Lecture 6.3 – CLIP and Latent Diffusion

Generative AI Teaching Kit

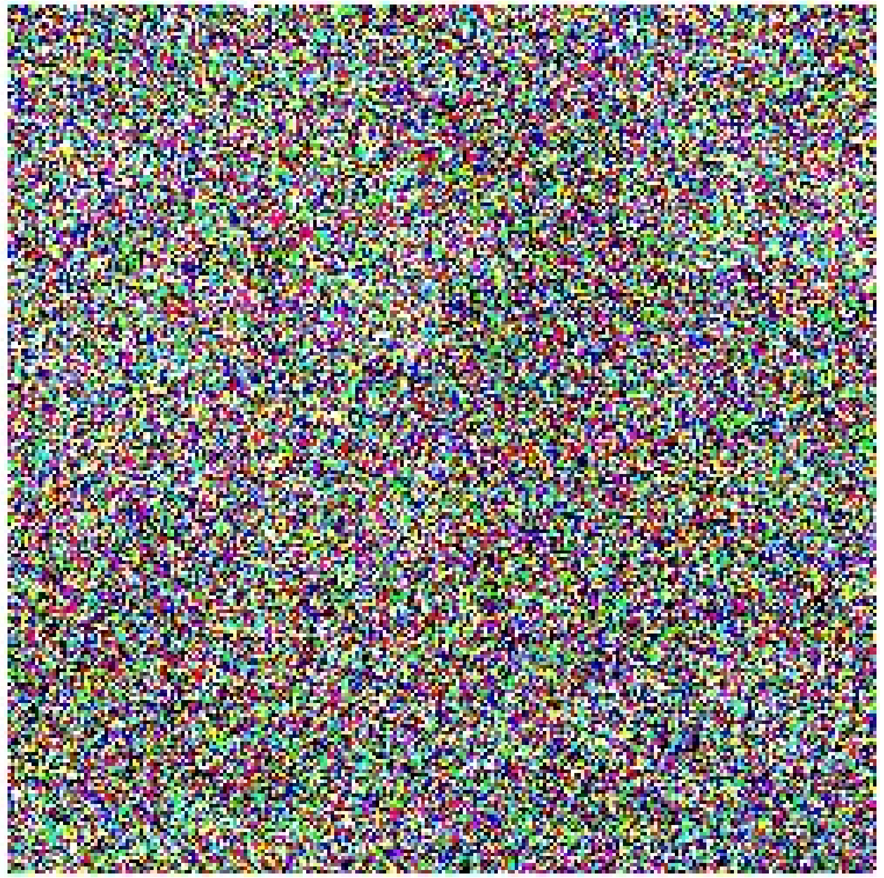
This lecture

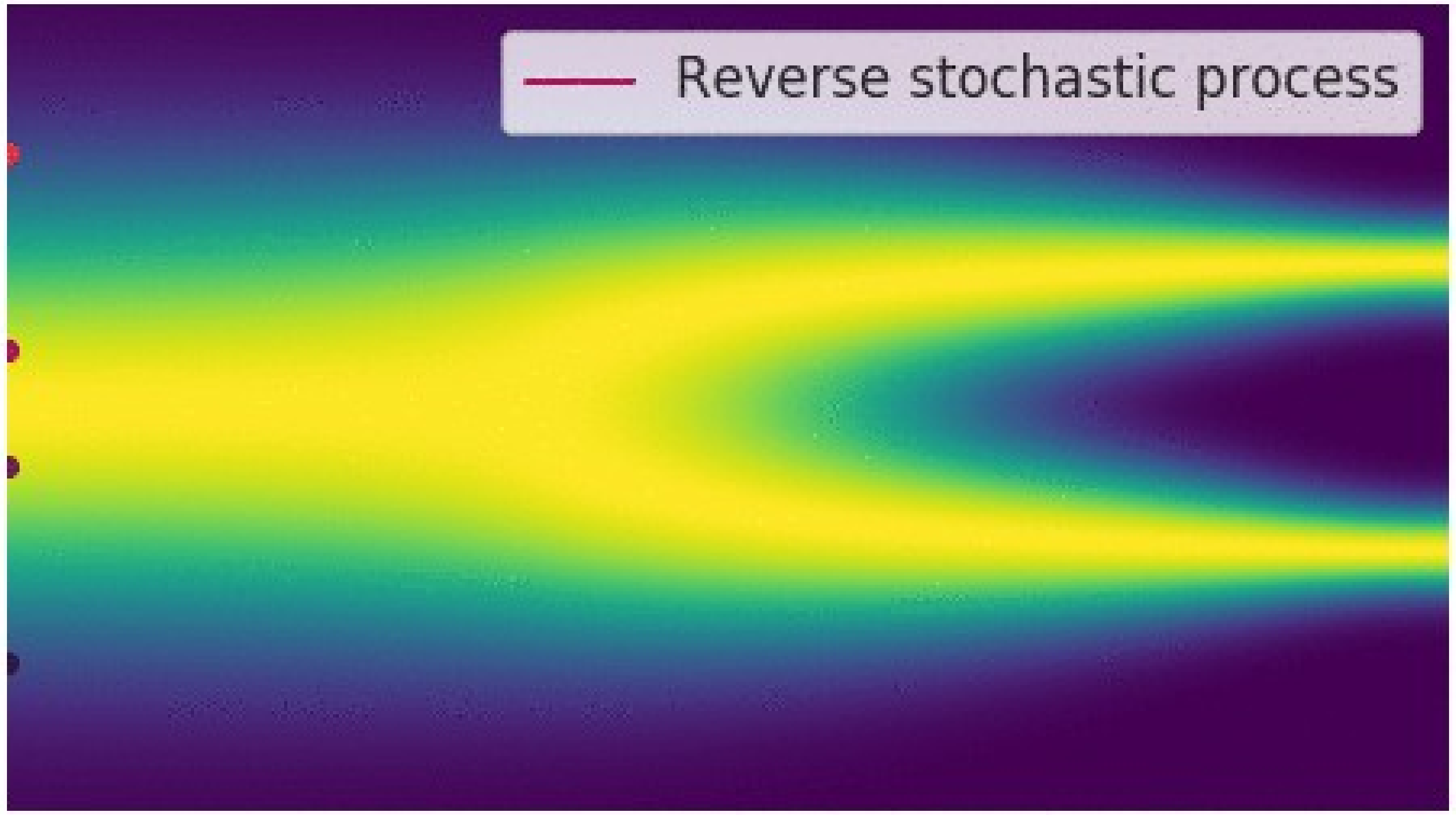
- Recap on Diffusion Models
- Latent Diffusion Models
- Text-Image Embeddings: CLIP
- Stable Diffusion and SOTA
- Future: Video Generation
- Wrap Up

