

Vision Transformers

Docentes:

Esp. Abraham Rodriguez - FIUBA

Mg. Oksana Bokhonok - FIUBA

Programa de la materia

1. Arquitectura de Transformers e imágenes como secuencias
2. Arquitecturas de ViT y el mecanismo de Attention.
3. Ecosistema actual, Huggingface y modelos pre entrenados.
4. [GPT en NLP e ImageGPT](#).
5. Modelos multimodales: combinación de visión y lenguaje
6. Segmentación con SAM y herramientas de auto etiquetado multimodales.
7. OCR y detección con modelos multimodales.
8. Presentación de proyectos.

GPT en NLP

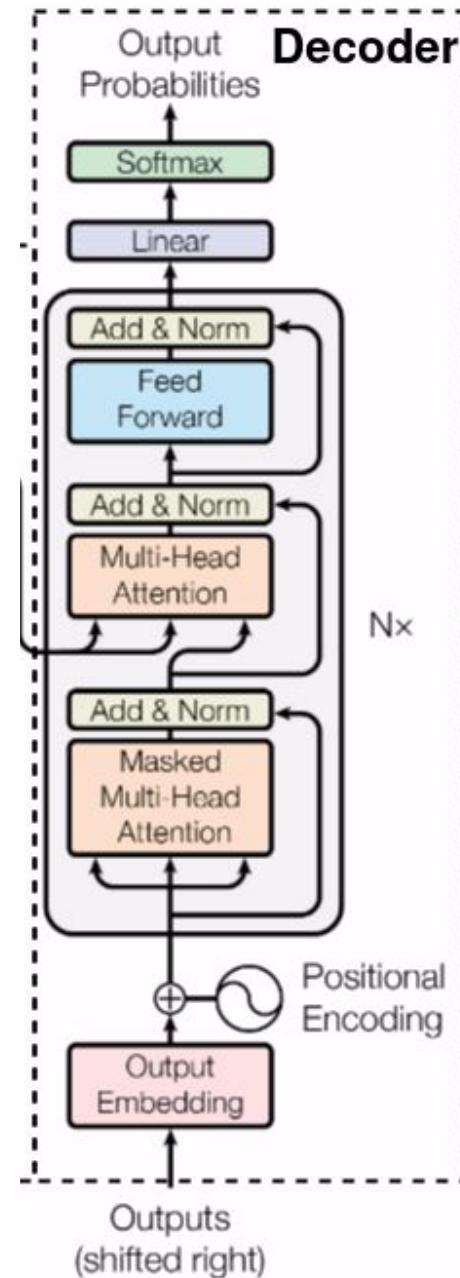
Transformers Decoders

Las LLMs y modelos generativos basados en Transformers son normalmente Transformer Decoders. Las LLMs son realmente una arquitectura de Decoder llamada **Generative Pretrained Transformer (GPT)**.

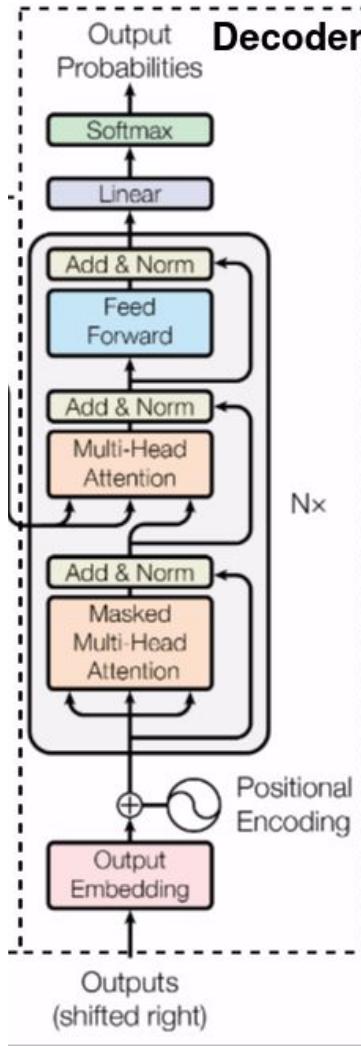
El Decoder se encarga de decodificar un espacio latente, en el caso de la arquitectura completa, es el espacio generado por el encoder.

El espacio latente es un espacio de información que consta de las características de los datos como la relación entre palabras.

El espacio latente es utilizado por el decoder para “generar” información a vectores de entrada, por ejemplo completar secuencias de texto.



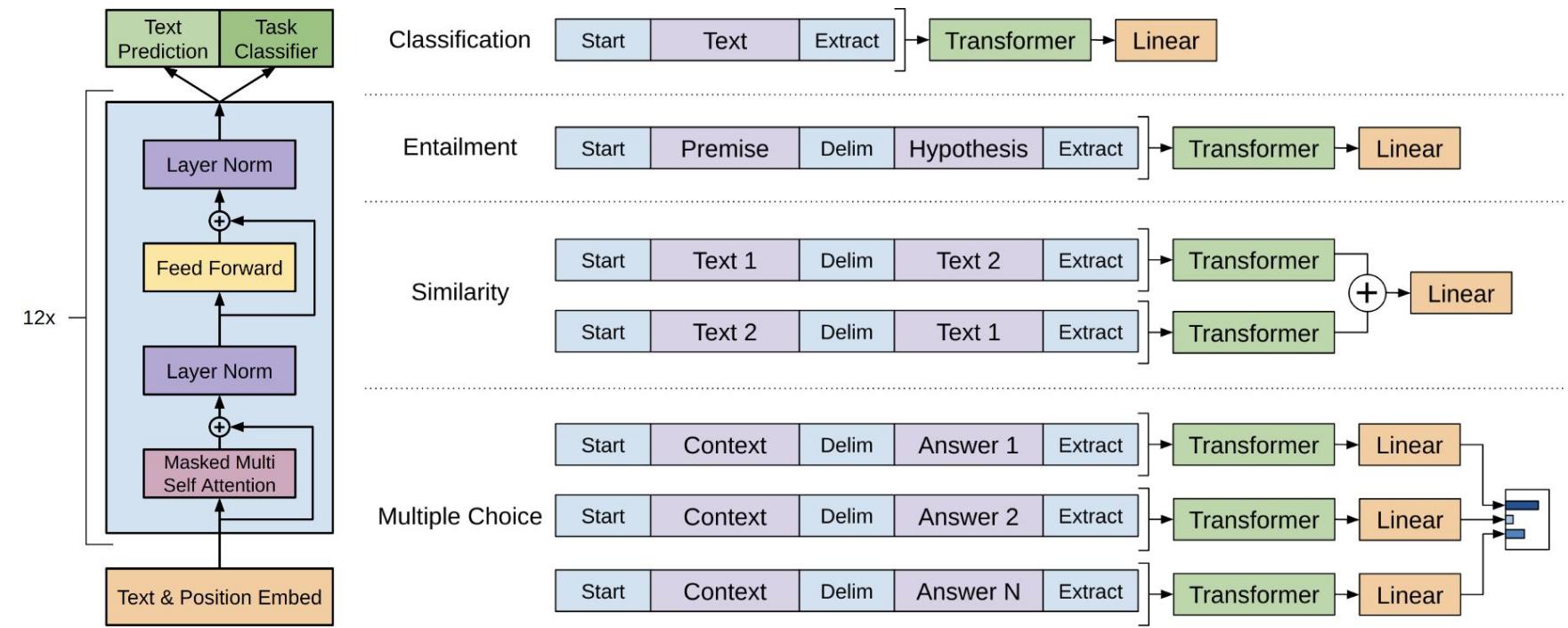
Generative Pretrained Transformer (GPT)



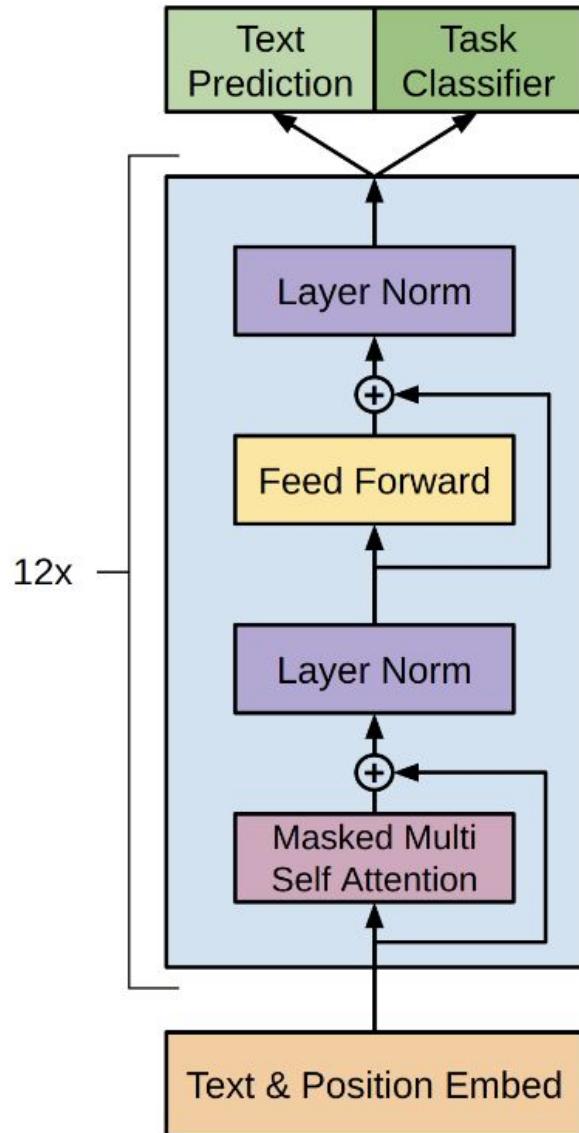
GPT es la arquitectura que trajo como consecuencia modelos del estado del arte como chatGPT (closed-source), Llama, Mistral, Gemma (open-source). Así como modelos **multimodales** como CLIP y modelos **generativos** de imágenes como ImageGPT.

En 2018 OpenAI presentó el paper [Improving Language Understanding by Generative Pre-training](#). Donde se introdujo el concepto de Generative Pretraining, utilizando la arquitectura del **Transformer Decoder ligeramente modificado**.

El paper demuestra distintos objetivos de entrenamiento de GPT.



GPT Unsupervised Pre-training



Dado un corpus de tokens $U = \{u_1, \dots, u_n\}$, se utiliza un objetivo de modelado estándar de lenguaje para maximizar la verosimilitud:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

donde k es el tamaño de la ventana de contexto, la probabilidad condicional P es modelada usando una red neuronal con parámetros Θ entrenables mediante SGD.

La red neuronal referida es el **Transformer Decoder**, el cual da como resultado una distribución de probabilidad de tokens de salida.

$$h_0 = UW_e + W_p$$

$$h_l = \text{transformer_block}(h_{l-1}) \quad \forall i \in [1, n]$$

$$P(u) = \text{softmax}(h_n W_e^T)$$

$U = (u_{-k}, \dots, u_{-1})$ vector de contexto de tokens

W_e matriz de token embeddings

W_p position embeddings

n # de capas

GPT Unsupervised Pre-training

PRE ENTRENAMIENTO	
Característica	Detalles
Arquitectura	12 capas de transformer decoder.
Dimensiones	768 embeddings 12 attention heads 3072 MLP.
Entrenamiento	Optimizador Adam LR Max: 2.5e-4 100 epochs
BatchSize	64 minibatch de 512 tokens.
Dropout	0.1 (embeddings)
Regularización	L2 (weight 0.01) Dropout (p = 0.1)
Funcion de Activacion	GELU
Tokenizer	BPE con 40k tokens (merges).
Corpus	BooksCorpus

El objetivo del preentrenamiento es simplemente predecir la siguiente palabra de una secuencia, con la meta de aprender y estructurar patrones de lenguaje natural.

Next Token Prediction

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

GPT Supervised Fine-Tuning (SFT)

Luego de entrenar al modelo con el objetivo de maximizar la verosimilitud, se asume que se cuenta con un dataset de secuencias de tokens $X = \{x_1, \dots, x_m\}$ etiquetado con y . Las entradas son pasadas por un modelo preentrenado para obtener la última activación h_l^m del transformer block. Se agrega una capa lineal con parámetros W_y para predecir a y .

$$P(y|x^1, \dots, x^m) = \text{softmax}(h_l^m W_y)$$

Esto genera un nuevo objetivo, siendo maximizar la verosimilitud:

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \dots, x^m)$$

En otras palabras consta de maximizar la probabilidad de la etiqueta con respecto a las secuencias.

