

Generación de imágenes a partir de texto

Informe técnico

Vision Transformers - CEIA - FIUBA

Alumnos:

- Fabricio Denardi
- Valentín Pertierra
- Leandro Saraco
- Sofía Speri

Fecha: diciembre 2024

Índice

Introducción	3
Motivación	3
Primeros pasos	4
Versatile Diffusion	4
DiffusionCLIP-CelebA_HQ	4
Stable Diffusion	5
Desarrollo y decisiones de diseño	7
Entorno	7
Dataset	7
Modelo	8
Fine tuning	8
Data augmentation y transformaciones	9
Función de pérdida	9
Inferencia utilizando el modelo fine-tuned	11
Evaluación	11
Análisis de los resultados	13
Evaluación usando CLIP	13
Evaluación usando BLIP	15
Conclusiones	18

Introducción

En este trabajo se presenta el uso de la arquitectura *Stable Diffusion*, basada en Latent Diffusion Models, para la generación de imágenes a partir de texto; adaptando el modelo preentrenado mediante fine-tuning para lograr un estilo en particular en las imágenes generadas. Además, se muestra que, mediante el uso de técnicas como la cuantización, es posible entrenar el modelo *Stable Diffusion 2* en hardware con recursos limitados.

El desempeño del modelo ajustado se evalúa utilizando métricas basadas en *CLIP* y *BLIP*, asegurando un análisis detallado de la calidad y coherencia entre las imágenes generadas y las descripciones textuales.

Motivación

Elegimos la tarea de “Generación de imágenes a partir de un texto” para desafiarnos a hacer algo diferente a las tareas tradicionales vistas en las asignaturas de visión de computadora anteriores, como clasificación, detección y segmentación.

Esta elección nos trajo muchos retos, tales como la selección de un dataset que contenga textos e imágenes, la elección del propio modelo base e incluso las tareas de fine tuning. Todas estas problemáticas las iremos comentando durante este informe.

Adicionalmente, tareas nuevas (para nuestro estudio), requieren métricas nuevas, lo que nos llevó a explorar CLIP y BLIP.

Primeros pasos

El primer paso en nuestro trabajo fue seleccionar un modelo preentrenado que pudiera servir como base para la tarea de generación de imágenes a partir de texto. Inicialmente, consideramos los siguientes modelos:

- DiffusionCLIP (https://huggingface.co/gwang-kim/DiffusionCLIP-CelebA_HQ)
- VersatileDiffusion (<https://huggingface.co/shi-labs/versatile-diffusion>)

Sin embargo, durante la implementación, nos encontramos con múltiples dificultades técnicas que no eran conceptuales, sino técnicas, por falta de documentación clara, errores en las bibliotecas utilizadas y complejidad en la configuración de los modelos.

Ante esta situación, optamos por cambiar el enfoque y trabajar con un modelo más conocido y robusto como Stable Diffusion, que cuenta con buena documentación y una integración sólida con herramientas de Hugging Face. Este cambio nos permitió continuar con el trabajo práctico de manera eficiente y enfocar nuestros esfuerzos en la adaptación y evaluación del modelo seleccionado.

A continuación abordaremos de manera sintética algunos de los inconvenientes presentados en la implementación de los modelos:

Versatile Diffusion

Al instanciar el modelo, obtuvimos el siguiente error:

ValueError: text_unet/versatile_diffusion.py as defined in model_index.json does not exist in shi-labs/versatile-diffusion and is not a module in 'diffusers/pipelines'.

Consultamos tanto el foro del modelo en Hugging Face como al creador directamente. Sin embargo, no obtuvimos respuestas favorables que permitieran resolver el inconveniente. Tras investigar, llegamos a la conclusión de que el error podría estar relacionado con incompatibilidades entre las versiones de las bibliotecas utilizadas. Una solución podría haber sido gestionar las dependencias mediante herramientas como Poetry u otros gestores de versiones. Sin embargo, dado que resolver este tipo de conflictos técnicos escapa al objetivo principal de la materia, y teniendo en cuenta la complejidad de usar Poetry sobre Colab, decidimos descartar este modelo y continuar con una alternativa mejor documentada.