Recap: Diffusion Models

Diffusion Models

- In the last lesson we introduced Diffusion Models as a new wayto generate data using the reverse diffusion process.
- With diffusion models, we can generate images from noise using a learned diffusion model

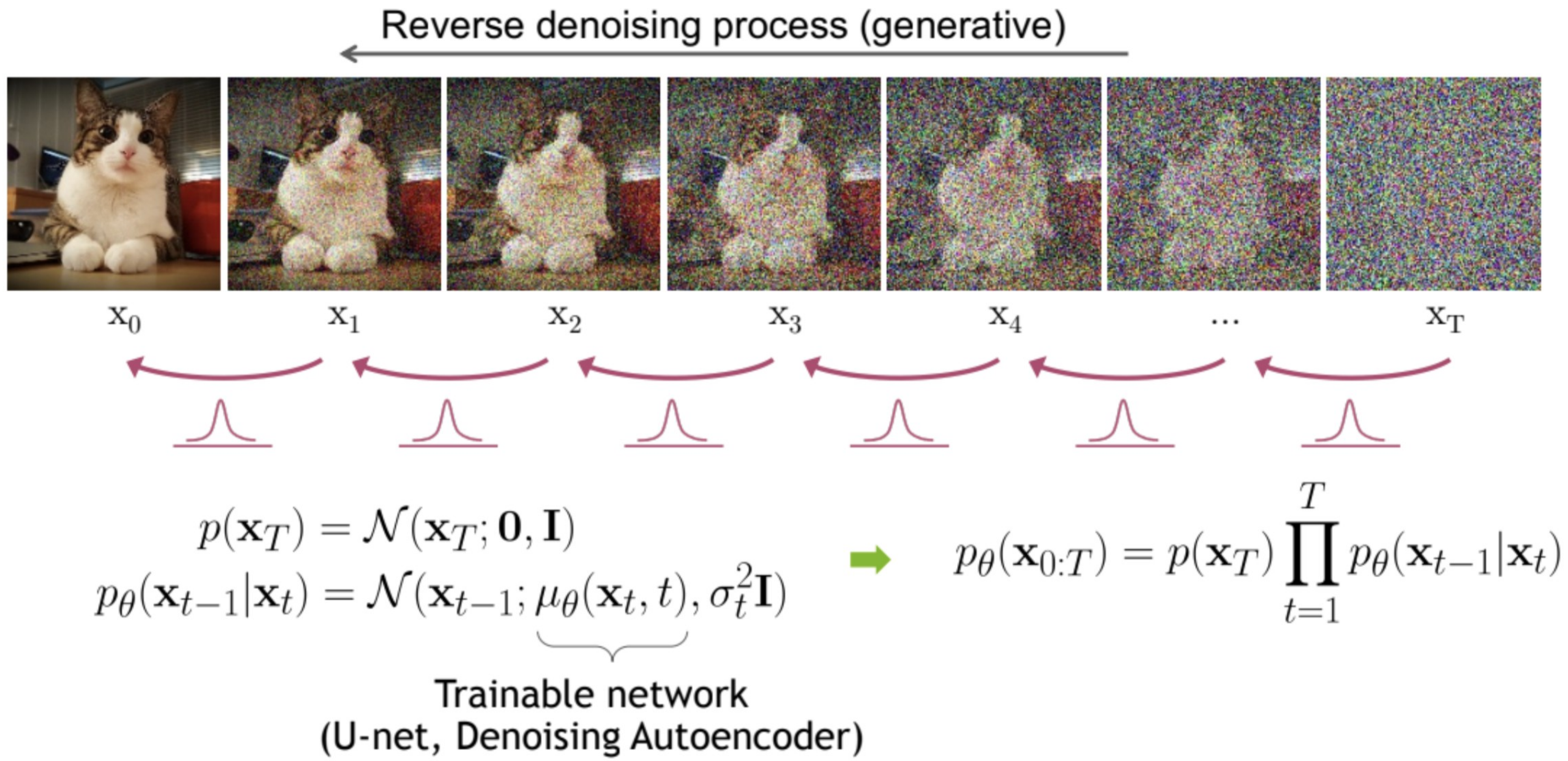




Diffusion Models

- A classical diffusion model predicts the noise in the images at each “timestep” and slowly reveals the image
- A neural network is trained to predict the noise to be removed at each stage using a timestep as a guide

- However, this classical model cannot take any input other than the noise and is slow to generate images.