Adicionalmente para acelerar y mejorar la generalización y convergencia, se define el lagrangiano que es maximizar la verosimilitud sobre el dataset dado un peso λ .

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

GPT Supervised Fine-Tuning (SFT)

Understanding and Using Supervised Fine-Tuning (SFT) for Language Models

FINE TUNING	
Característica	Detalles
<i>Entrenamiento</i>	Optimizador Adam LR Max: 6.25e-5 3 epochs
<i>BatchSize</i>	32 minibatch de 512 tokens.
<i>Dropout</i>	0.1 (embeddings)
<i>Regularización</i>	Dropout ($p = 0.1$)

Generative Pretrained Transformer (GPT-2)

GPT-2 es la arquitectura que definió a **Chat LLMs** y a los modelos generativos actuales.

En 2019 OpenAI presento el paper [Language Models are Unsupervised Multitask Learners](#). Donde se reutilizo la arquitectura de GPT muy ligeramente modificada. Básicamente es mas grande y robusta, el cambio más significativo es un nuevo dataset diverso en contenido.

Gracias a su capacidad aumentada es capaz de hacer Zero-shot lo que significa que es capaz de hacer tareas las cuales no fue entrenado, dando la cualidad de generalizar tareas.

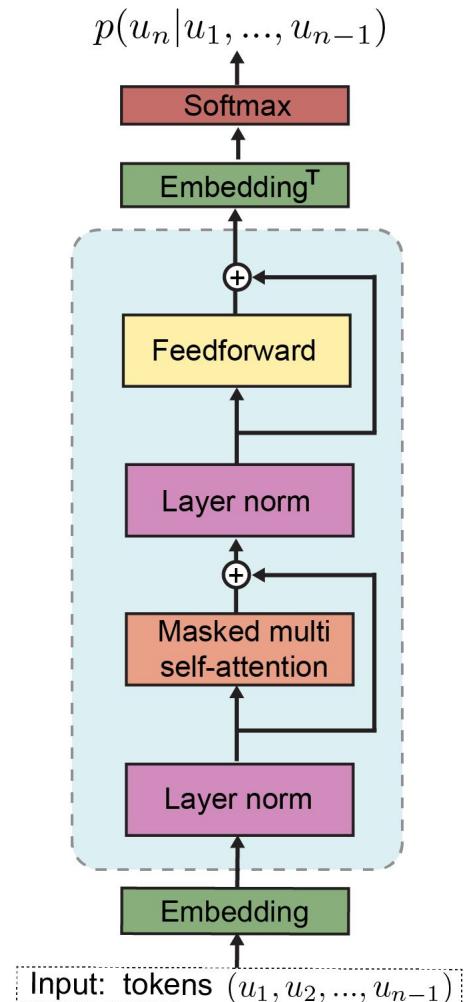
Característica	GPT-1	GPT-2
Parámetros	110 millones	hasta 1.5 billones
Hidden Dimensions	768	1024 (modelos grandes)
Attention Heads	12	16
Training Dataset	BooksCorpus (5GB)	WebText (45GB)
Text Length	Max 512 tokens	Max 1024 tokens
Zero-shot Learning	No	Si
Text Generation	Decente y simple	Coherente y largo

GPT-2 Marcó la necesidad de datasets de alta calidad, más grandes, así como modelos más robustos +7B, 21B, 90B...

The Illustrated GPT-2

[Guia explicativa](#)

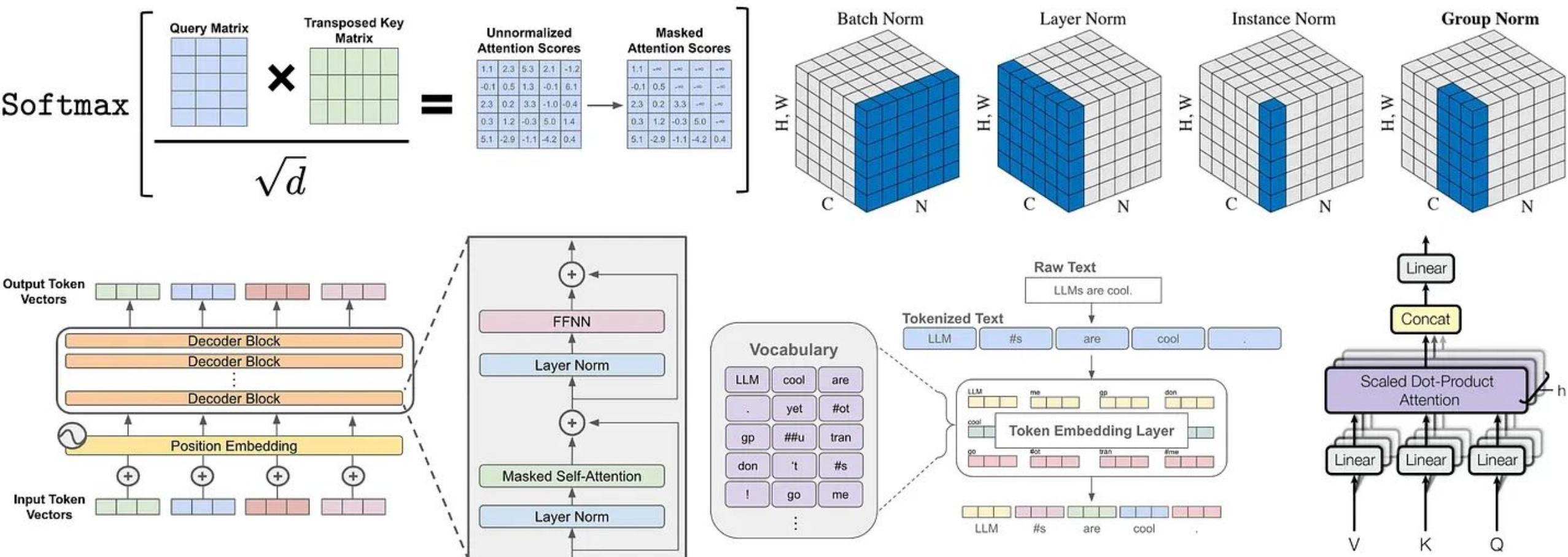
Recomendable leer!



Transformers Decoders

Decoder-Only Transformers: The Workhorse of Generative LLMs.

LLM Visualization



Generative Pretrained Transformer (GPT-3)

En 2020 OpenAI presentó el paper [Language Models are Few-Shot Learners](#). Donde se reutilizo la arquitectura de GPT-2 modificada. Utiliza un nuevo mecanismo de atención optimizado similar a [Sparse Transformer](#). Entrenaron 8 versiones del modelo.

GPT-3 “davinci” trato de comprobar la hipótesis de que escalar un modelo mejora el rendimiento de múltiples tareas sin necesidad de realizar fine-tuning, buscando que el modelo sea capaz de generalizar solamente entrenando con un dataset diverso

El entrenamiento de GPT-3 fue computacional y energéticamente costoso, consumió miles de petaflops/día durante preentrenamiento.

GPT-3 no utilizo **fine-tuning**, en su lugar utilizó **Few-shot, One-shot, Zero-shot**.

[Resumen del paper](#)

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate	Dataset	Quantity (tokens)	Weight in training mix
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}	Common Crawl (filtered)	410 billion	60%
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}			
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}			
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}			
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}			
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}			
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}			
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}			

Generative Pretrained Transformer (GPT-3)

Segun [LambdaLabs](#) GPT-3 tuvo un costo aproximado de \$4.6M.

Recomendable leer!

GPT-3 Zero-shot y One-shot

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

- 1 Translate English to French: ← *task description*
- 2 cheese => ← *prompt*

Zero-shot implica que el modelo es capaz de realizar tareas sin demostraciones previas, gracias a las grandes cantidades de datos.

[Prompt Engineering Guide: Zero-shot Prompting](#)

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

- 1 Translate English to French: ← *task description*
- 2 sea otter => loutre de mer ← *example*
- 3 cheese => ← *prompt*

One-shot, implica que el modelo realiza una tarea con **un solo** ejemplo demostrativo, esto es útil cuando Zero-shot no es suficiente.

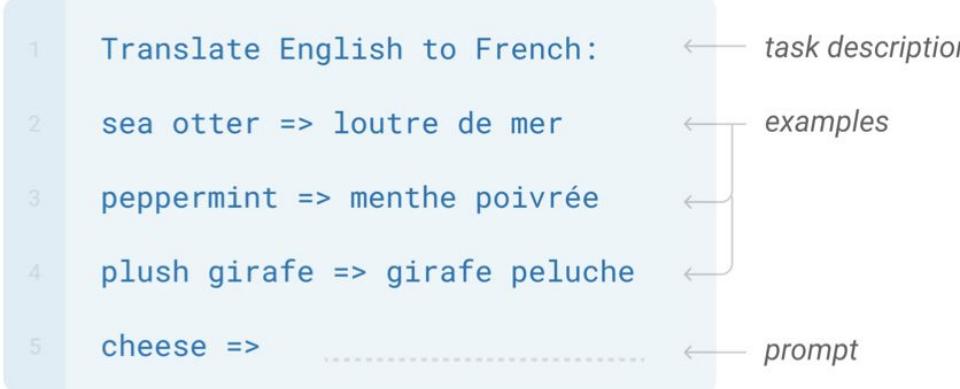
[Prompt Engineering Guide: One-shot Prompting](#)

GPT-3 Few-shot

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

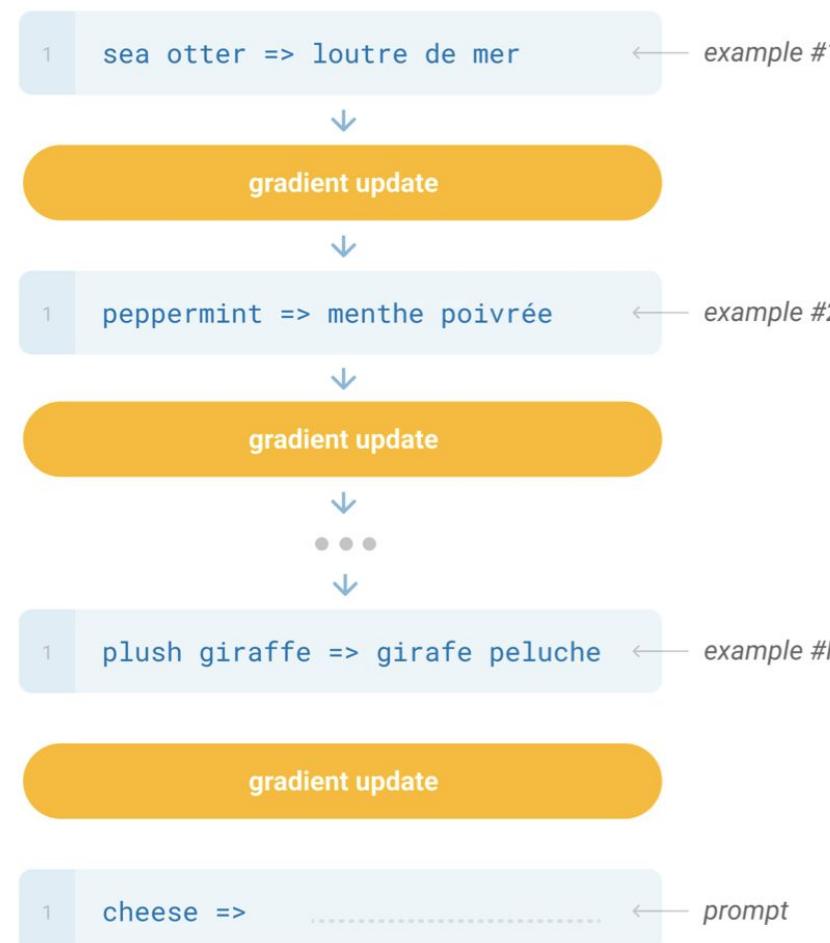
Few-shot, es dar múltiples ejemplos demostrativos, en otras palabras es una generalización de one-shot.



GPT-3 ¿Sin Fine-tuning?

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



Fine-tuning, implica actualizar gradientes del modelo, la idea de GPT-3 es evitar manipular los gradientes para lograr generalizar en lugar de realizar tareas específicas.