DiffusionCLIP-CelebA_HQ

Este modelo pre entreno, toma las bases de Diffusion Clip vistas en clase para el entrenamiento (<https://github.com/gwang-kim/DiffusionCLIP>)

DiffusionCLIP-CelebA_HQ solo ofrece un checkpoint en Pytorch y se dificulta su integración con las librerías de Hugging Face utilizadas en la materia. Además, requería tareas adicionales de preprocesamiento y codificación que excedían los tiempos disponibles.

Aunque no fue viable para este proyecto, es un modelo interesante para explorar en el futuro.

Stable Diffusion

Si bien es el modelo elegido, se evaluaron dos versiones del mismo (v2 y v3). Se presentaron ciertos inconvenientes técnicos para el entrenamiento para fine tuning, que detallamos a continuación:

1. Limitaciones de recursos: Al trabajar en entorno de Google Colab, se tienen recursos de hardware limitados. Dentro de los cuáles se tuvo que reducir el batch size de entrenamiento.

```
    return F.group_norm(input, self.num_groups, self.weight, self.bias, self.eps)
File "/usr/local/lib/python3.10/dist-packages/torch/nn/functional.py": line 2055, in group_norm
    return torch.group_norm(
torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to allocate 60.00 MiB. GPU 0 has a total capacity of 14.75 GiB of
Steps:  2% 1/50 [00:12<09:54, 12.13s/it, lr=3e-5, step_loss=0.115]
Traceback (most recent call last):
  File "/usr/local/bin/accelerate", line 8, in <module>
    sys.exit(main())
  File "/usr/local/lib/python3.10/dist-packages/accelerate/commands/accelerate_cli.py", line 48, in main
```

2. Problemas con Trainer de Hugging Face: Al implementar modelo con las herramientas de Diffusers, las mismas no están muchas veces diseñadas para entrenarse con el Trainer de Huggin Face. Se realizaron varios intentos con Trainer, sin rainer implementando la función, pero se tuvieron errores del estilo:

AttributeError: 'FrozenDict' object has no attribute 'to_json_string'

Por lo tanto como solución a ambos, se logró hacer el entrenamiento a través de la biblioteca accelerate para el finetuning usando el script train_text_to_image.py de diffusers.

El código detallado puede consultarse en la notebook vinculada en este informe.

Como resumen, se detalla la comparativa de los modelos explorados para llegar a la elección del modelo final en el siguiente cuadro:

MODELO	CARACTERÍSTICAS / VENTAJAS	PARÁMETROS	ARQ.	DATASET PREENTRENADO	DIFICULTADES o INCONVENIENTES PRESENTADOS
DiffusionCLIP	- Control fino sobre atributos de imágenes. Ideal para edición guiada por texto. Permite ajustar expresiones faciales y otros detalles. - Imagenes 256x256	110M (solo CLIP) + UNet	UNet + CLIP	CelebA-HQ (Human Face).	- Dificultad en implementación. - Checkpoint de Pytorch (incompatible con librerías de Hugging Faces) - Documentación limitada y sin ejemplos claros.
Versatile Diffusion	- Multimodalidad: texto a imagen, imagen a texto, y transformación de imágenes. - Flexibilidad y capacidad de generalización.	1.3B	Transformer-based	LAION-400M	- Problemas de compatibilidad de librerías y dependencias (versión de Diffusers por ejemplo) - Mayor complejidad para configuraciones. - Documentación limitada y sin ejemplos claros.
Stable Diffusion 2-Base	- Síntesis texto a imágenes, procesamiento eficiente - Resolución hasta 768×768	865M	Latent U-Net diffusion	LAION-5B	- No fue posible entrenar el modelo, en etapa de fine-tuning, utilizando la librería Trainer provista por Hugging Face y utilizada en algunos trabajos individuales de la materia.
Stable Diffusion 3	- Más alta fidelidad en generación de imágenes - Mejora en manejo de prompts - Resolución hasta 2048×2048	1.2B	Latent U-Net diffusion	LAION-5B	- La inferencia resultó en out-of-memory en las instancias gratuitas de Google Colab.

