Experimentos y resultados


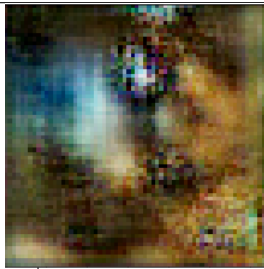
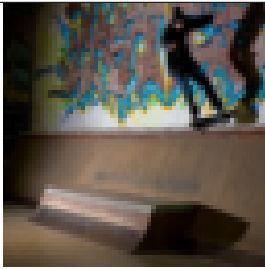
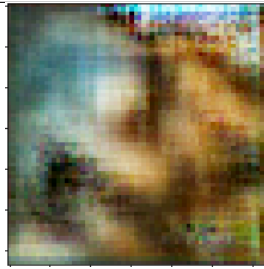
Primero, el espacio latente es tanto para el *embedding* de texto como para el ruido gaussiano; en lugar de concatenarlos, emparejamos sus dimensiones y fusionamos la información del texto en la imagen vía una multiplicación. Elegimos un espacio latente de dimensión de 128 solamente. Destacamos el enorme papel que juega aquí el *embedding* de texto (RoBERTa), pues por su preentrenamiento tan robusto, podemos incrustar las frases a un espacio relativamente pequeño. Al mismo tiempo, a diferencia de concatenar ambas entradas, hacer una multiplicación reduce el número de parámetros a entrenar y cumple el mismo objetivo. Adicionalmente, convino escoger ese espacio latente para incrementar las dimensiones naturalmente en el generador para que resultara una imagen de 64×64 .

Después, destacamos lo proporcionalmente rápido que fue entrenar el modelo. Tensorflow y Keras proveen las herramientas para entrenar en GPUs y TPUs, que acelera el entrenamiento radicalmente. Entrenamos durante 75 épocas. A continuación, está la gráfica de la pérdida de Wasserstein en las primeras 50 épocas. Recordando las propiedades de esta función de pérdida como distancia entre distribuciones de probabilidad, subrayamos algunos elementos.



Lo que ilustran estas gráficas son dos ideas: es más correcto llamar al discriminador un crítico, puesto que el soporte no son solo probabilidades y no clasifica; solamente asigna calificaciones mayores a las imágenes reales que a las falsas. En ese sentido, observamos un decrecimiento cíclico e inestable; esto quiere decir que la minimización se está llevando a cabo en una variedad sumamente compleja, por lo que el optimizador no puede seguir una dirección de descenso. En contraparte, el comportamiento de la pérdida en el generador no ejemplifica más que una traslación aleatoria de la pérdida del crítico o discriminador y por sí sola no nos permite concluir nada.

Posteriormente, evaluamos la calidad de la predicción del modelo dentro y fuera de la muestra. Por el lado positivo, la calidad en ambos casos es comparable; por otro, en ningún caso podemos recrear imágenes realistas. Un par de ejemplos de los resultados dentro de la muestra son los siguientes:

Texto	Imagen Real	Imagen sintética
A group of children are sitting on a raft wearing blue helmets and carrying paddles		
A skateboarder performs a skateboard trick against a graffiti wall		

Cuadro 1: Resultados dentro de la muestra

Finalmente, si bien las imágenes sintéticas no son realistas ni están claramente relacionadas con el texto, en un sentido más amplio nuestra *WCGAN* es capaz de generar imágenes diversas y no solamente monocromáticas o virtualmente idénticas. Por ello, arguimos que sí es capaz de extraer información del texto, a pesar de que no captura toda la complejidad ni puede plasmarla en el resultado. Un ejemplo del resultado de prueba fuera de la muestra es el siguiente.

Texto: *A woman meditating in a garden*

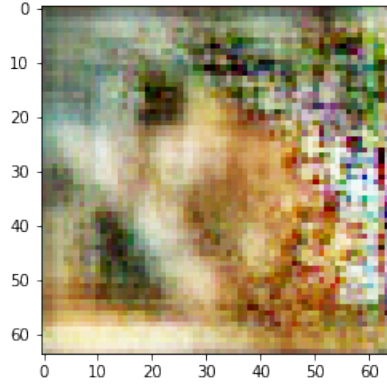


Figura 1: Resultado en la iteración 50

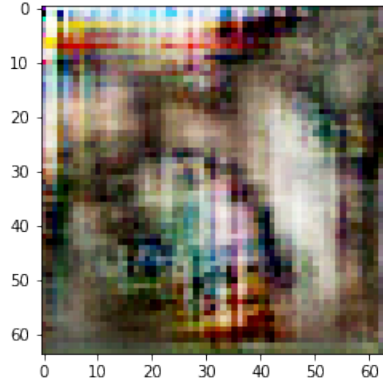


Figura 2: Resultado en la iteración 75

Observamos una evolución con el número de iteraciones pero la mejora es modesta. Los resultados objetivamente son adecuados; no se comparan con las arquitecturas en el estado del arte, pero corresponden a la complejidad de nuestro modelo y el tiempo de entrenamiento.