Para GPT-3, lo mas importante es la flexibilidad y eficiencia al tener un modelo capaz de hacer distintas tareas “on the fly” solamente utilizando **prompts**.

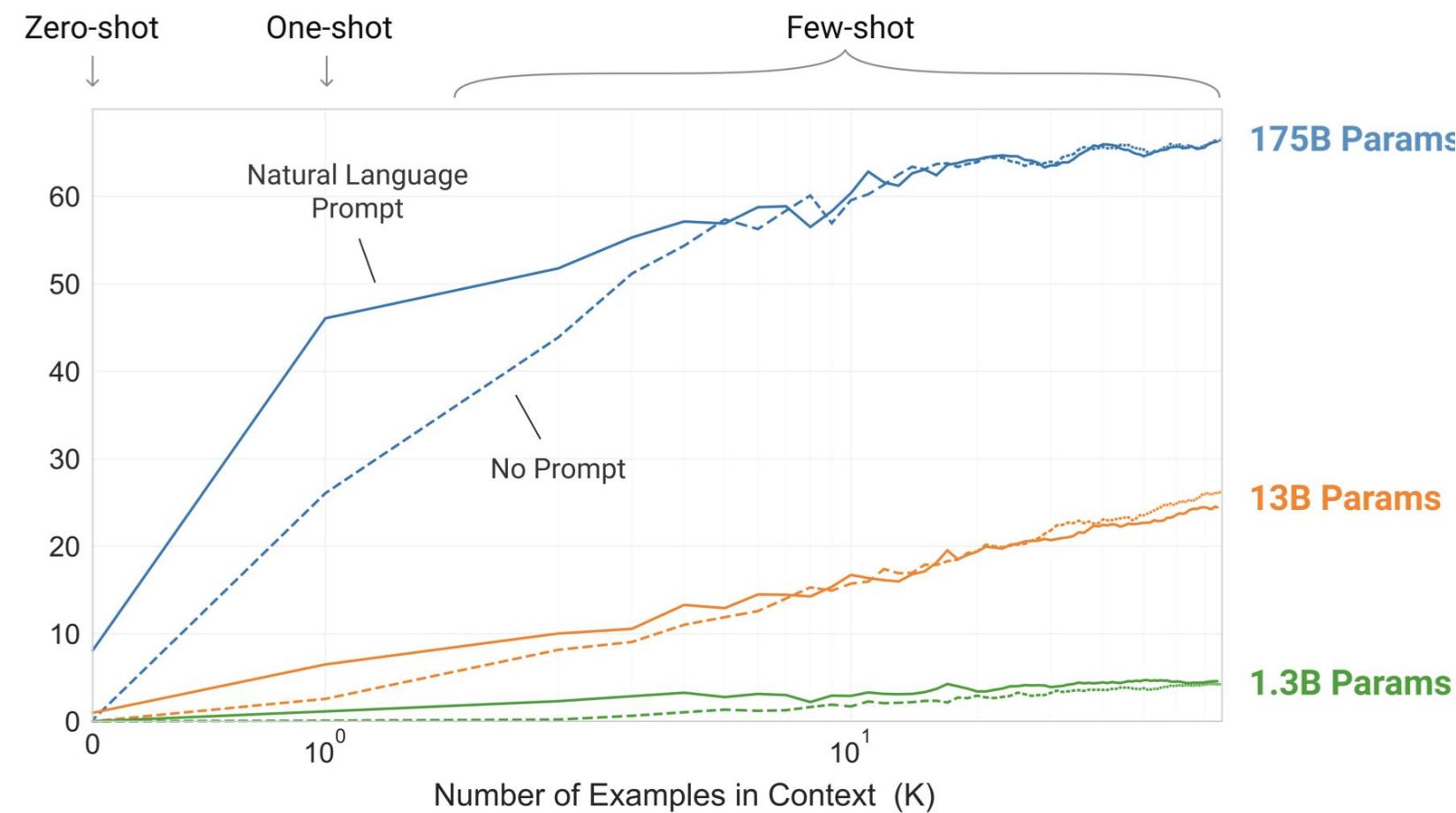
La razón por la cual GPT-3 es capaz de interpretar distintas tareas sin fine-tuning es debido a la diversidad de los datos. Por ejemplo, preguntas y respuestas en reddit, texto informativo de Wikipedia.

Los datasets utilizados cuentan con datos variados distintas tareas, idiomas, etc.

[GPT-3 Explanation \(Video\)](#)

La manera de aprender de manera generalizada es llamado “in-context learning”

GPT-3 ¿Sin Fine-tuning?



Los modelos más grandes utilizan de manera más eficiente información “in-context”

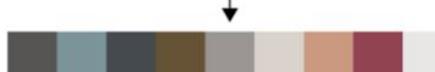
GPT en Visión Artificial

ImageGPT

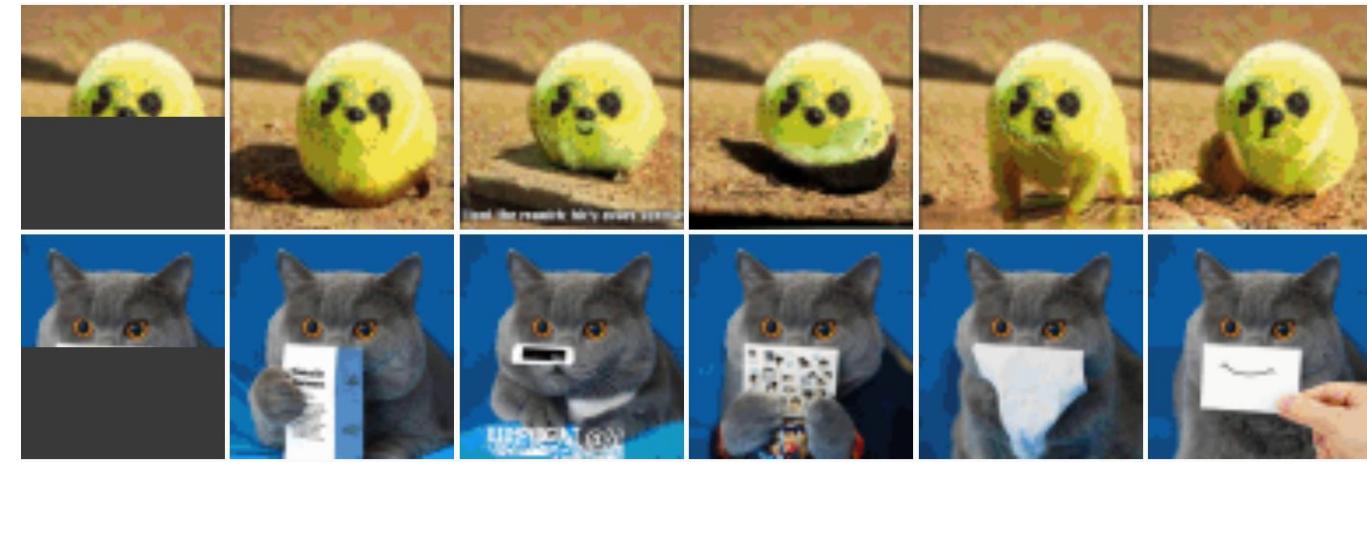
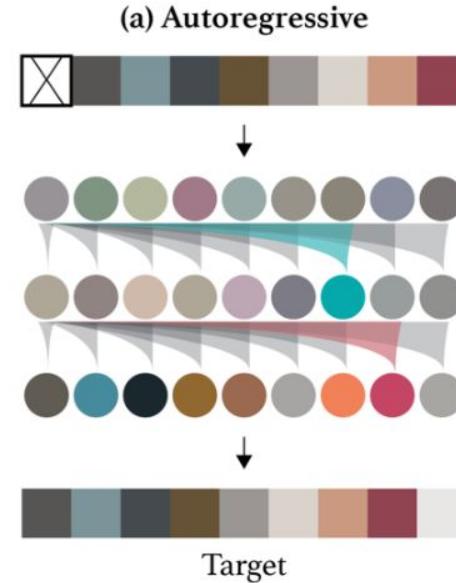
ImageGPT fue propuesto en el paper “[Generative Pretraining from pixels](#)” (2020) y consiste en un GPT-2 (Transformer Decoder) para la tarea de predicción de píxeles, de manera similar a NLP con la predicción de tokens.

[Huggingface](#)
[Website](#)

I



2



ImageGPT Pre-training

Dado un conjunto de datos no etiquetado X que consiste en datos de alta dimensión $x=(x_1, \dots, x_n)$.

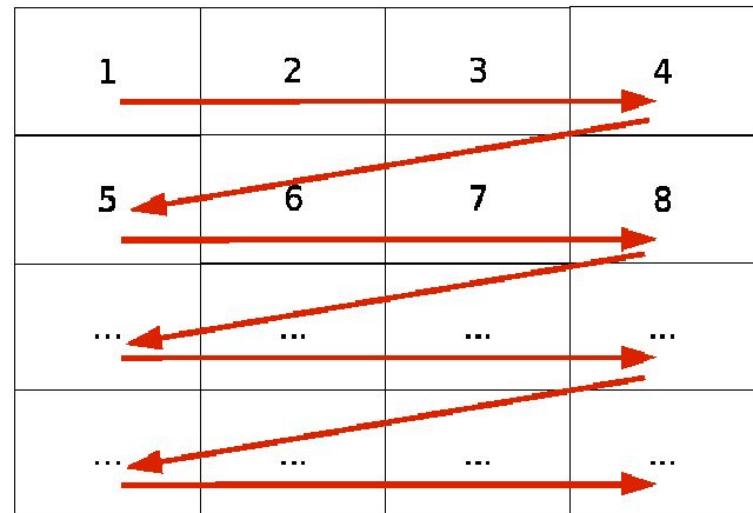
Se elige una permutación π del conjunto $[1, n]$ y modelar la densidad $p(x)$ de forma auto-regresiva de la siguiente manera:

$$p(x) = \prod_{i=1}^n p(x_{\pi_i} | x_{\pi_1}, \dots, x_{\pi_{i-1}}, \theta)$$

Cuando se trabaja con imágenes, se escoge la permutación π_i conocida como orden de rasterización.

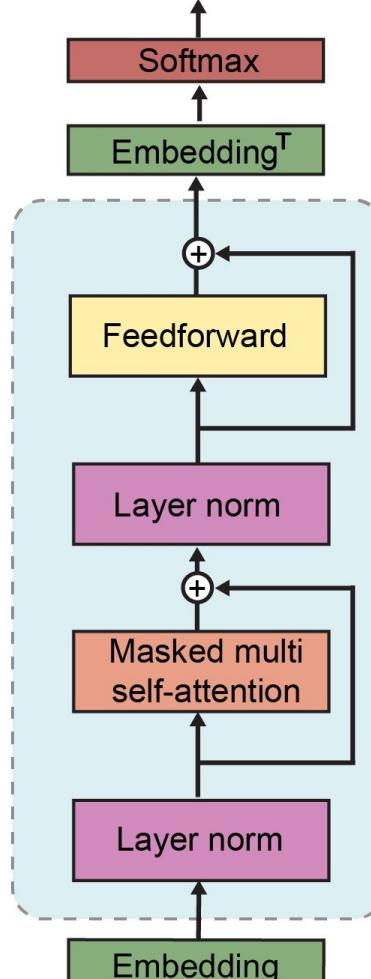
La función de perdida busca minimizar la verosimilitud:

$$L_{AR} = \mathbb{E}_{x \sim X} [-\log p(x)]$$



ImageGPT

$$p(u_n | u_1, \dots, u_{n-1})$$

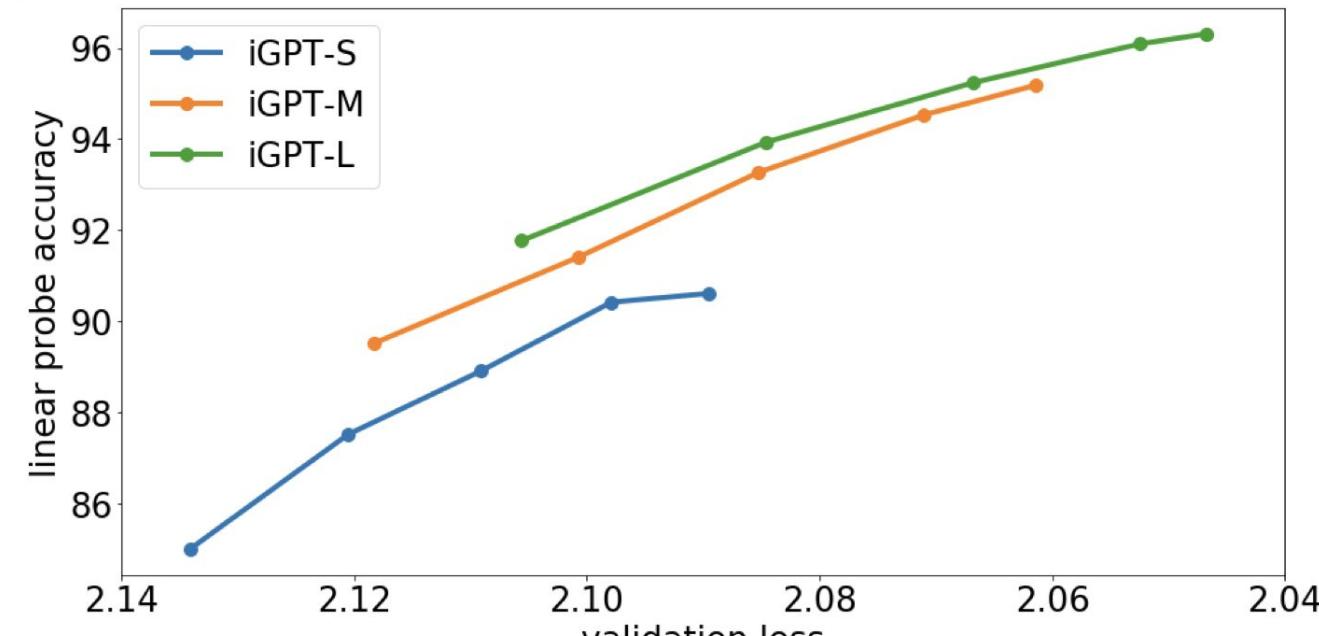


El transformer decoder toma como entrada la secuencia de tokens y produce embeddings para cada posición. Los embeddings son la entrada a GPT, utilizando la misma arquitectura que GPT-2.

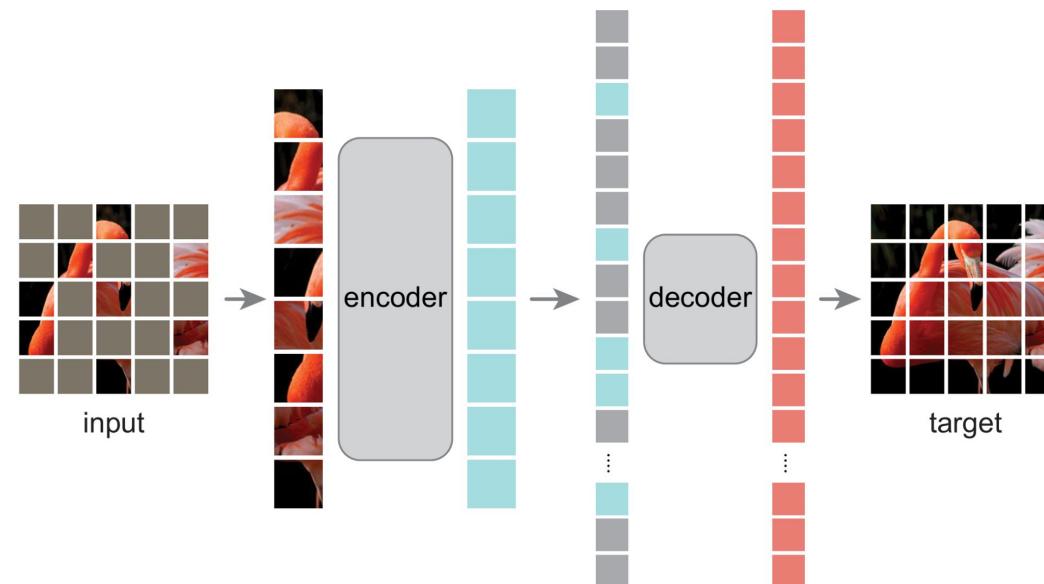
$$n^l = \text{layer_norm}(h^l)$$

$$a^l = h^l + \text{multihead_attention}(n^l)$$

$$h^{l+1} = a^l + \text{mlp}(\text{layer_norm}(a^l))$$



Masked AutoEncoder (MAE)



MAE fue propuesto en “[Masked Autoencoders Are Scalable Vision Learners](#)” en 2021 como un método escalable para tareas de visión mediante el uso de autoencoders enmascarados. [Link-huggingface](#)

El objetivo consiste en **enmascarar parches aleatorios** de la imagen de entrada (encoder) y reconstruirlas (decoder), para aprender representaciones visuales.

Se enmascara hasta 75% de la imagen. Se aplica en encoder sobre el 25% restante, luego se agregan mask tokens que sirve como input al decoder el cual reconstruye la imagen en pixeles.

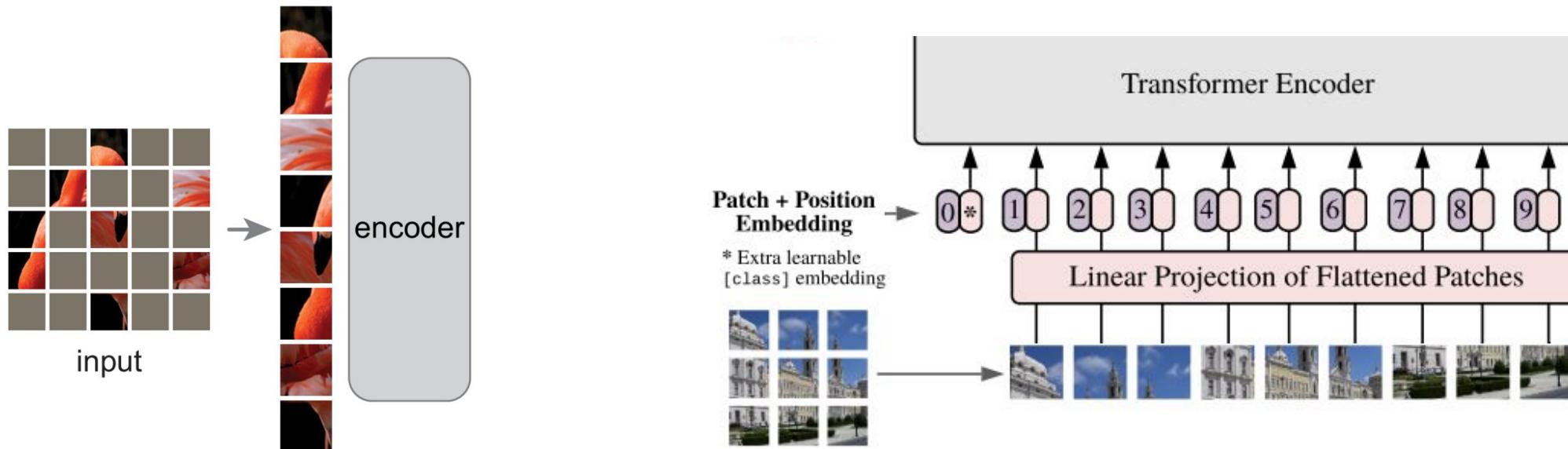
MAE Encoder

El encoder no es nada más que **ViT vanilla**, donde realmente el proceso inicial es idéntico con la diferencia en realizar **masking** sobre los parches.

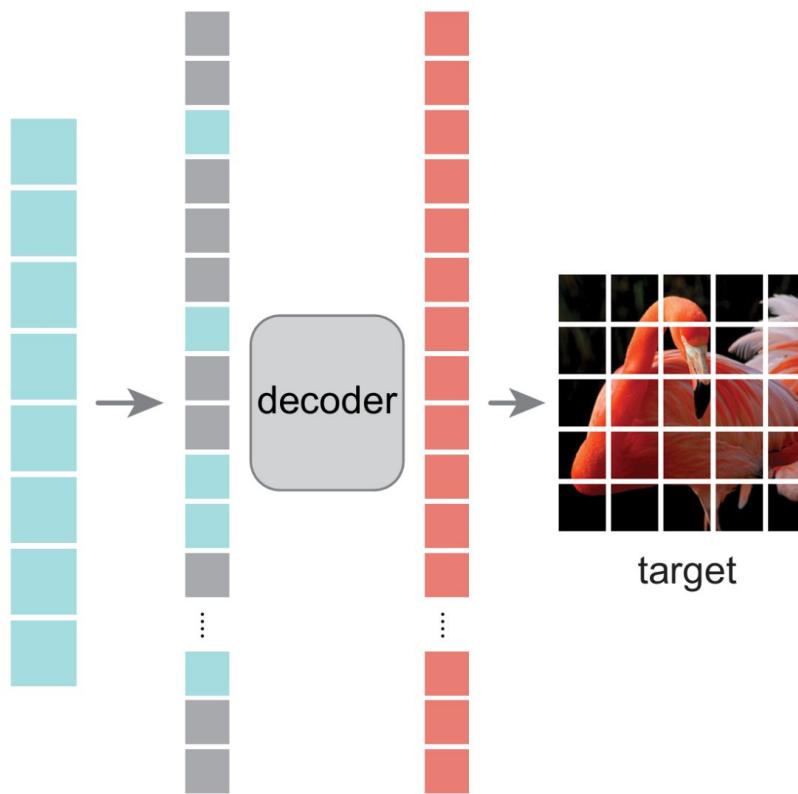
Debido a que 25% de la imagen es enmascarada, esto vuelve eficiente y escalable al ViT vanilla, donde se aplica attention sobre una porción muy pequeña.

El encoder agrega **positional embeddings senoidales**.

ViT puede ser sustituido por variantes más eficientes como Swin-T, etc.



MAE Decoder



El decoder (Transformer blocks) se encarga de mapear la representación latente del input mediante la reconstrucción de pixeles faltantes, da como **salida** un vector de valores de píxeles.

El decoder recibe como entrada:

- parches visibles codificados
- mask tokens

Cada mask token es un vector aprendido (parámetro entrenable) que indica la presencia de un parche faltante para ser predicho.

Se debe tomar en cuenta agregar **positional embeddings senoidales** a todos los tokens ya que los mask tokens no tienen información espacial.

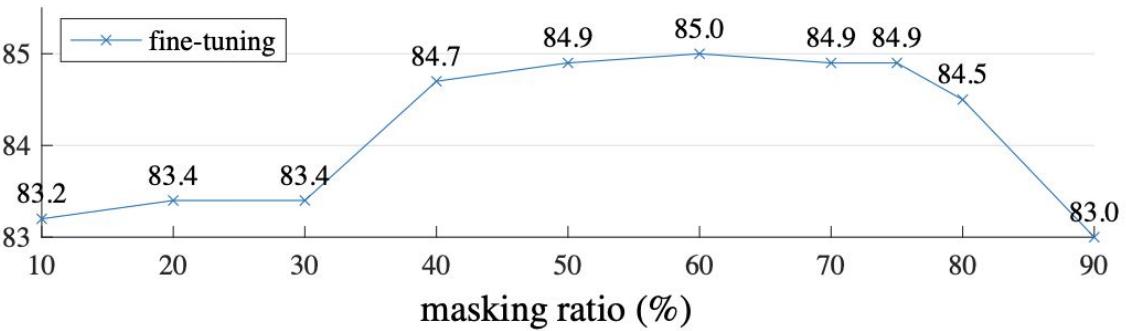
El decoder puede ser solamente utilizado durante pre-training para realizar la tarea de reconstrucción de imágenes.

El encoder puede funcionar solo para producir representación de imágenes para reconocimiento.