Desarrollo y decisiones de diseño

El notebook sobre el que se desarrolló el trabajo se encuentra en el siguiente link:

<https://drive.google.com/file/d/1OMaSafYw9n4427QBr-sU5RV0k4FVfgfO/view?usp=sharing>

A continuación se detallan algunas decisiones de diseño y desarrollo llevadas a cabo.

Entorno

Por los recursos con los que se cuentan, el proyecto se llevó a cabo en Google Colab, que como hemos comentado, tiene recursos limitados para ciertos procesos y entrenamientos.

Para el desarrollo del trabajo, se instalaron las librerías necesarias para el entrenamiento e inferencia del modelo. Se optó por utilizar las librerías *accelerate* y *diffusers* de *HuggingFace*, debido a las siguientes ventajas clave:

- **Configuración Automática:** Detectan y optimizan el uso de GPU y memoria sin requerir ajustes manuales.
- **Distribución Eficiente de Datos:** Facilitan la gestión de batches, mejorando el rendimiento.
- **Entrenamiento Optimizado:** Reducen tiempos y costos al implementar optimizaciones automáticas.

Además, se clonaron los repositorios de *diffusers* y del dataset seleccionado para realizar el fine-tuning a través de líneas de comandos, maximizando la compatibilidad con el entorno de Colab y los objetivos del proyecto.

Dataset

El dataset seleccionado es [*sketch-scene*](#) que contiene bocetos de escenas a mano alzada. El dataset tiene las siguientes características:

- Se basa en FS-COCO.
- Contiene 10.000 imágenes aproximadamente.
- Resolución de las imágenes: 256x256.
- Se compone de pares [image, text].
- No se encuentra dividido en subsets de entrenamiento, test.

Se decidió utilizar el dataset de por 2 motivos:

1. Es un conjunto de datos liviano, lo que nos permitió manipularlo, sin mayores problemas computacionales, en instancias gratuitas de Google Colab.
2. Otra razón, un poco más subjetiva en este caso, es que es un conjunto de datos fuera de lo común. La idea nuestra fue entrenar imágenes que no se asemejan a las que

están presente en datasets tradicionales como COCO. Esto permite una inspección empírica de los resultados, mucho más clara.

Modelo

El modelo elegido fue Stable Diffusion versión 2. Dentro de este modelo, hay dos versiones la base y la final, que se diferencian en que la primera tiene menos épocas de entrenamiento.

Se realizaron inferencias con *StableDiffusionPipeline* que permite ejecutar y cambiar de modelo de manera muy simple. A modo de ejemplo, se muestra el resultado de ambos modelos frente al prompt "*A sketch of a zebra and a monkey in front of a mansion*"



Imágenes generadas por Stable diffusion v2 base.



Imágenes generadas por Stable diffusion v2 final.

Si bien los resultados son parecidos, como la versión base tiene menos épocas de entrenamiento consideramos es más apropiada para mostrar un cambio de estilo con el fine tuning y fue entonces, el modelo elegido.

Fine tuning

El objetivo del fine tuning es obtener un modelo que tenga mejor desempeño en una tarea específica. En el caso de este trabajo se buscó generar imágenes de dibujos en blanco y negro, con trazos bruscos, como si fueran bocetos.

Como se mencionó anteriormente, se realizó a través del paquete *accelerate* de HuggingFace, con los siguientes parámetros:

- **use_8bit_adam**: Reduce la memoria consumida por el optimizador usando una versión con 8 bits de precisión.
- **gradient_checkpointing**: Reduce el uso de memoria durante entrenamiento forzando a recomputar activaciones intermedias solo cuando es necesario. Es más lento pero consume menos VRAM.