5. Discusión y conclusiones

Para sintetizar, este trabajo presenta una *WCGAN* que compagina muchos elementos aprendidos en el curso: desde una plétora de capas, funciones de activación y de pérdida, hasta un entendimiento plástico del funcionamiento de la misma, no en la vecindad de reglas de dedo, sino de un entendimiento profundo. Puntualmente, la red incorpora el *embedding* de texto preentrenado RoBERTa junto con capas densas, *LSTM* bidireccionales y convolucionales; tiene la arquitectura de una *CGAN* sencilla y usa la pérdida de Wasserstein.

Los resultados fueron adecuados para una red –comparativamente– pequeña, en el sentido que logró capturar información del texto, generando imágenes variadas. Sin embargo, las imágenes no se apegan al texto ni contienen elementos claramente distinguibles. El porqué se explica en varios niveles. Primero, la base de datos es mucho más complicada que otras que se usan para el mismo propósito; no tiene una temática, ni un tamaño, ni un encuadre de los objetos amigable. Segundo, modelos que sobresalen en esta tarea son prácticamente inaccesibles, pues son diseñados *ad hoc* a cada base y tienen un mantenimiento pobre. Tercero, la limitación de tiempo y conocimiento conjugada con la complejidad y especificidad de las arquitecturas y sus implementaciones, de la mano de nuestros recursos computacionales imposibilitaron que intentáramos modelos más sofisticados.

Por último, la mayor área de oportunidad es aprovechar los modelos que ya han sido probados y encontrados exitosos, si se cuenta con el tiempo y los recursos para utilizarlos. Por ejemplo, algunas posibles mejoras son hacer *fine-tuning* al *embedding* de texto, así como aumentar la profundidad o

anchura de la red a la par de incorporar mecanismos de atención o bloques residuales para mejorar la calidad y tamaño de las imágenes. Los resultados alcanzados con estas técnicas están a niveles futurísticos, superhumanos en ocasiones; distinguir de lo orgánico y lo sintético se dificulta tanto, que hasta al lector más perspicaz se le habrá dificultado identificar que la primer línea de este trabajo fue escrita artificialmente por un generador de texto.

Referencias

- [1] Jorge Agnese, Jonathan Herrera, Haicheng Tao, and Xingquan Zhu. A survey and taxonomy of adversarial neural networks for text-to-image synthesis, 2019.
- [2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.
- [3] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks, 2016.
- [4] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks, 2017.
- [5] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks, 2017.
- [6] Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 2013.
- [7] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P. Xing. On unifying deep generative models, 2017.
- [8] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.
- [9] A. S. Trotter, D. E. Reichart, R. E. Egger, J. Stýblová, M. L. Paggen, J. R. Martin, D. A. Dutton, J. E. Reichart, N. D. Kumar, M. P. Maples, B. N. Barlow, T. A. Berger, A. C. Foster, N. R. Frank, F. D. Ghigo, J. B. Haislip, S. A. Heatherly, V. V. Kouprianov, A. P. LaCluyzé, D. A. Moffett, J. P. Moore, J. L. Stanley, and S. White. The fading of cassiopeia a, and improved models for the absolute spectrum of primary radio calibration sources. 2017.