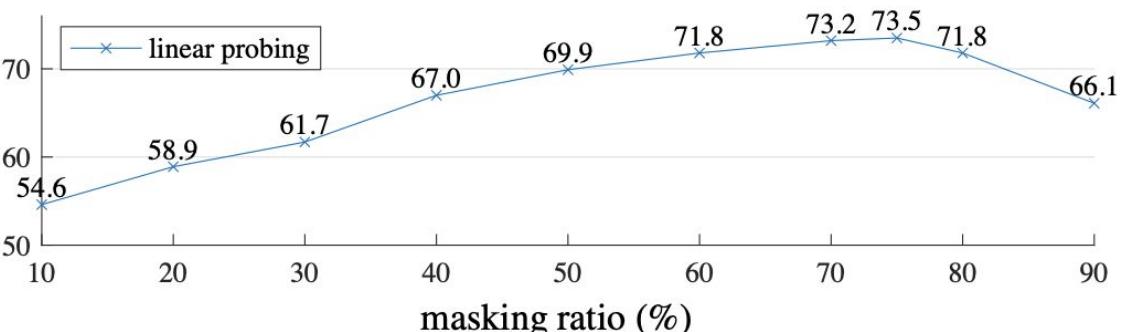
MAE Decoder

blocks	ft	lin
1	84.8	65.5
2	84.9	70.0
4	84.9	71.9
8	84.9	73.5
12	84.4	73.3

(a) **Decoder depth.** A deep decoder can improve linear probing accuracy.



(b) **Decoder width.** The decoder can be narrower than the encoder (1024-d).



dim	ft	lin
128	84.9	69.1
256	84.8	71.3
512	84.9	73.5
768	84.4	73.1
1024	84.3	73.1

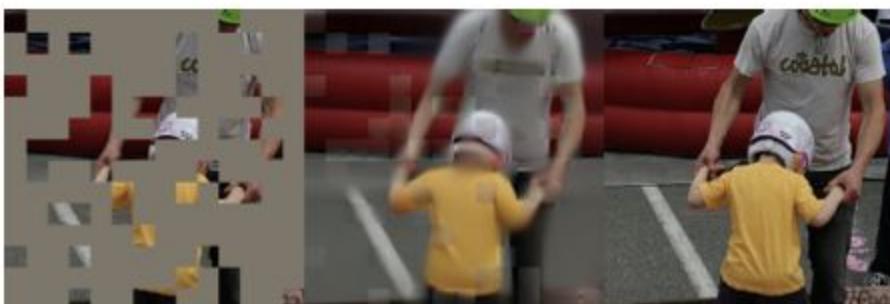
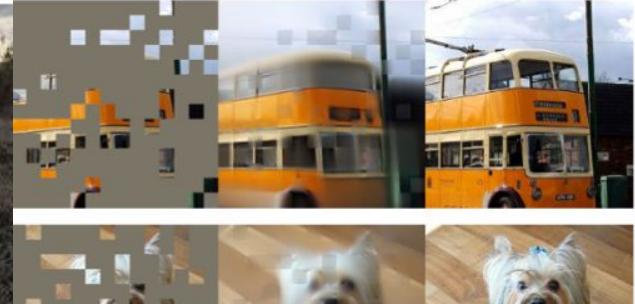
encoder	dec. depth	ft acc	hours	speedup
ViT-L, w/ [M]	8	84.2	42.4	-
ViT-L	8	84.9	15.4	2.8×
ViT-L	1	84.8	11.6	3.7×
ViT-H, w/ [M]	8	-	119.6 [†]	-
ViT-H	8	85.8	34.5	3.5×
ViT-H	1	85.9	29.3	4.1×

MAE visualized

MAE implementation

MAE visualize

MAE Medium Explanation



DALL-E

DALL-E fue presentado en el paper “[Zero-Shot Text-to-Image Generation](#)” consiste en realizar la tarea de generación de imágenes mediante texto.

Entrenando un transformer para el modelado autoregresivo de texto e imágenes como un solo stream de data.

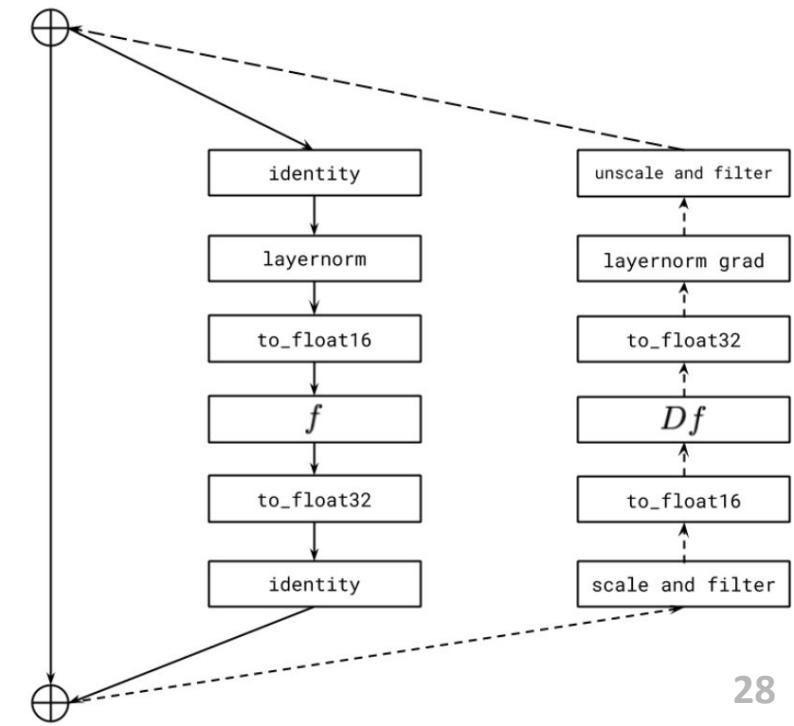


(a) a tapir made of accordion.
a tapir with the texture of an
accordion.

(b) an illustration of a baby
hedgehog in a christmas
sweater walking a dog

(c) a neon sign that reads
“backprop”. a neon sign that
reads “backprop”. backprop
neon sign

(d) the exact same cat on the
top as a sketch on the bottom



DALL-E

Utiliza dos etapas para el entrenamiento:

- Entrenar un [variational autoencoder discreto, paper](#).
- Entrenar un transformer para modelar la distribución conjunta del texto tokenizado e image tokens.

Lo que se procura realizar es maximizar la verosimilitud conjunta del modelado de la distribución sobre imágenes x , captions y y los tokens z .

$$\ln p_{\theta, \psi}(x, y) \geq \mathbb{E}_{z \sim q_{\phi}(z | x)} (\ln p_{\theta}(x | y, z) - \beta D_{\text{KL}}(q_{\phi}(y, z | x), p_{\psi}(y, z)))$$

$$p_{\theta, \psi}(x, y, z) = p_{\theta}(x | y, z)p_{\psi}(y, z)$$

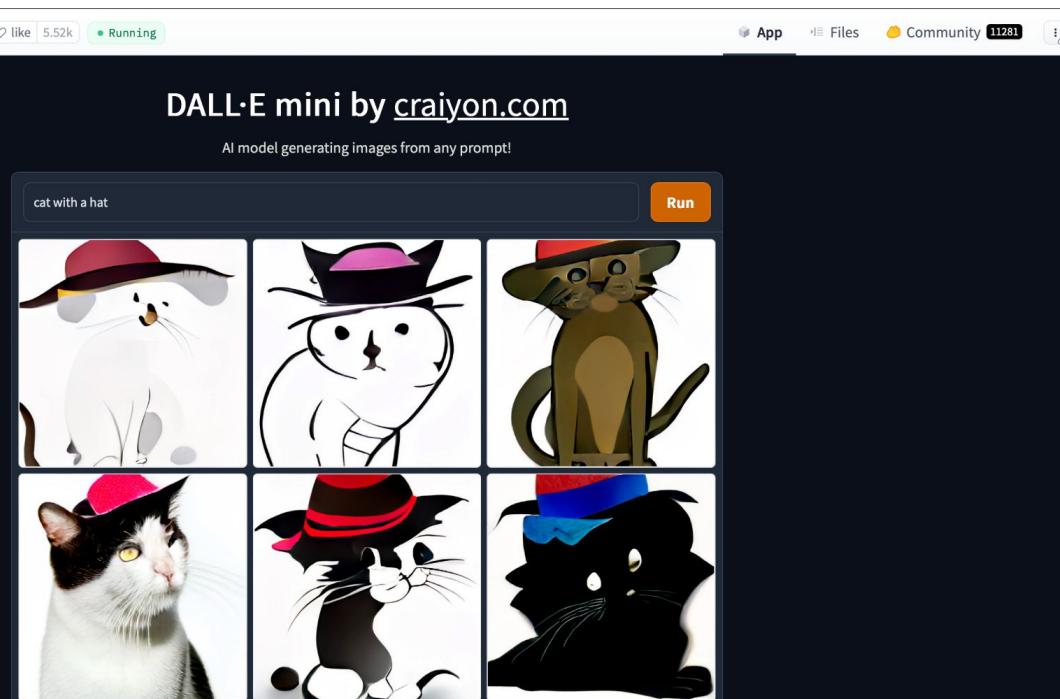
- q_{ϕ} denotes the distribution over the 32×32 image tokens generated by the dVAE encoder given the RGB image x^2 ;
- p_{θ} denotes the distribution over the RGB images generated by the dVAE decoder given the image tokens; and
- p_{ψ} denotes the joint distribution over the text and image tokens modeled by the transformer.

DALL-E

DALL-E Mini HF Space

DALL-E XL LoRA

DALL-E Explained (Recomendable leer)



a very cute giraffe making a funny face.



a kitchen with a fridge, stove and sink



a group of animals are standing in the snow.



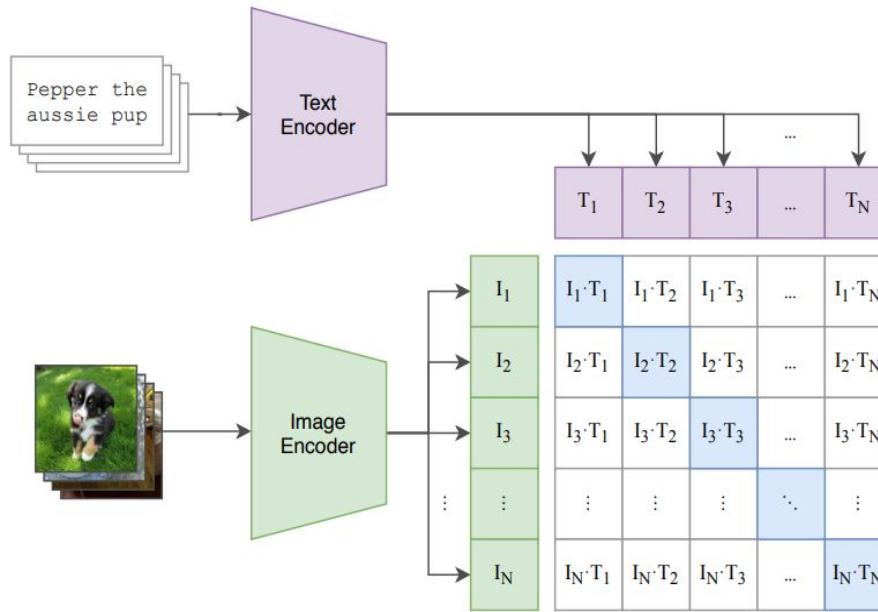
CLIP

CLIP(Contrastive Language-Image Pretraining) fue presentado en el paper “[Learning Transferable Visual Models From Natural Language Supervision](#)” (2021) para la tarea de predicción de descripción de imágenes.

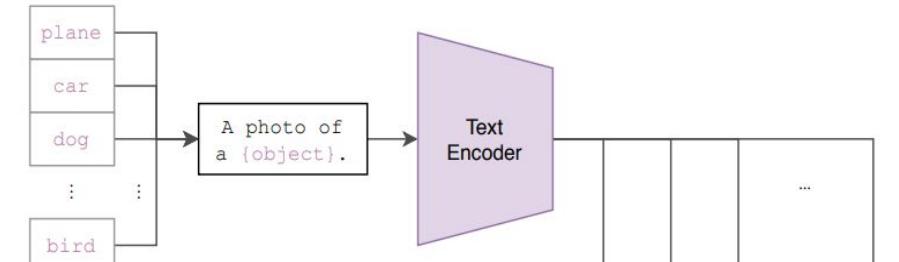
En NLP, el pre-training de GPT-3 trae la premisa de generalizar tareas mediante el uso de datasets masivos sin necesidad de labels. En CLIP, se investiga si lo mismo es aplicable en Visión.