- **mixed_precision="fp16"**: Utiliza mitad de la precisión que se usa normalmente (float 16 en vez de float32). Así se reduce tanto memoria como tiempo de cómputo.
- **use_ema**: Usa una media móvil exponencial (EMA) para los pesos del modelo, mejorando la generalización y estabilidad durante el entrenamiento.
- **max_grad_norm=1**: Hace clipping del gradiente para prevenir gradientes explosivos y mejorar estabilidad durante el training.

```

11/28/2024 00:50:31 - INFO - __main__ - ***** Running training *****
11/28/2024 00:50:31 - INFO - __main__ - Num examples = 9999
11/28/2024 00:50:31 - INFO - __main__ - Num Epochs = 1
11/28/2024 00:50:31 - INFO - __main__ - Instantaneous batch size per device = 4
11/28/2024 00:50:31 - INFO - __main__ - Total train batch size (w. parallel, distributed & accumulation) = 16
11/28/2024 00:50:31 - INFO - __main__ - Gradient Accumulation steps = 4
11/28/2024 00:50:31 - INFO - __main__ - Total optimization steps = 30
Steps: 100% 30/30 [01:15<00:00, 2.43s/it, lr=3e-5, step_loss=0.263]11/28/2024 00:51:52 - INFO - __main__ - Running validation...
  
```

Accelerate se encarga automáticamente de varias tareas, como determinar el batch size máximo para el dispositivo disponible.

Data augmentation y transformaciones

- **Resolución**: se utiliza una resolución de 256x256 para menor consumo de memoria.
- **center_crop**: Se recorta la imagen preservando el centro.
- **random_flip**: Se hace un random horizontal flip con probabilidad del 50%.

Función de pérdida

Para almacenar las métricas se utiliza la plataforma de *Weights and Biases* (wandb). En cuanto a la función de pérdida, el script que se invoca con accelerate utiliza *denoising loss*, una forma de MSE que toma en cuenta las variables durante el entrenamiento de un modelo *diffusion*:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0, \epsilon, t} [\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2]$$

Donde:

L: Pérdida de denoising.

X0: Muestra original de datos (por ejemplo, una imagen).

ε: Ruido aleatorio añadido a los datos.

t: paso temporal en cada proceso de difusión.

x_t : La versión ruidosa de x0 en el paso t.

ϵ_θ : Ruido predicho por el modelo.

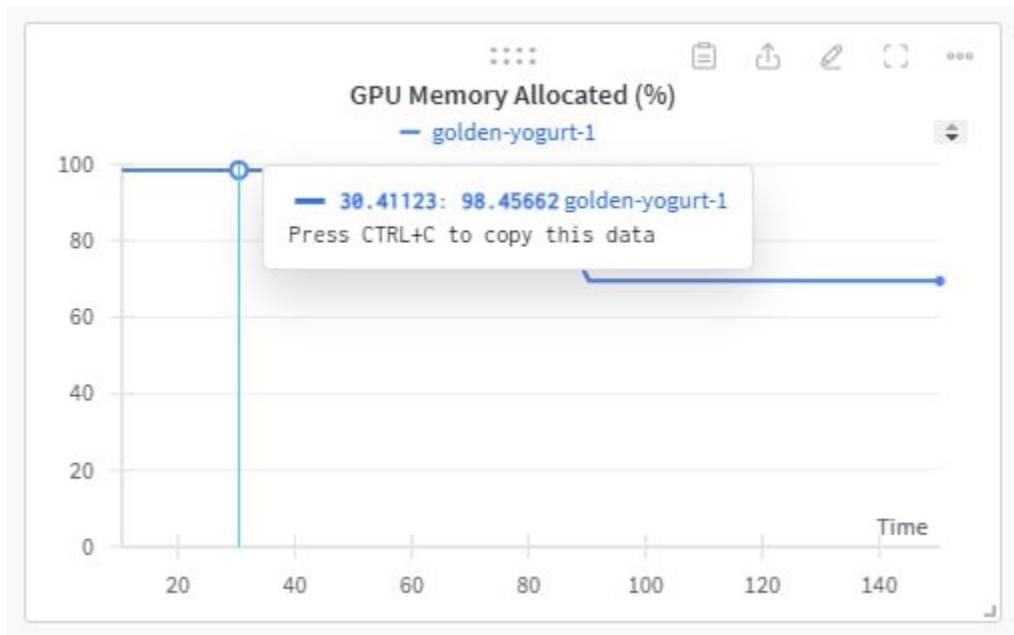
$\|\cdot\|$: Error cuadrático medio (MSE).