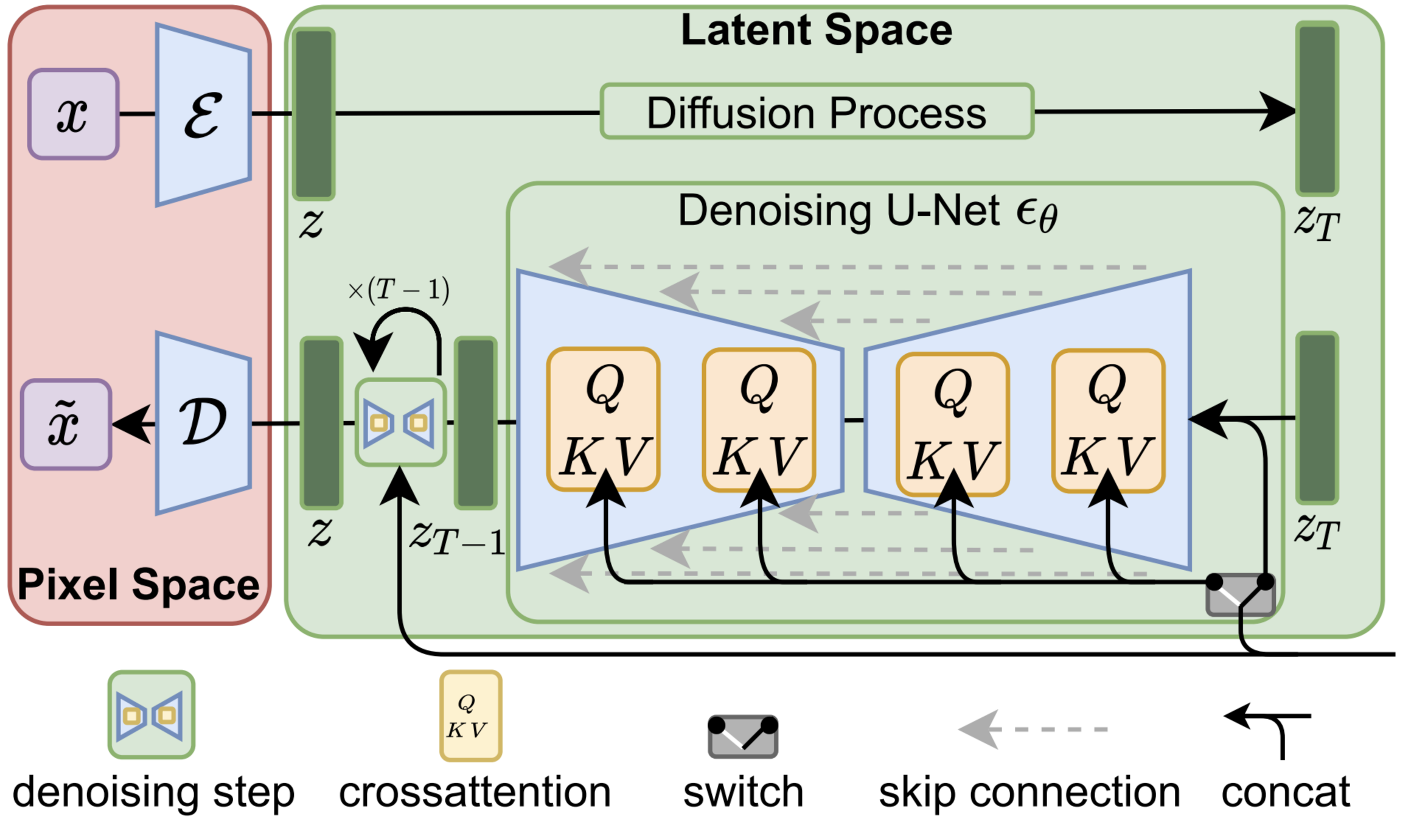
Latent Diffusion Models

Latent Diffusion Models

What if we perform the diffusion process on the latent space instead of the images?

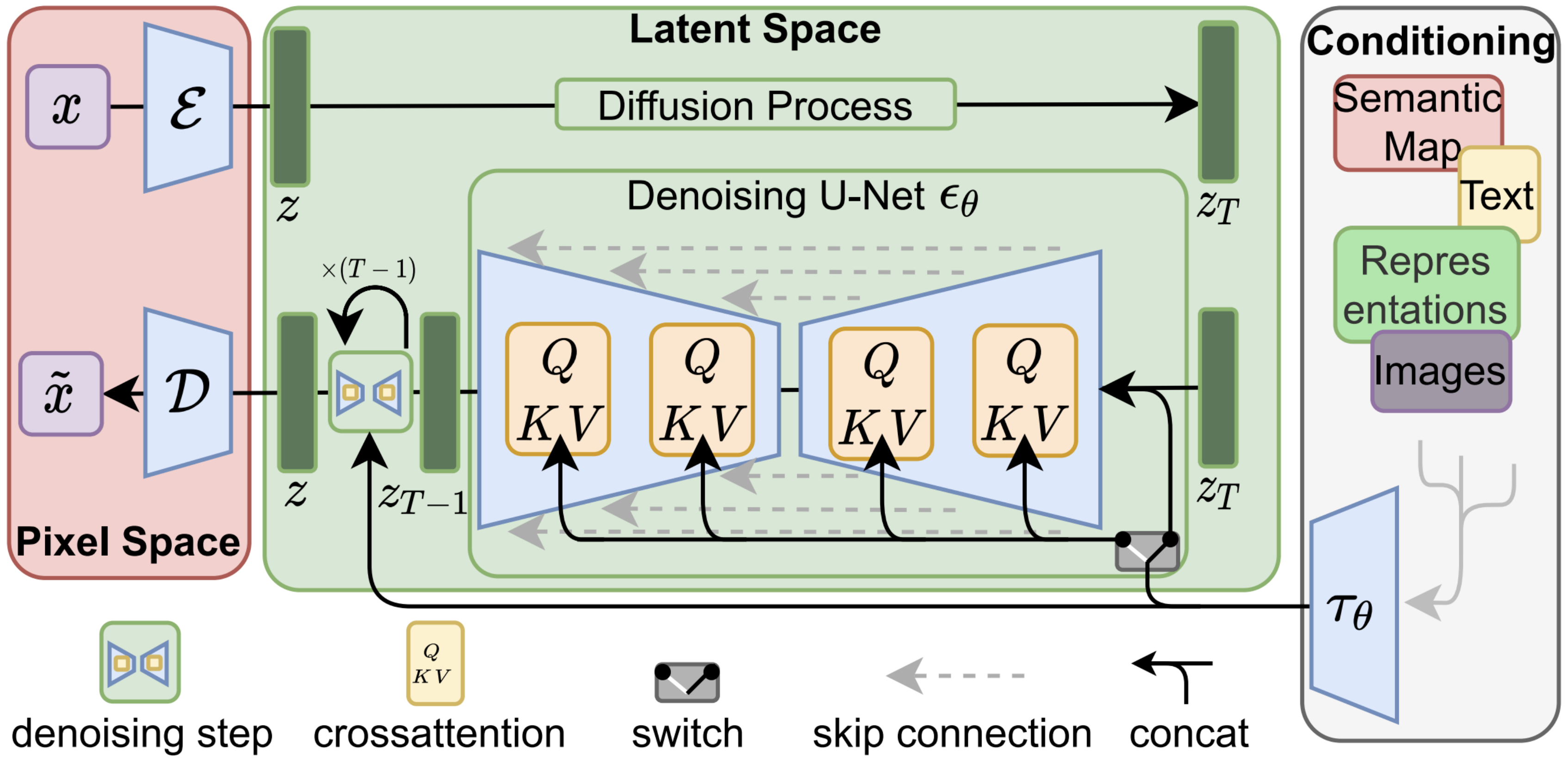
Latent Diffusion Models

- Have access to an efficient, low-dimensional latent space
- High-frequency, imperceptible details are abstracted away.
- Focus on the important, semantic bits of the data
- Train in a lower dimensional
- Computationally much more efficient space



Conditional Generation with LDM

- The latent space abstracts out the specific form of the data
- This allows for conditioning of generation using:
  - Text
  - Semantic Maps
  - Images
  - etc.



Essentially any data can be converted to the latent space and the attention mechanism will inform the reverse diffusion process

Text Conditioning in Latent Diffusion Models

1. Image Preparation

Image Feature Extraction: The U-Net processes the input image and extracts feature maps.

