NOMBRE: Benjamín Farías Valdés

N.ALUMNO: 22102671

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3692 — Tópicos Avanzados en Inteligencia Artificial — 2' 2022

Informe de Avance

Introducción

En los años recientes, la aparición de modelos con grandes cantidades de parámetros (como *GPT-3*, que tiene varios billones) ha traído consigo tanto ventajas como desventajas al momento de aplicarlos en la práctica. Las ventajas apuntan a un gran rendimiento en tareas de generalización sobre los datos que aprenden, evidenciando el potencial de tener arquitecturas a gran escala. Por otro lado, las principales limitaciones aparecen respecto a la gran cantidad de información, memoria y tiempo de procesamiento que estos modelos demandan al momento de ser entrenados. En este contexto surge la técnica conocida como *prompting*, que será el enfoque de este proyecto.

Prompting

Esta técnica consiste en condicionar un modelo pre-entrenado para que aprenda a realizar una nueva tarea o actuar sobre un nuevo dominio, tan sólo mostrándole ejemplos sobre esta tarea como parte de la entrada y salida del modelo. Es decir, existe un cambio de enfoque: en vez de que el modelo se ajuste a los datos (entrenamiento y *fine-tuning*), ahora ajustamos los datos al modelo generalizador. Dada la naturaleza de este enfoque, se puede clasificar como una forma de *few-shot learning*, ya que mediante unos pocos ejemplos, se pueden extender las funcionalidades de un modelo pre-entrenado. En cuanto a los *approaches* existentes, se tiene que para modelos no tan grandes (*BERT*, *RoBERTa*), se busca aplicar *fine-tuning* basado en *prompts*. Para modelos enormes (*GPT-3*, *T5*), dado el gran costo de ajustar sus parámetros internos, el enfoque está en congelarlos y añadir una cantidad pequeña de parámetros en la entrada y/o salida, los que serán utilizados para modular el modelo pre-entrenado.

Propuesta

Para este proyecto, se utilizará el modelo pre-entrenado *Stable Diffusion v1.5*, que se utiliza para generar imágenes pseudo-realistas a partir de una instrucción entregada en texto plano. Este modelo cumple con ciertas características de interés:

- Está compuesto por 3 modelos internos: un *encoder* de texto sacado de *CLIP*, un *Variational Auto-encoder* capaz de mapear imágenes a un espacio latente de menor dimensionalidad, y una red *U-Net* que aprende a filtrar ruido visual de forma iterativa hasta que se obtenga una representación visual cercana a la representación del texto entregado.

- La suma total de los parámetros de estos 3 modelos es superior a 1 billón, por lo que es una arquitectura de gran tamaño.
- Debido a su gran tamaño, es esperable que el modelo pre-entrenado contenga conocimiento amplio sobre representaciones latentes de texto e imágenes.

La arquitectura de *Stable Diffusion* se puede apreciar en la siguiente imagen:

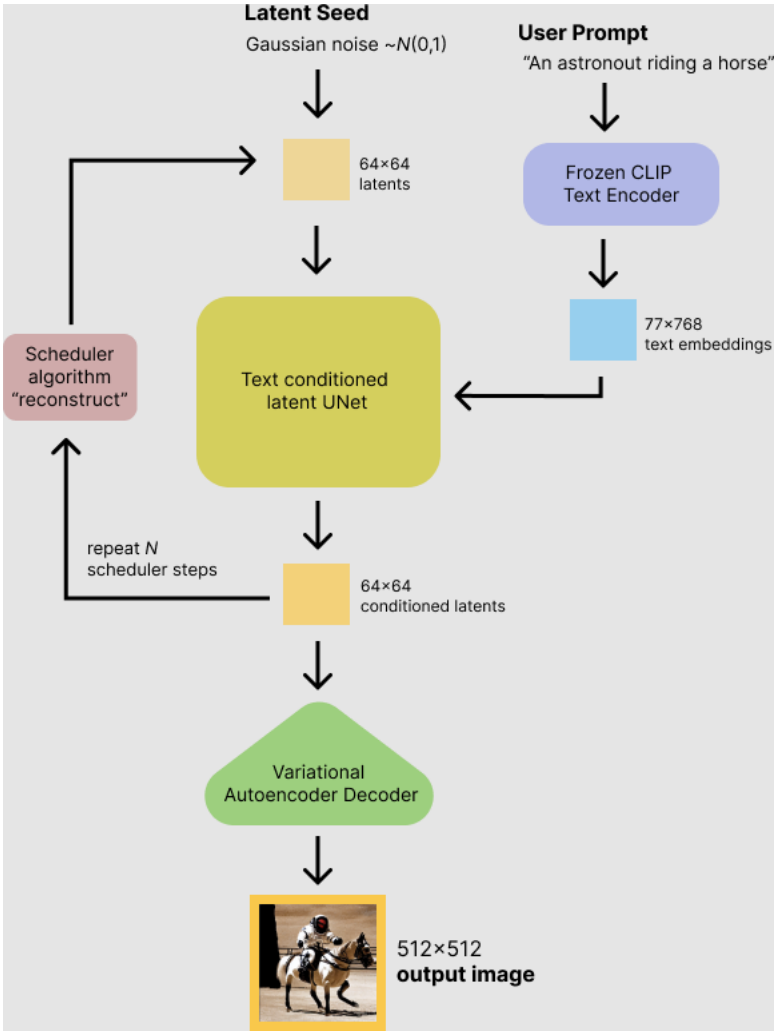

Figura 1: Arquitectura de *Stable Diffusion*.

Gracias a estas características, se espera que este modelo sea capaz de adaptarse a cambios de dominio (generar imágenes de un cierto estilo o temática), o bien, a tareas nuevas (generar imágenes que no necesariamente sean lo mismo que dice el texto de entrada, pero relacionadas de alguna forma).

Dicho esto, la **hipótesis de trabajo** es la siguiente: El modelo *Stable Diffusion* puede adaptarse a cambios de dominio y/o a tareas nuevas mediante la utilización y aprendizaje de *prompts*, manteniendo congelados los parámetros de los modelos internos pre-entrenados.

Avances (Investigación)

Lo primero que se realizó fue una fase de investigación sobre las propuestas actuales de interés. En particular, es importante mencionar que actualmente existen múltiples algoritmos de entrenamiento que intentan realizar la adaptación del modelo a distintos dominios (véase *DreamBooth*, *Textual Inversion*). Estos algoritmos ya tienen implementaciones propias y logran buenos resultados, pero no son tan rápidos de entrenar, debido a que realizan *fine-tuning* de los modelos pre-entrenados. Además, realizar estos entrenamientos consume bastante memoria de la tarjeta gráfica. Por otro lado, no se encontraron *approaches* que sólo utilicen *prompting* (congelando los modelos pre-entrenados).

Debido a esto, el objetivo de este proyecto se definió en base a diseñar y probar distintas extensiones del modelo *Stable Diffusion*, las que corresponderán a añadir ciertos componentes nuevos que permitan modular la red pre-entrenada. La idea es que al entrenar estas extensiones del modelo, estos componentes aprendan a condicionar a la red pre-entrenada para lograr experimentar con la hipótesis de trabajo, sin tener que modificar los componentes internos pesados.

La ventaja de este *approach* respecto a los existentes es su eficiencia, ya que entrenar componentes ligeros y mantener lo demás congelado significa una disminución de varios órdenes de magnitud respecto a la cantidad de optimizaciones de parámetros. Por otro lado, la principal desventaja que puede surgir durante el trabajo, es que estos componentes no sean capaces de lograr satisfactoriamente sus objetivos, en el caso de que las tareas sean demasiado complejas o dependientes de los parámetros pre-entrenados, y por lo tanto no sean manejables desde componentes externos ligeros.

Durante el desarrollo del proyecto, se espera probar de forma experimental con distintas extensiones posibles del modelo base, según lo indicado previamente. Con los resultados de estos experimentos, se podrá probar si la hipótesis de trabajo era correcta o errónea, concluyendo así el objetivo del proyecto.