Se utilizaron 30 épocas para el entrenamiento y la pérdida fue la siguiente:



Pérdida por época durante el entrenamiento.

Además, se guardaron varias métricas de uso del sistema para poder garantizar que el modelo entra dentro de la VRAM disponible.



Uso de memoria durante el entrenamiento. Wandb permite registrar todas estas métricas fácilmente.

Para futuro, se pueden considerar funciones de pérdida alternativas o combinadas a denoising loss. Por ejemplo, usar CLIP como una función de pérdida a través de la medida de la similitud coseno entre los *embeddings* de las imágenes generadas y el prompt textual. En ese caso podría usarse la similitud coseno negativa entre los *embeddings* generados por la imagen y los *embeddings* generados por el prompt, de manera que el objetivo del modelo sea maximizar la similitud (minimizando la pérdida).

Inferencia utilizando el modelo fine-tuned

Utilizando nuevamente *StableDiffusionPipeline* se volvió a realizar inferencia pero ahora con el modelo fine-tuned. Se creó otro pipeline para poder comparar ambas versiones (antes y luego del entrenamiento) durante la etapa de evaluación.



Resultados del cambio de estilo luego de la inferencia.

Evaluación

Para evaluar los resultados, se utilizó CLIP, un modelo que genera embeddings de texto e imágenes en un espacio compartido. Esto permite comparar directamente la similitud entre un prompt textual y las imágenes generadas.

El modelo CLIP usado es el siguiente: <https://huggingface.co/openai/clip-vit-base-patch32>. Está preentrenado con diversos datasets como YFCC100M y otros set image-caption que se encontraba disponible públicamente en internet.

Métrica de Evaluación

El proceso de evaluación consistió en el siguiente procedimiento:

1. Prompts Seleccionados: Se eligió un conjunto de prompts representativos.
2. Imágenes Generadas: Por cada prompt, se generaron seis imágenes.
3. Cálculo de Similitud:
 - Por Prompt: Se calculó el promedio de las similitudes coseno de las seis imágenes generadas.
 - Global: Se promedió el resultado de todos los prompts para obtener la métrica final del experimento.

Experimentos Realizados

Se realizaron dos variantes principales para evaluar el efecto de agregar un estilo específico en el prompt:

1. Agregando una Preposición de Estilo:

- Texto agregado: "*A black and white sketch of...*".
- Esta preposición se incluyó tanto en el prompt para generar la imagen como en el prompt ingresado a CLIP.
- Ejemplo:
 - i. Prompt: "*A zebra running*".
 - ii. Entrada al modelo: "*A black and white sketch of a zebra running*".
 - iii. Entrada a CLIP: "*A black and white sketch of a zebra running*".

2. Sin Preposición en el Generador:

- Se eliminó la preposición en el generador de imágenes, pero se mantuvo en el prompt ingresado a CLIP.
- Ejemplo:
 - i. Prompt: "*A zebra running*".
 - ii. Entrada al modelo: "*A zebra running*".
 - iii. Entrada a CLIP: "*A black and white sketch of a zebra running*".

Estos experimentos buscaban identificar si el modelo de difusión respondía correctamente a cambios estilísticos requeridos en los prompts.

Confirmación Adicional con BLIP

Además, se utilizó BLIP, un modelo capaz de generar texto a partir de imágenes, que permitió validar si el modelo reconocía y captaba adecuadamente el estilo deseado en las imágenes generadas.