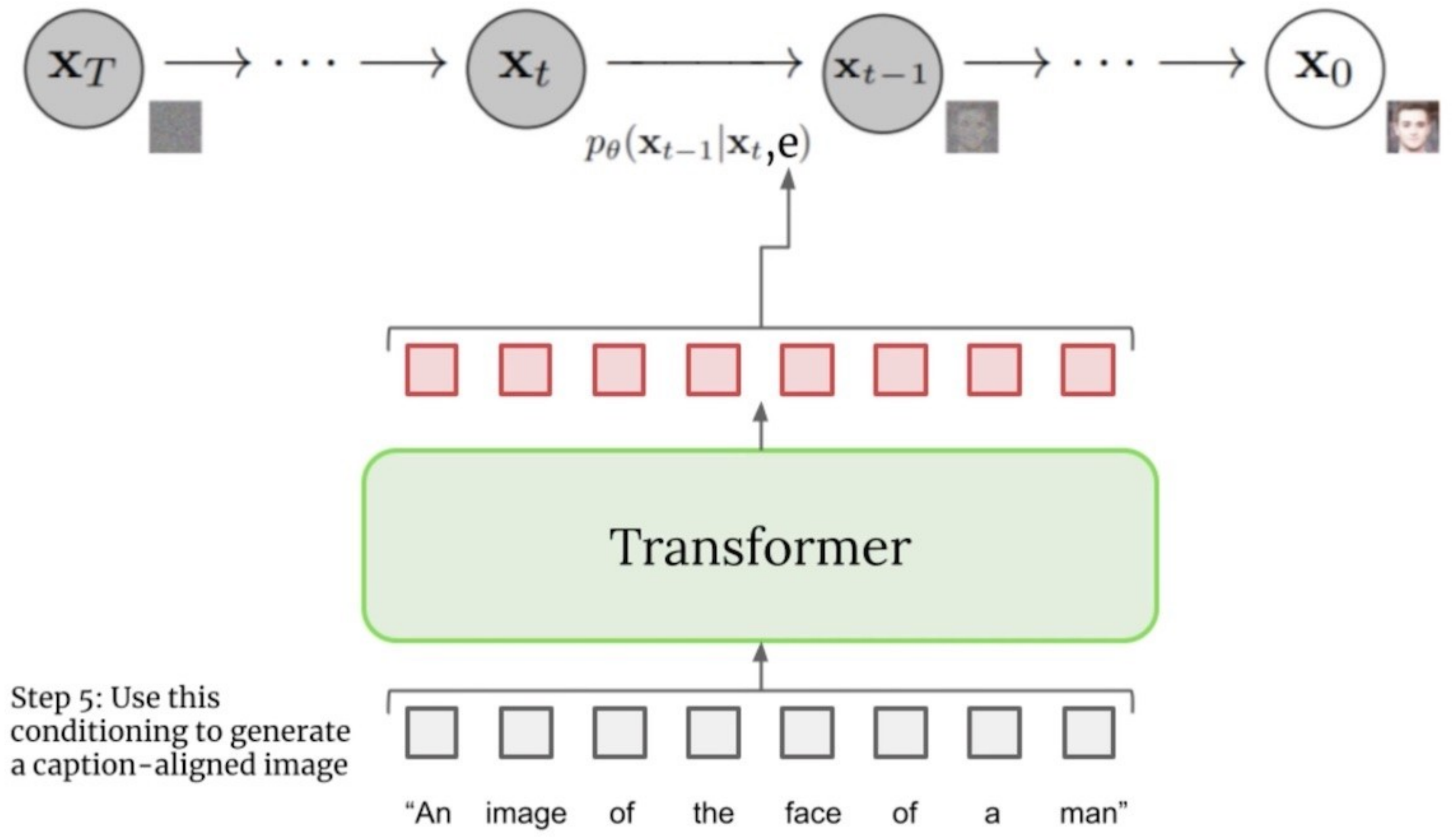
Reshaping for Attention: The 3D feature maps can be flattened into a 2D format.

2. Text Preparation

Text Embedding Generation: Use a pretrained text encoder to convert the text input into a fixed-length embedding vector.

3. Compute Attention Mechanism between Text and Image Features

Compute Cross-Attention: The cross-attention mechanism computes the relationships between the image features (queries) and the text embeddings (keys and values). For each of the tokens in the image features, attention weights are computed based on the similarity to the text embedding.

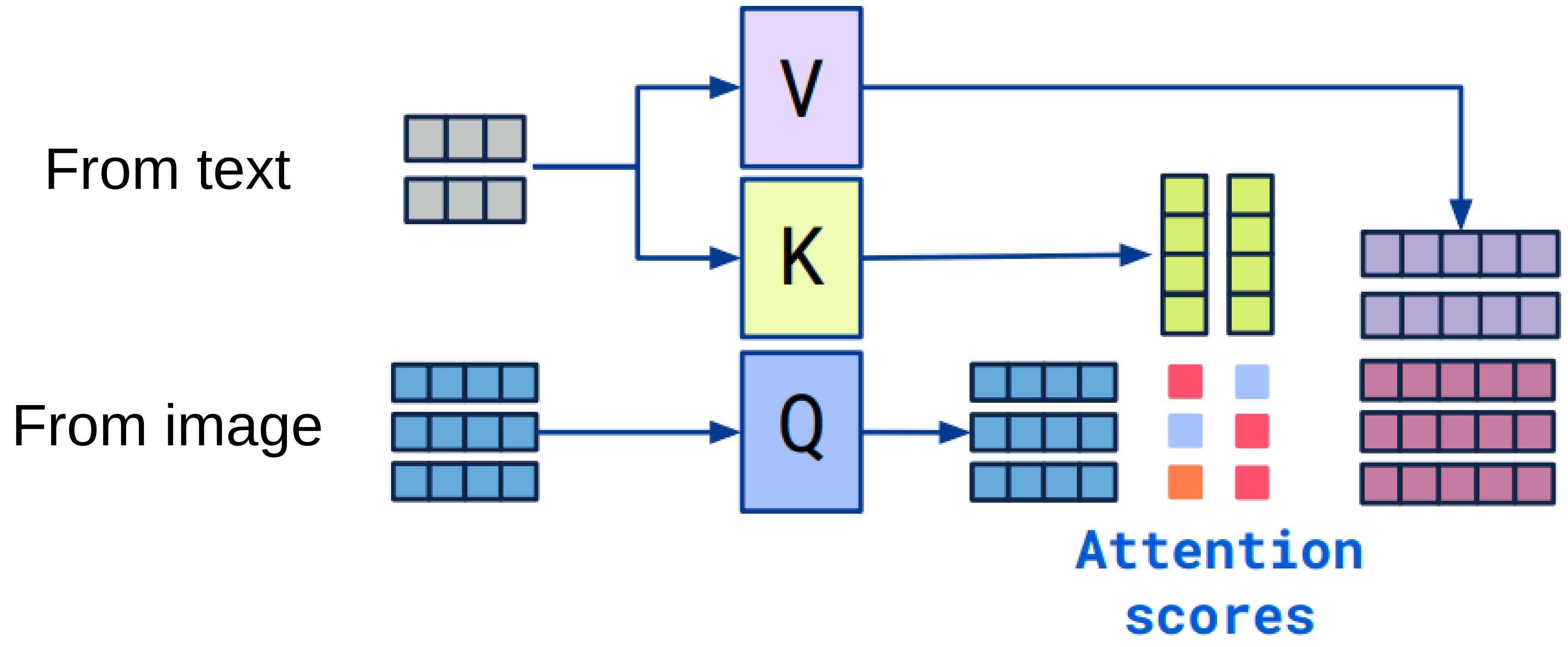


Integration of Conditioned Features

The attention output, now containing text-conditioned information, is reshaped back to the original spatial dimensions of the U-Net feature maps. The U-Net processes the conditioned feature maps through its remaining layers to generate the final output image, which is conditioned on the text input.