Github

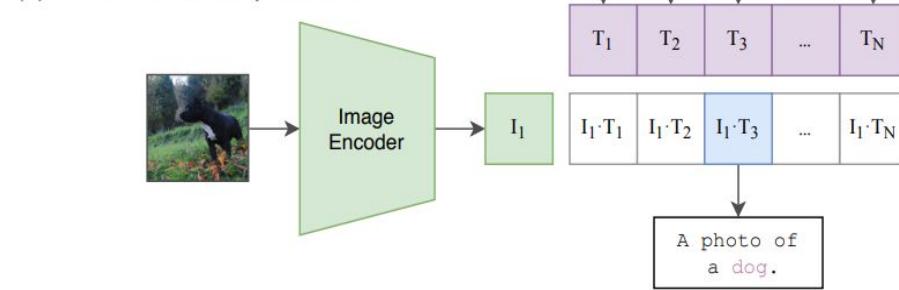
(1) Contrastive pre-training



(2) Create dataset classifier from label text



(3) Use for zero-shot prediction



Aprender características de imágenes mediante lenguaje natural es más fácil de escalar a comparación de labeling estándar.

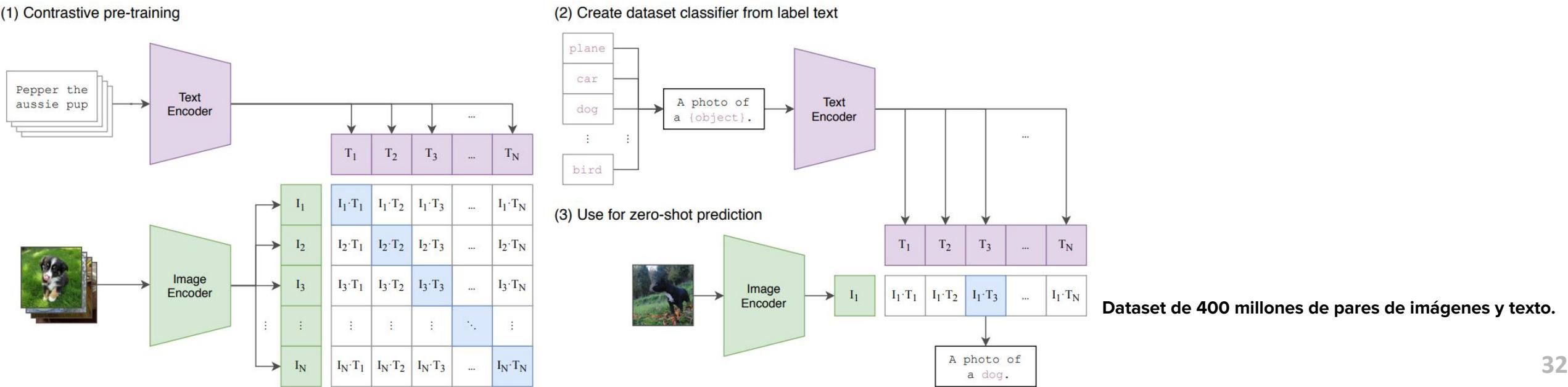
Esta manera permite no “solo” aprender la representación sino conectar la representación con lenguaje, haciendo que sea posible **zero-shot** transfer similar a GPT-3.

CLIP

CLIP, consiste en modelar la distribución conjunta entre las imágenes y el texto para encontrar los pares de imagen y texto.

Dado un batch de **N** pares de imágenes y texto, CLIP predice cuáles de los pares **N x N** realmente sucedieron. Para ello CLIP aprende los espacio de embeddings multimodales mediante el conjunto de los encoder para maximizar la **similitud del coseno** de los embeddings de los pares **N** reales mientras se minimiza la similitud del cos para los pares erróneos.

El paper utiliza como dos propuestas como Image encoder Resnet-50 y ViT, pero los **mejores resultados** fue mediante ViT.



CLIP

El text encoder consiste en un Transformer con modificaciones encontradas en GPT-2 y Byte Pair Encoding (BPE) como tokenizer.

La ResNet tomo 18 días en entrenar en 592 V100 GPUs mientras ViT tomo 12 días en 256 V100 GPUs.