El modelo BLIP usado es el siguiente (preentrenado con COCO dataset):

<https://huggingface.co/Salesforce/blip-image-captioning-base>.

Análisis de los resultados

Evaluación usando CLIP

A continuación se muestran las imágenes generadas para los siguientes prompts. Cabe destacar que idealmente se debería tomar un porcentaje representativo de las etiquetas del dataset utilizado y obtener un promedio de la evaluación por similitud coseno. Esto no pudo ser posible debido a limitaciones de tiempo de GPU disponible en Colab.

Prompts:

- "a boy is practicing martial arts"
- "a statue of a man on horse"
- "a boy is riding bike on the road"
- "giraffe standing in a forest"
- "zebra running"

Resultados - Experimento 1:



Prompt: “A black and white sketch of a boy is practicing martial arts”.



Prompt: “A black and white sketch of a statue of a man on horse”.



Prompt: “A black and white sketch of a boy riding a bike on the road”.



Prompt: “A black and white sketch of giraffe standing in a forest”.



Prompt: “A black and white sketch of zebra running”.

Resultados - Experimento 2 (sin agregado de preposición para la generación de imágenes).



Prompt: “A black and white sketch of a boy practicing martial arts”.



Prompt: “A black and white sketch of a statue of a man on horse”.



Prompt: “A black and white sketch of a boy riding a bike on the road”.



Prompt: “A black and white sketch of a giraffe standing in a forest”.



Prompt: “A black and white sketch of zebra running”.

En cuanto a la evaluación de similitud coseno de embeddings con CLIP se obtuvieron las siguientes métricas para ambos experimentos:

Experimento	Similitud coseno promedio
1. Utilizando pre-prompt de estilo en generador y CLIP	36.48%
2. Utilizando pre-prompt de estilo sólo en CLIP	33.58%

Aquí se puede ver que no sólo es notorio el cambio de estilo que induce agregar el texto “A black and white sketch of...” antes del prompt, sino que además es posible obtener un número de afinidad al prompt deseado utilizando CLIP.

Evaluación usando BLIP

BLIP (Bootstrapped Language-Image Pretraining) permite generar directamente un texto a partir de una imagen. Teniendo en cuenta los prompts utilizados anteriormente, se generó un texto por cada una de las imágenes del *experimento 1*.

La idea fue poder validar si el pie de imagen generado por BLIP coincide con el prompt indicado en la generación de imágenes.

Prompt	Texto generado por BLIP	Imagen
A black and white sketch of a boy practicing martial arts	A drawing of two people in karate stance	
A black and white sketch of a <i>statue</i> of a man on horse	A drawing of a horse and rider	
A black and white sketch of a boy riding a bike on the road	A drawing of a man riding a bike	

A black and white sketch of a giraffe standing in a forest	A drawing of a gi in the wild	
A black and white sketch of a zebra running	A a zebra running in the ai	

Puede observarse que en todos los textos BLIP retorna “A drawing of...” por lo que indica que capta el estilo que se busca en la generación de imágenes.

Conclusiones

- Entrenar un modelo de generación de imágenes a texto es una tarea que consume muchos recursos computacionales, especialmente VRAM.
- A través de optimizaciones como el uso de *half precisión* (fp16) y Adam en 8 bits, fue posible reducir drásticamente el consumo de memoria.
- El modelo Stable Diffusion Base pre-entrenado posee un buen desempeño pero el fine tuning muestra que es posible refinar el modelo para que se adapte a una tarea específica.
- Se pueden mejorar o profundizar sobre CLIP como función de pérdida alternativa o combinada a denoising loss.
- Se logró un cambio de estilo pre y pos fine tuning, perceptible a simple vista.
- A través del uso de modelos como CLIP fue posible obtener una métrica objetiva de qué tan bien se desempeña el modelo en la tarea requerida.
- Además, el uso de BLIP ayudó a confirmar que es posible captar el cambio de estilo a través de un modelo de IA.
- A partir de librerías de HuggingFace pudo realizarse inferencia y entrenamiento de modelos basados en difusión de manera sencilla.