Cross-Attention Mapping – Understanding prompts

We can use the cross-attention mechanism in these latent diffusion models to trace the highest attention for each word in the generated image



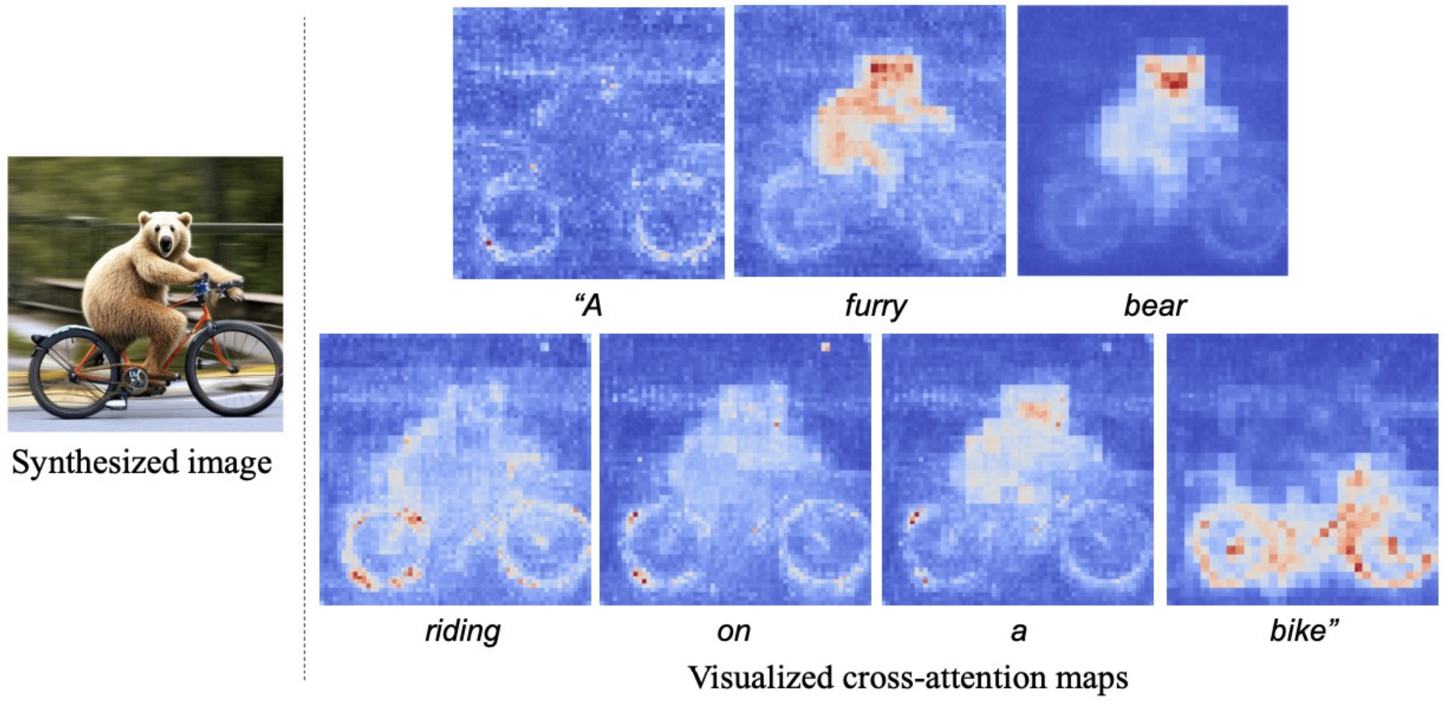


Fig. 2. Visualization of cross-attention maps estimated by Stable Diffusion. The redder pixels indicate stronger relationships with each word. The network attends to different pixels for each word.