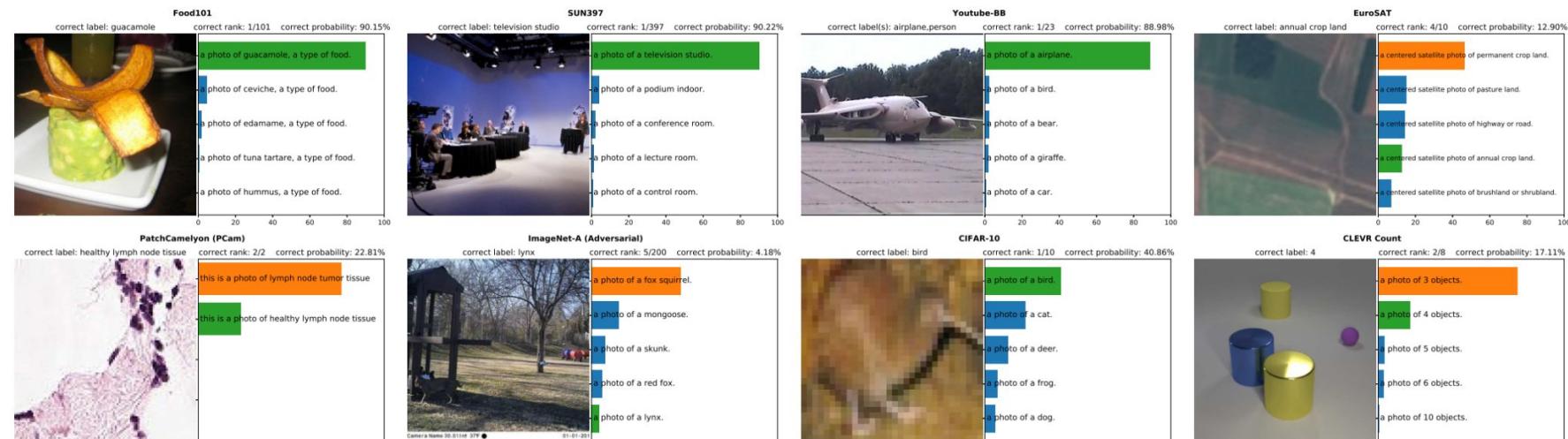
```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = 12_normalize(np.dot(I_f, W_i), axis=1)
T_e = 12_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```



Model	Learning rate	Embedding dimension	Input resolution	Vision Transformer layers	width	heads	Text Transformer layers	width	heads
ViT-B/32	5×10^{-4}	512	224	12	768	12	12	512	8
ViT-B/16	5×10^{-4}	512	224	12	768	12	12	512	8
ViT-L/14	4×10^{-4}	768	224	24	1024	16	12	768	12
ViT-L/14-336px	2×10^{-5}	768	336	24	1024	16	12	768	12

CLIP

[Website](#)

[Understanding OpenAI's CLIP model](#)

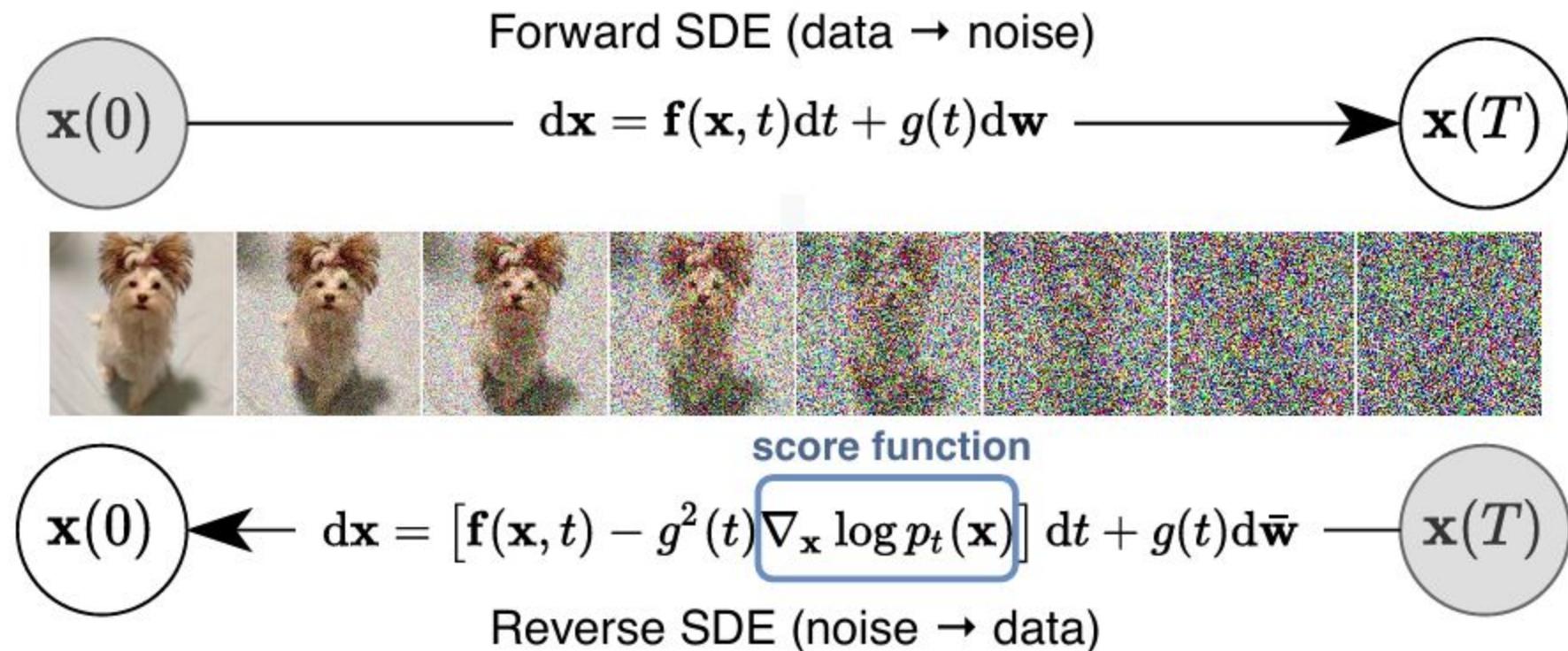
[Huggingface](#)

[Zero-shot object detection with CLIP](#)

[Roboflow CLIP](#)

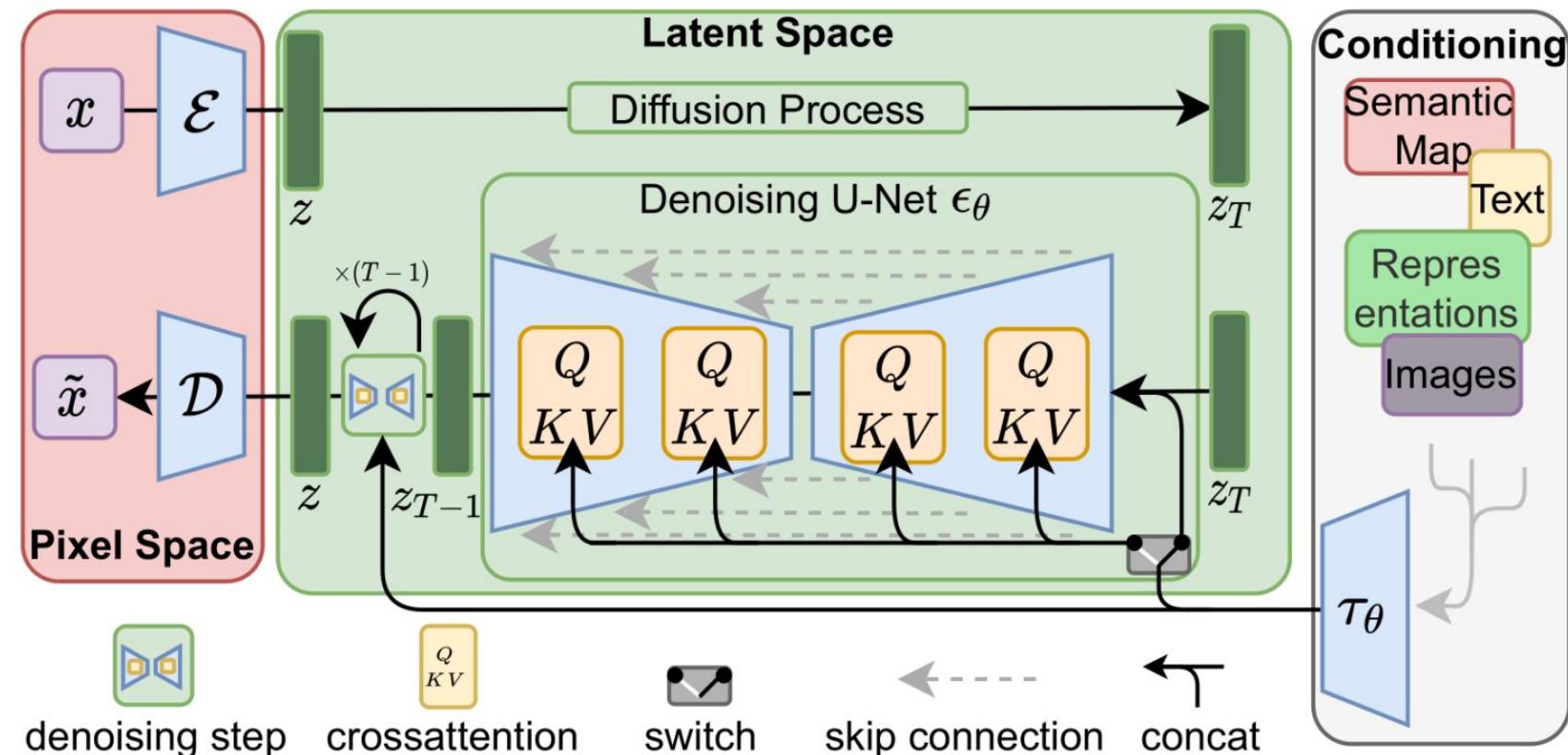
Diffusion Models

El enfoque score-based en modelado generativo mediante ecuación diferencial estocástica (SDE) fue presentado en el paper “[SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS](#)”



Diffusion Models

Latent Diffusion Model (LDM) fue presentado en el paper
[“High-Resolution Image Synthesis with Latent Diffusion Models”](#)

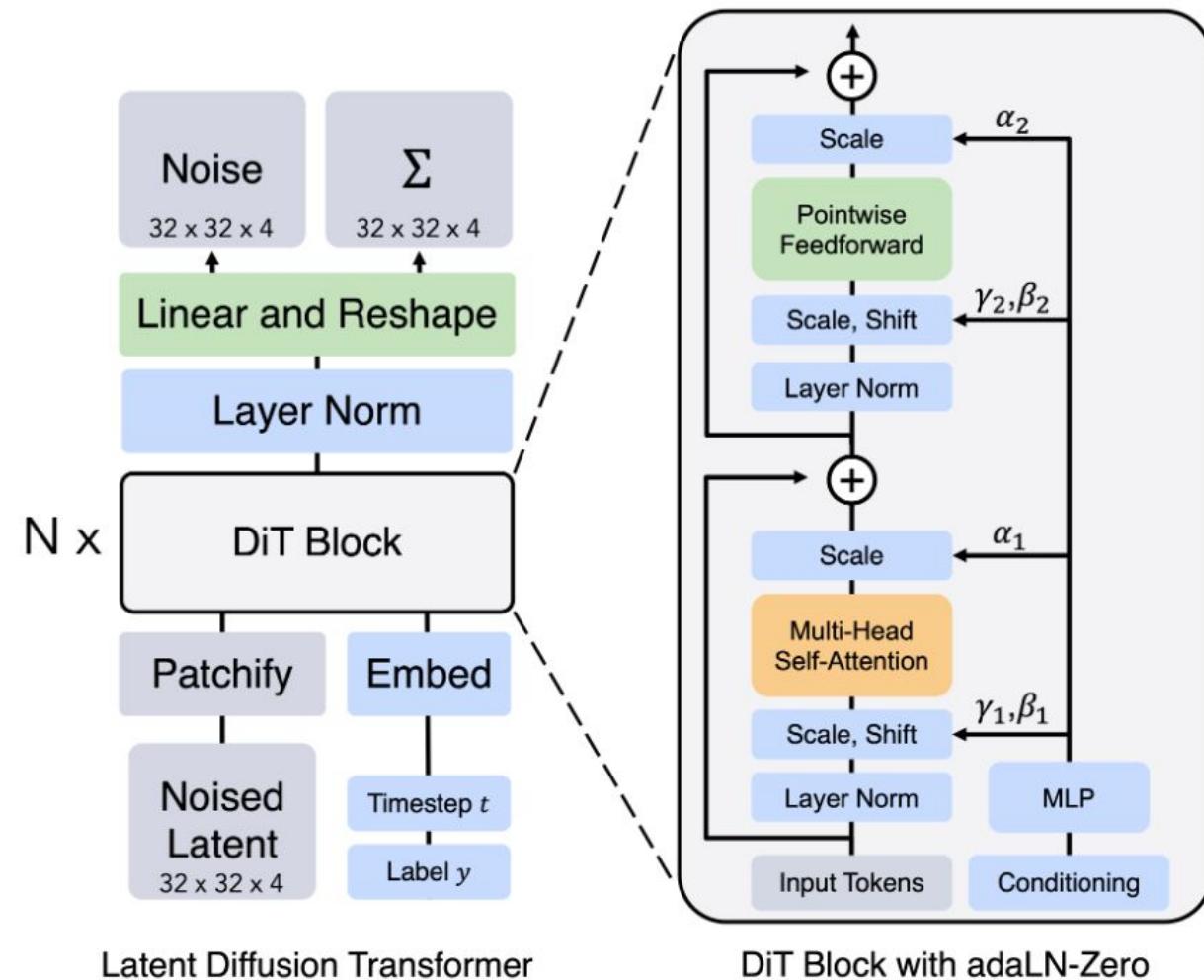


Stable Diffusion

- CompVis: Latent Diffusion Model (LDM), base técnica de Stable Diffusion. ([CompVis](#))
 - [“High-Resolution Image Synthesis with Latent Diffusion Models”](#)
 - [CompVis/stable-diffusion: A latent text-to-image diffusion model](#)
 - [CompVis/stable-diffusion-v1-4 · Hugging Face](#)
- Stability AI: Financiación y recursos para entrenar el modelo en grandes datos. ([Stability AI](#))
 - [Stable Diffusion launch announcement — Stability AI](#)