- [10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

6. Anexo

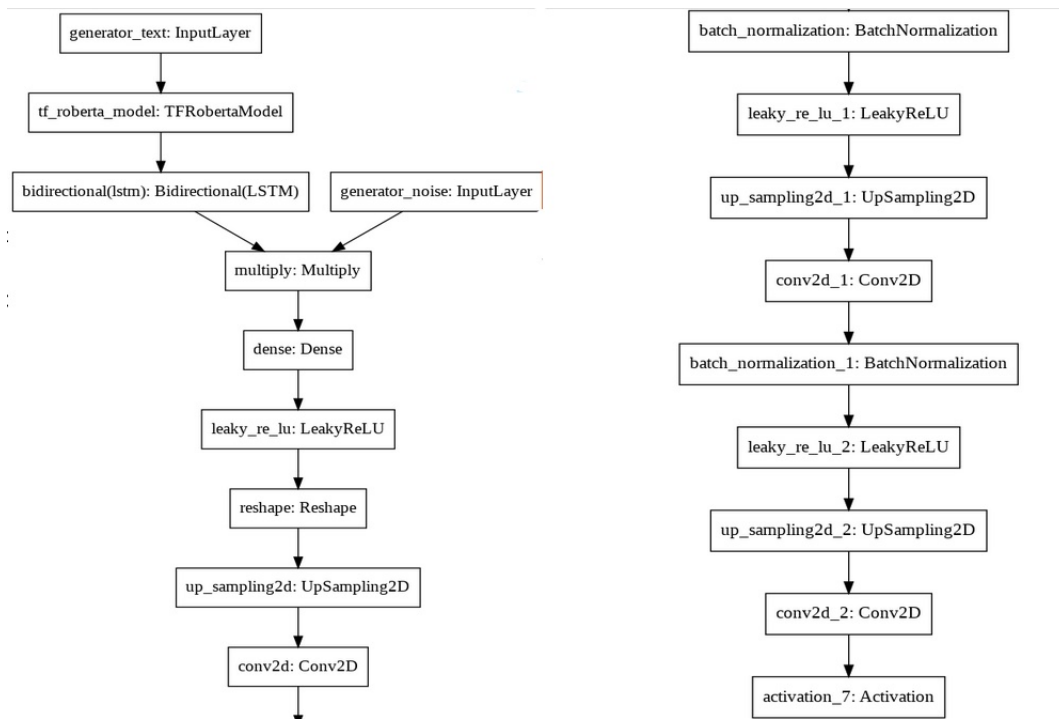


Figura 3: Arquitectura del generador

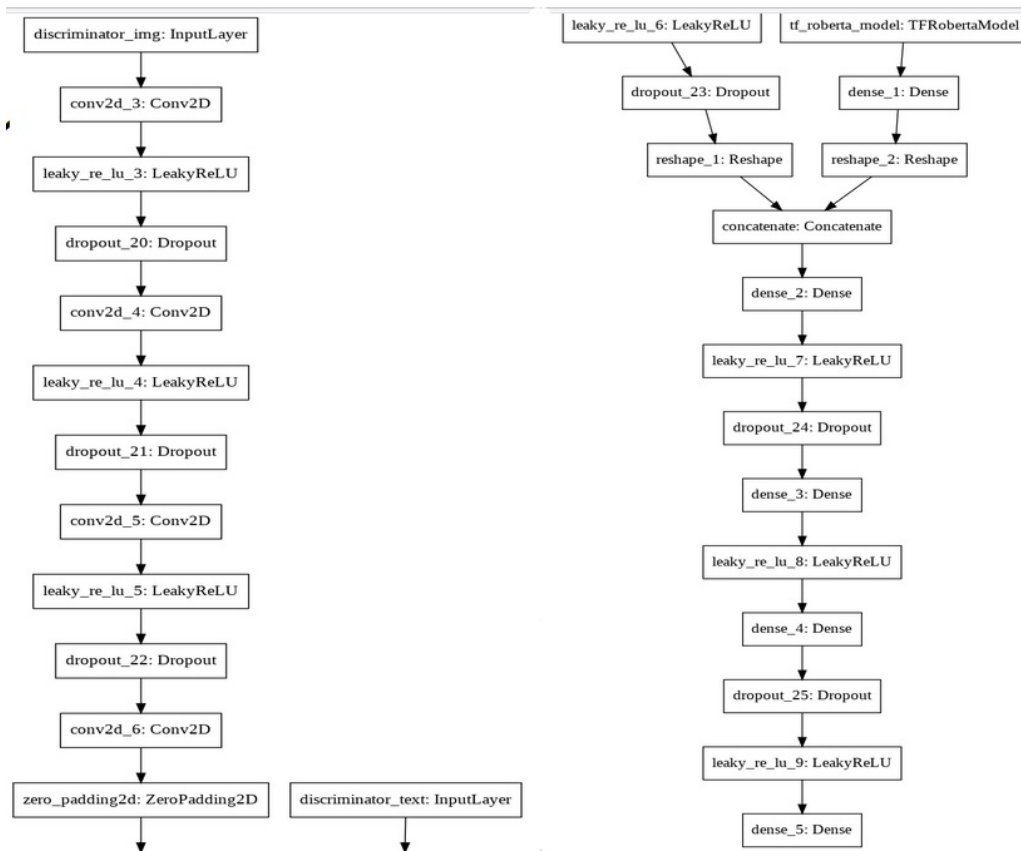


Figura 4: Arquitectura del discriminador