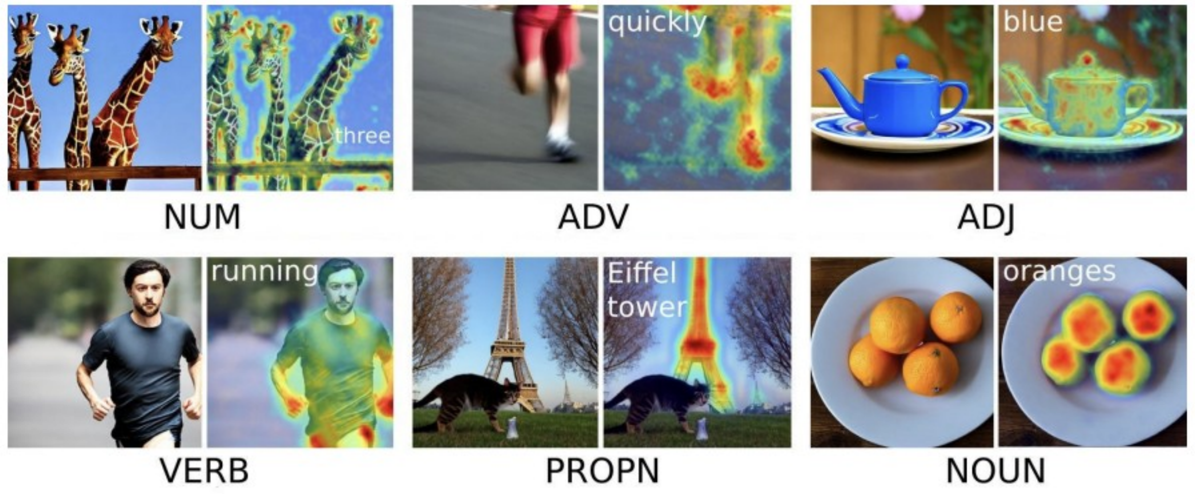
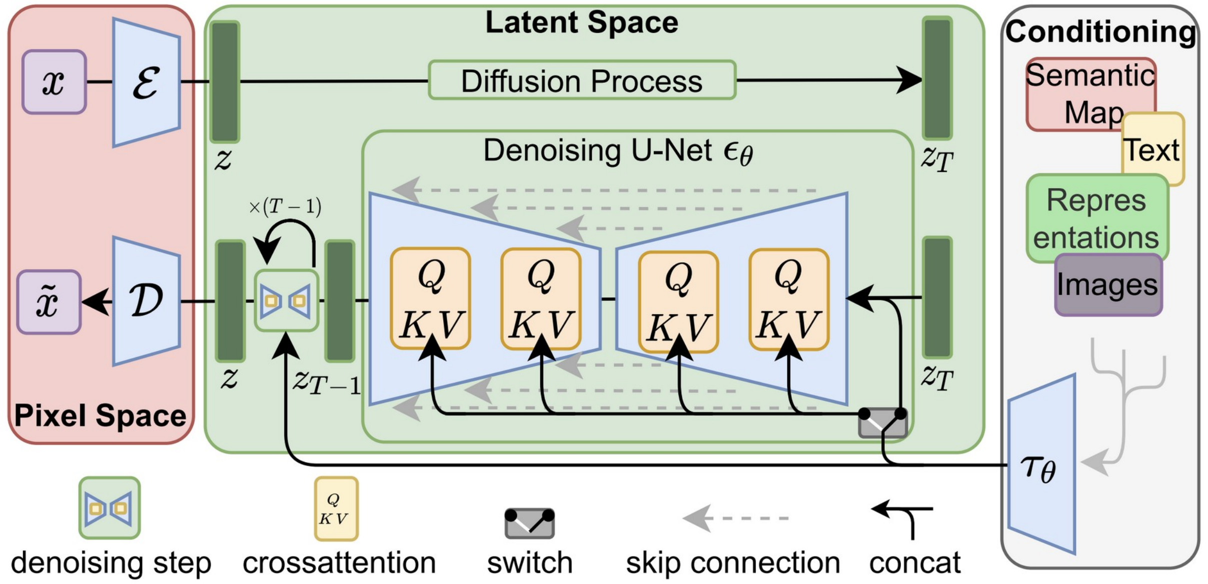


Figure 4: Example generations and DAAM heat maps from COCO for each interpretable part-of-speech.

Stable Diffusion

From the developer’s GitHub

*<u>Stable Diffusion</u> is a **latent text-to-image diffusion** model. Thanks to a generous compute donation from <u>Stability AI</u> and support from <u>LAION</u>, we were able to train a Latent Diffusion Model on 512x512 images from a subset of the <u>LAION-5B</u> database. Similar to Google's <u>Imagen</u>, this model uses a frozen **CLIP ViT-L/14 text encoder** to condition the model on text prompts. With its **860M UNet and 123M text encoder**, the model is relatively lightweight and runs on a GPU with at least 10GB VRAM.*



Text-Image Embeddings: CLIP

Dual Encodings Text and Images – Why?

In many scenarios, a multi modal interaction is very powerful

This is true not just for the conditional diffusion in LDM but in general Computer Vision tasks.

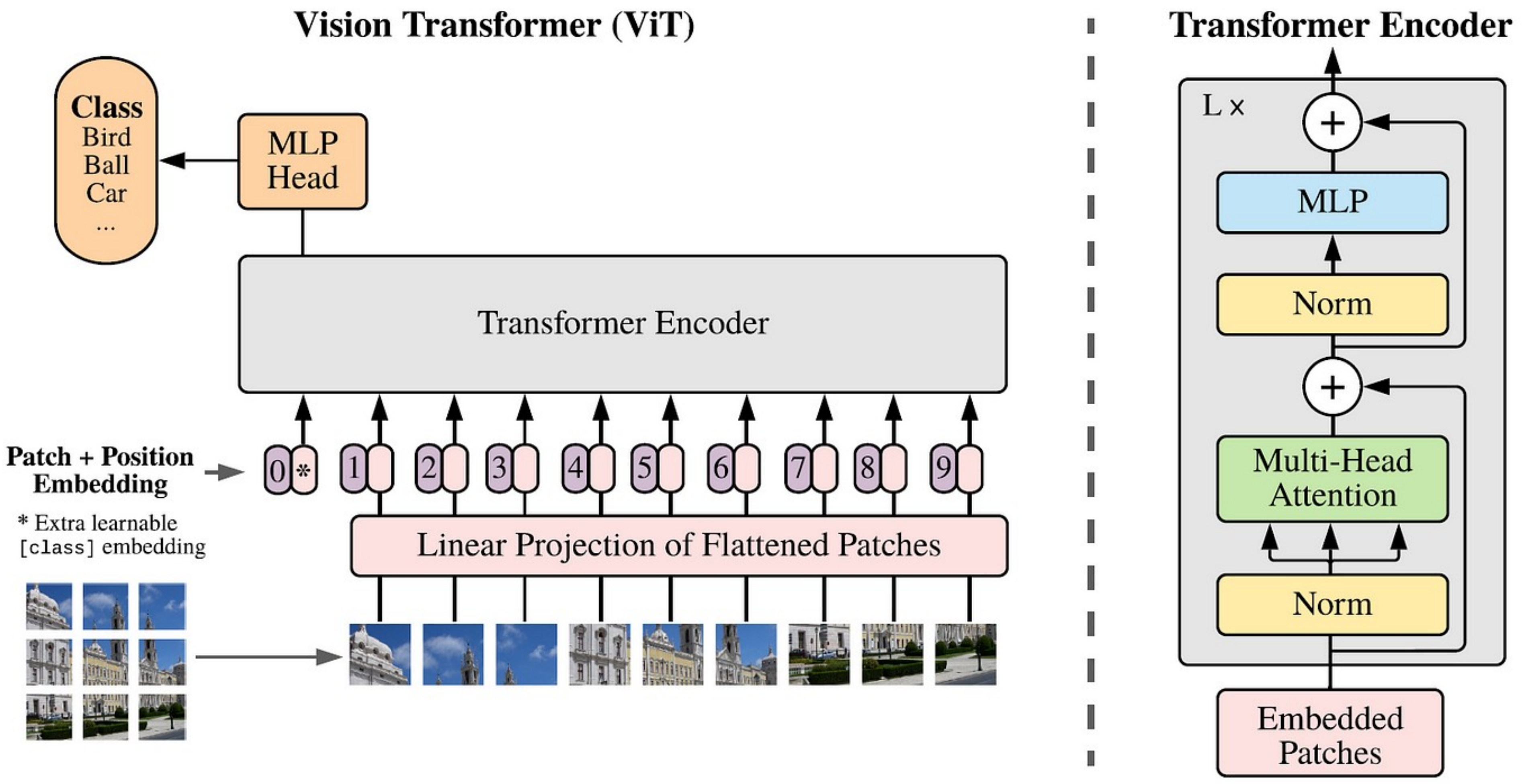
But how do we combine them?