Avances (Implementación)

Utilizando el *framework* de *PyTorch* y la implementación de *Stable Diffusion* presente en la librería de *Hugging Face*, se configuró inicialmente un *pipeline* de inferencia capaz de generar imágenes a partir de texto, utilizando el modelo base. A continuación se muestra una imagen generada:

Figura 2: Imagen generada por el modelo base, dado el texto **'the sun goes supernova'**.

El segundo avance correspondió a adaptar el código para poder hacer la inferencia utilizando los componentes por separado (en vez del *pipeline* que automatiza todo el proceso). Para lograr esto, se tuvo que manipular bastante el código y generar funciones encargadas de realizar paso a paso el algoritmo mediante los modelos pre-entrenados.

Teniendo la inferencia lista, se procedió a programar el *script* de entrenamiento, el que demandó una gran cantidad de trabajo debido a la poca documentación al respecto. Una vez armado este algoritmo, se colocó, como prueba inicial, un par de capas lineales entre los *embeddings* del texto y la entrada a la red *U-Net* (ver figura 1). La idea de esto es que estas capas lineales aprendan a condicionar a la red mediante manipulación de los *embeddings* originales del texto. Con los pesos iniciales de estas capas lineales (aleatorios), se obtuvo la siguiente imagen:

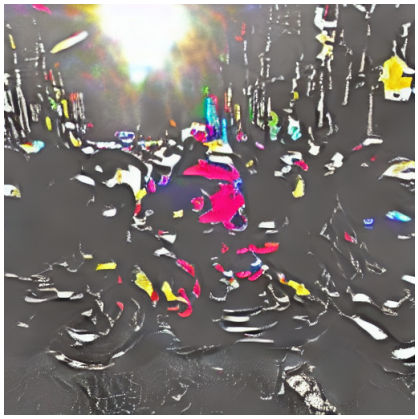

Figura 3: Imagen generada por la extensión al modelo (sin entrenar), dado el texto **'a tennis ball'**.

Como se puede observar, el añadir el nuevo componente a la red resulta en una imagen ruidosa y con poco sentido, lo que se debe a que los pesos de esta red manipulan erróneamente las representaciones latentes del texto (confundiendo al modelo). Tras entrenar por 10 épocas y con un *dataset* pequeño de imágenes pixeladas (buscando que la red aprenda a generar imágenes en ese dominio), se obtiene lo siguiente:

Figura 4: Imagen generada por la extensión al modelo (entrenada ligeramente), dado el texto **'a tennis ball'**.

Al comparar con la imagen anterior, se notan diferencias muy ligeras, y sigue siendo incorrecta la representación. Esto se puede explicar debido a que el entrenamiento no fue muy extensivo (poco tiempo, pocos ejemplos), y además me parece que el colocar estas capas lineales en esa zona de la arquitectura puede haber sido mala idea.

Trabajo Futuro

Dados estos resultados, estas son los siguientes pasos para el desarrollo del proyecto:

- Probar con *datasets* más grandes y personalizar mejor el entrenamiento, además de realizarlo por más tiempo.
- Probar con nuevas propuestas de diseño para los componentes moduladores, quizá empezar por colocar las *prompts* aprendibles antes de la entrada al *text encoder*. Experimentar y quedarse con las que mejor rendimiento muestren.
- Comparar con métricas cuantitativas, respecto a el modelo base (y quizás alguno que haya aplicado *fine-tuning*).
- Experimentar con distintos dominios y/o tareas, de forma que se pueda probar con mayor seguridad si la hipótesis de trabajo finalmente era correcta o no.

Bibliografía

[**1**] Brown, T. et al. (2020). *Language Models are Few-Shot Learners.*

[**2**] Lester, B., Al-Rfou, R., Constant, N. (2021). *The Power of Scale for Parameter-Efficient Prompt Tuning.*

[**3**] Radford, A. et al. (2021). *Learning Transferable Visual Models From Natural Language Supervision.*

[**4**] Wang, Z. et al. (2021). *Learning to Prompt for Continual Learning.*

[**5**] Rombach, R., et al. (2021). *High-Resolution Image Synthesis with Latent Diffusion Models.*