 - [Stability-AI/generative-models: Generative Models by Stability AI](#)
- Runway: Implementación práctica y accesibilidad del modelo.

[Stable Diffusion pipelines](#)

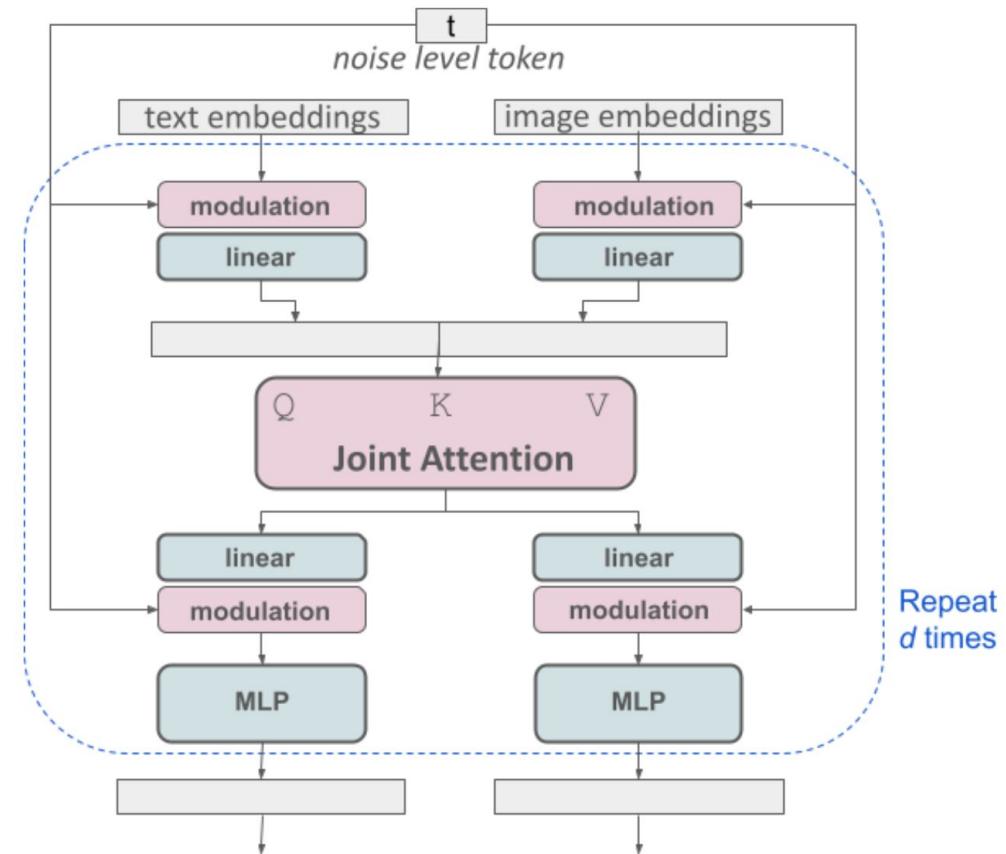
Diffusion Transformers



[Scalable Diffusion Models with Transformers](#)

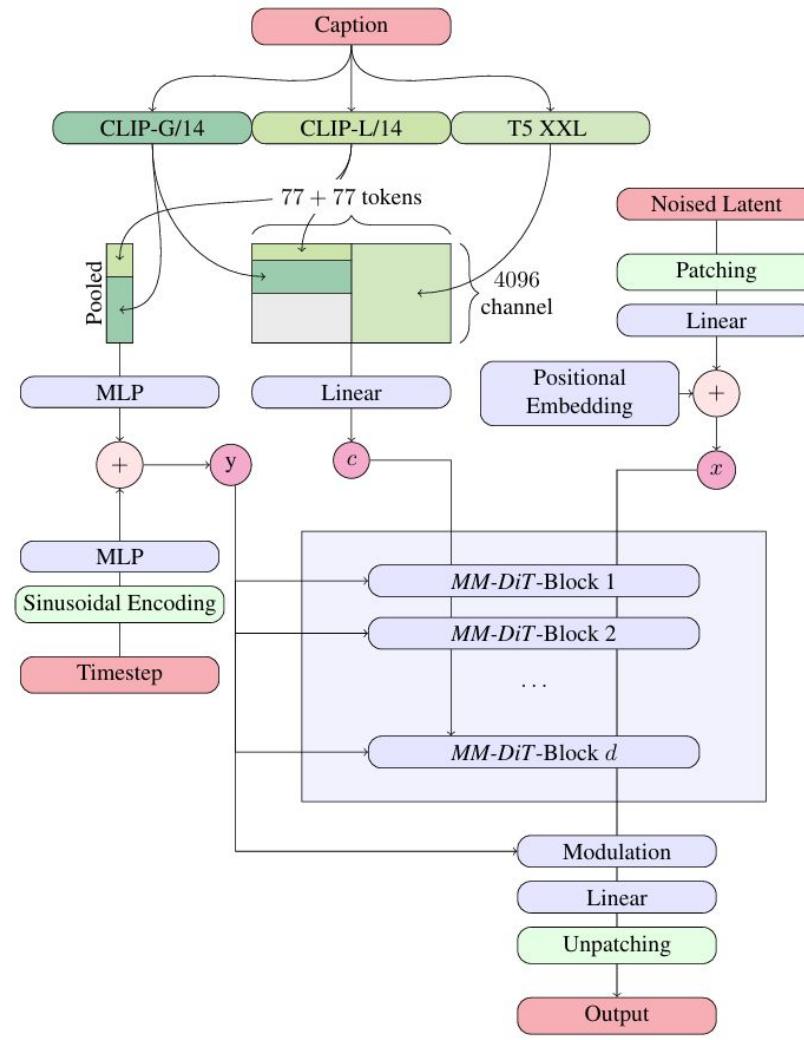
Stable Diffusion 3 (03/2024)

Stable Diffusion 3 es construido basado en el trabajo Scalable Diffusion Models with Transformers

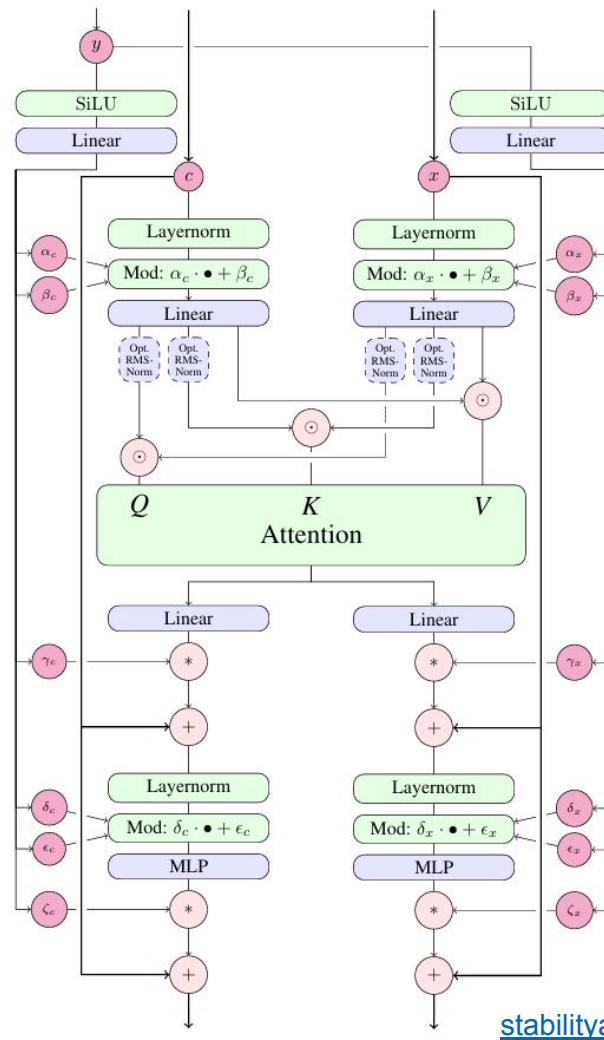


Conceptual visualization of a block of our modified multimodal diffusion transformer: MMDiT.

Stable Diffusion 3 (03/2024)



(a) Overview of all components.



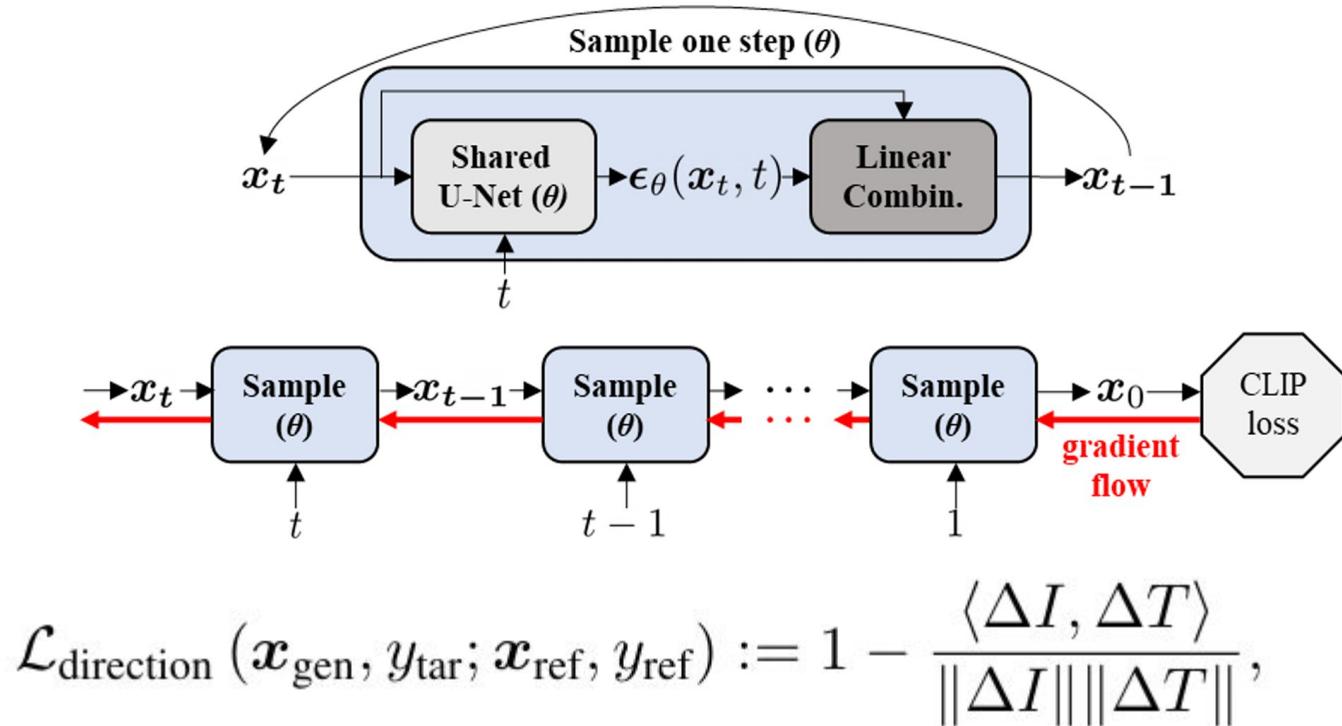
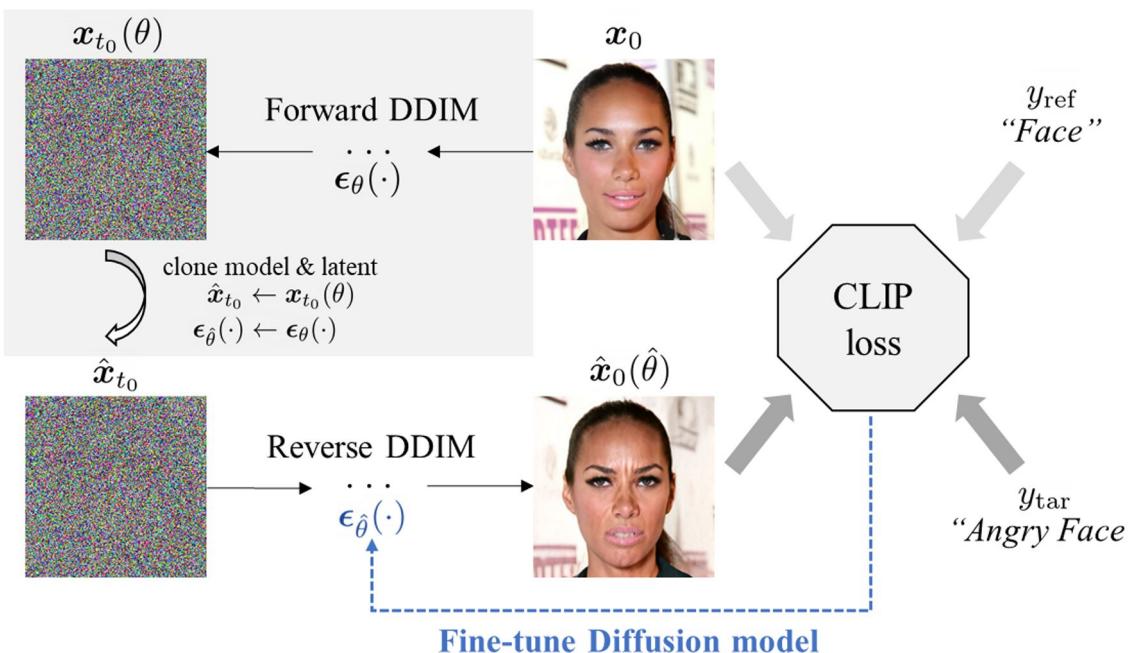
(b) One *MM-DiT* block

[stabilityai/stable-diffusion-3.5-large · Hugging Face](https://github.com/stabilityai/stable-diffusion-3.5-large)

DiffusionCLIP

[Github: gwang-kim/DiffusionCLIP](https://github.com/gwang-kim/DiffusionCLIP)

DiffusionCLIP fue presentado en el paper “[DiffusionCLIP: Text-Guided Diffusion Models for Robust Image Manipulation](#)”

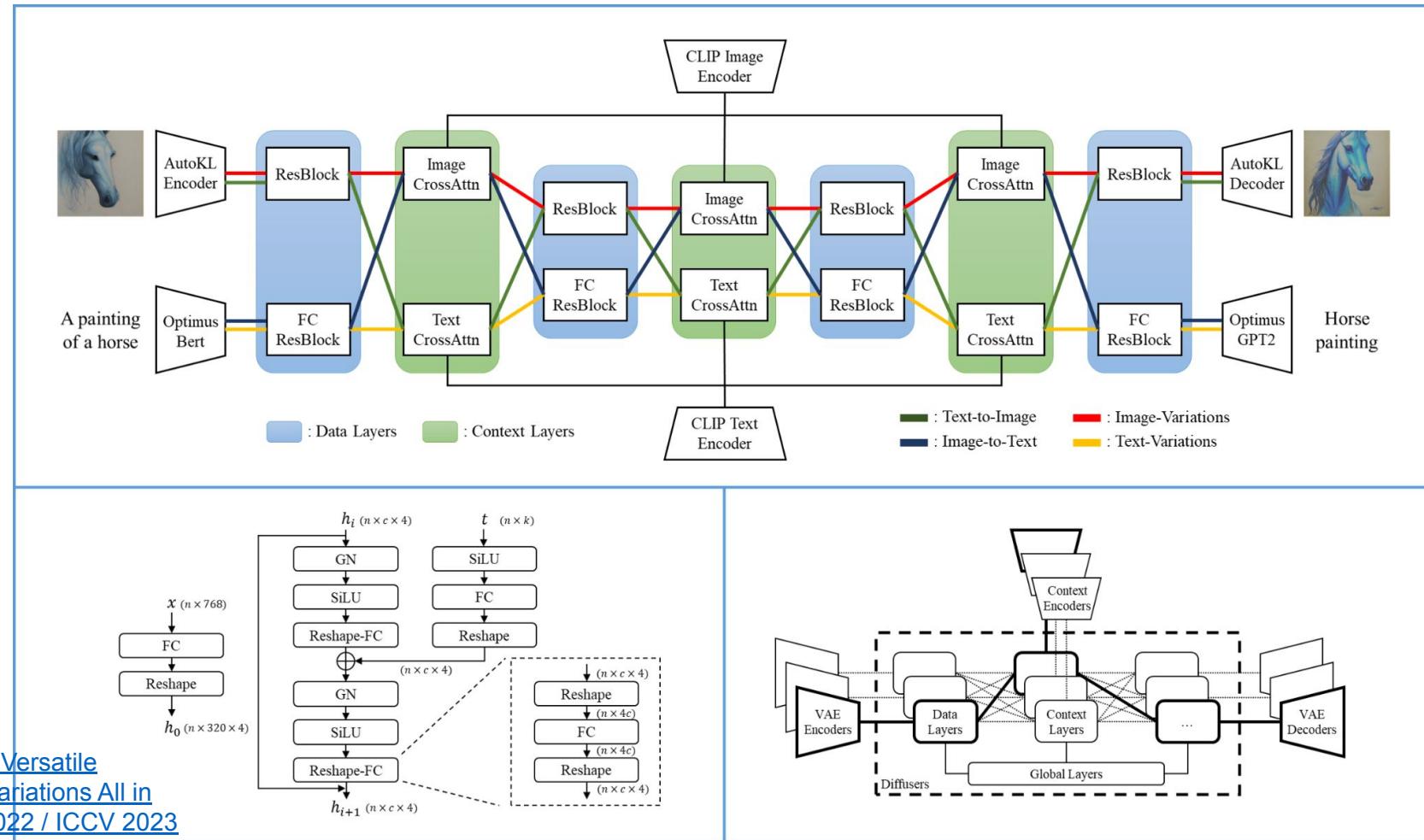


$$\mathcal{L}_{\text{direction}}(\mathbf{x}_{\text{gen}}, y_{\text{tar}}; \mathbf{x}_{\text{ref}}, y_{\text{ref}}) := 1 - \frac{\langle \Delta I, \Delta T \rangle}{\|\Delta I\| \|\Delta T\|},$$

$$\Delta T = E_T(y_{\text{tar}}) - E_T(y_{\text{ref}}), \quad \Delta I = E_I(\mathbf{x}_{\text{gen}}) - E_I(\mathbf{x}_{\text{ref}}).$$

Versatile-Diffusion (01/2024)

Versatile-Diffusion fue presentado en el paper “[Versatile Diffusion: Text, Images and Variations All in One Diffusion Model](#)”



Preguntas?