Text Encodings

Models like BERT and GPT can generate text embeddings

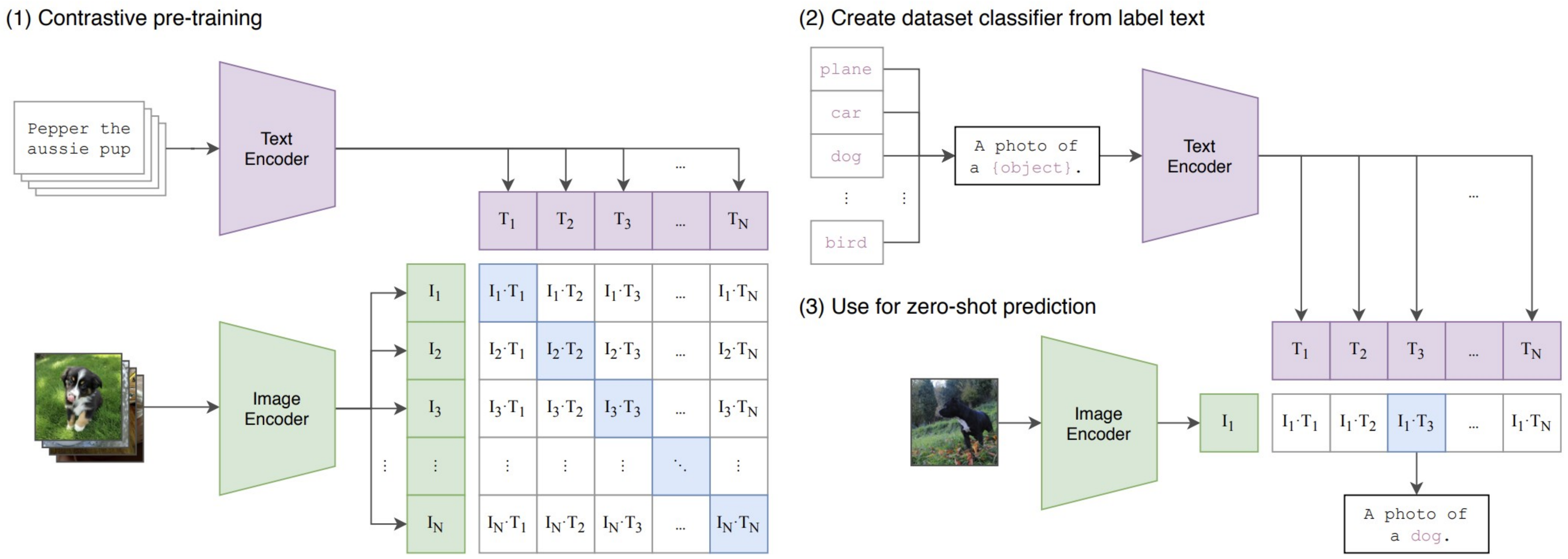
Image Encodings

- Convolutional Neural Networks like ResNet can generate a low dimensional representation
- Vision Transformers can generate image patch encodings with attention layers



Contrastive Language-Image Pretrained (CLIP)

- The Contrastive Language-Image Pretrained (CLIP) Model was developed in 2021 by OpenAI
- This model utilizes a text LLM such as GPT and BERT to generate text caption embeddings
- These text captions are paired with images that are sent through a Vision Transformer for a shared embedding space



Contrastive Language-Image Pretrained (CLIP) - Training

Contrastive Learning

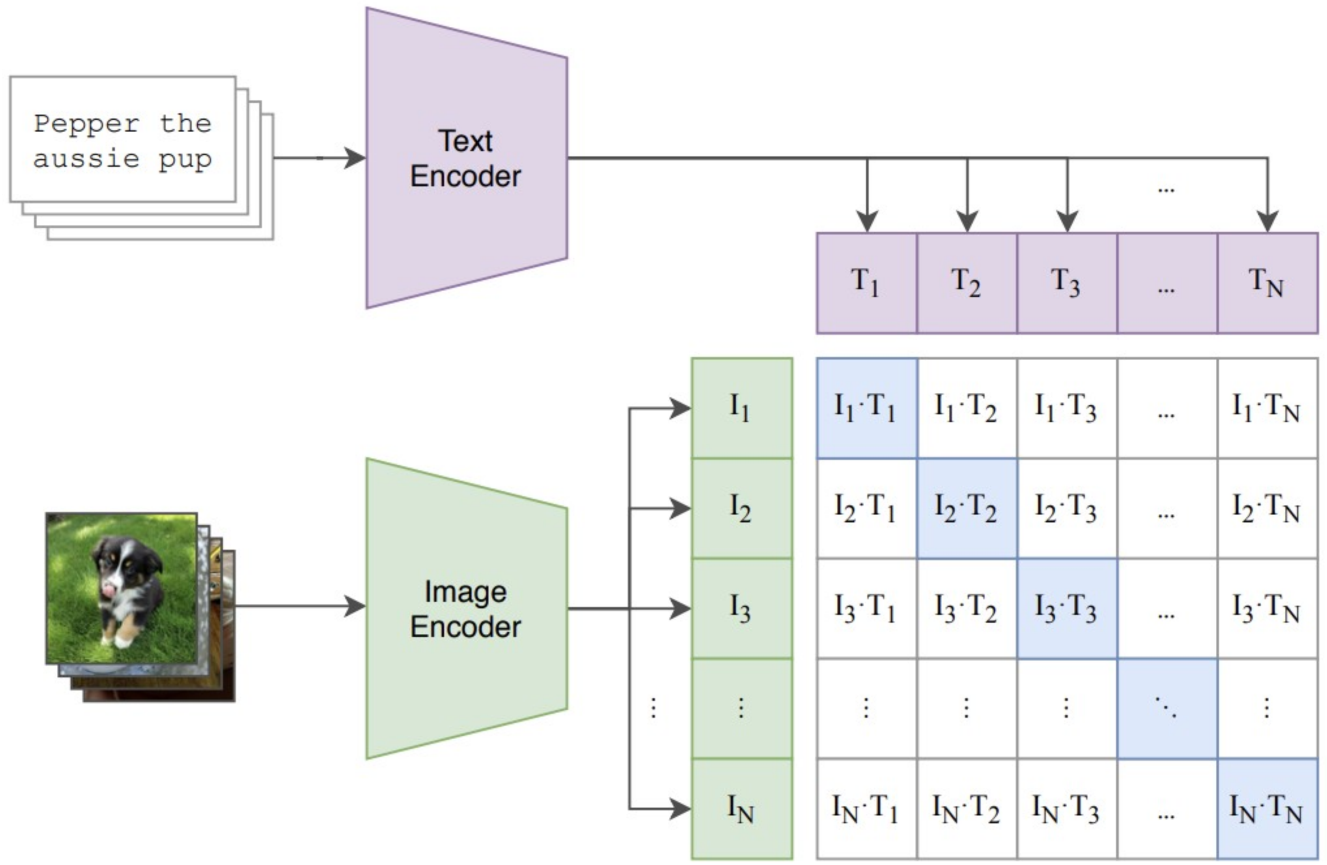
- The primary objective is to learn embeddings where matching text-image pairs are close together, and non-matching pairs are far apart.

Positive and Negative Pairs:

- Positive Pairs: Corresponding text and image pairs (e.g., an image of a cat and the caption "a cat").
- Negative Pairs: Randomly selected non-matching text and image pairs (e.g., an image of a cat and the caption "a dog").

Contrastive Loss Function:

- The model uses a contrastive loss function, such as InfoNCE (Information Noise-Contrastive Estimation), to optimize the alignment of embeddings.
- The loss function encourages the embeddings of positive pairs to be close and those of negative pairs to be distant.



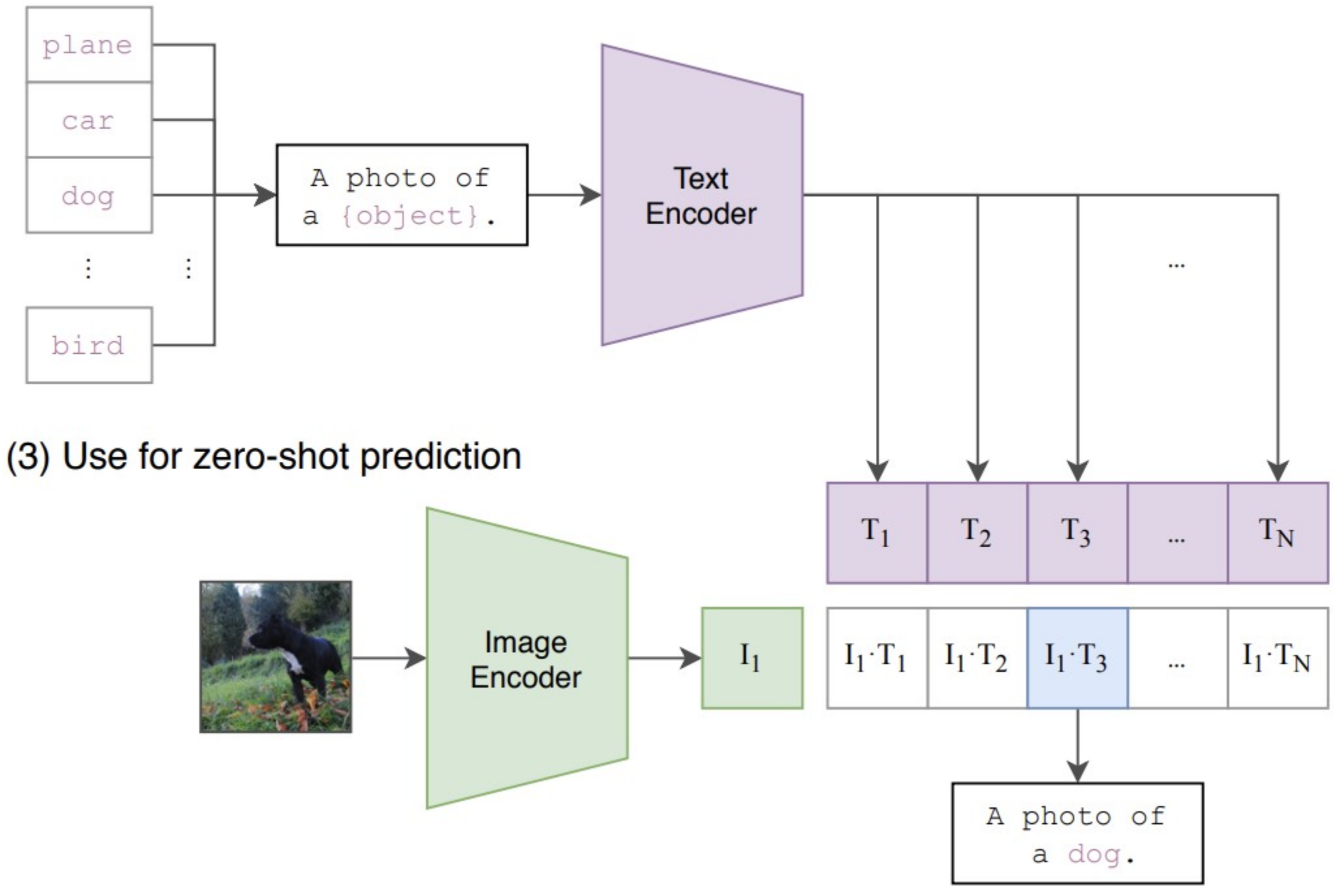
Contrastive Language-Image Pretrained (CLIP) - Inference

Zero-Shot Inference

Primarily, CLIP models are used for zero shot predictions of image descriptions

CLIP models do have some limitations, notably, they do not perform well on out of sample images, even struggling with the MNIST handwritten dataset.

Nonetheless, CLIP models, and variants are used extensively in diffusion models as a means to introduce text conditional embedded information to control image generation.



Current State of the Art and the Future

Comparing Diffusion Models

Feature/Aspect	Stable Diffusion	DALL-E	Imagen
Architecture	Latent Diffusion Model (LDM)	Transformer-based	VQ-VAE
Parameters	Approximately 1.3 billion	Approximately 3.5 billion	Not publicly disclosed (likely high)
Image Quality	High	Very High	Very High
Realism	High	High	Very High
Text-Image Alignment	Strong	Very Strong	Very Strong
Training Dataset	400 million image-text pairs	Not publicly disclosed	JFT-300M (300 million images)
Training Techniques	Advanced noise scheduling	optimized sampling	Sparse attention
Computational Efficiency	High	Moderate	Moderate
Flexibility	High	High	Moderate
Scalability	High	Moderate	Moderate
Zero-Shot Learning	Strong generalization to new prompts	Excellent zero-shot capabilities	Strong zero-shot capabilities
Use Cases	Text-to-image synthesis	image inpainting	super-resolution
Notable Strengths	Efficient and scalable	high-quality outputs	versatile
Availability	Open-source (various implementations)	Limited access	not fully open-source

Video Diffusion Models

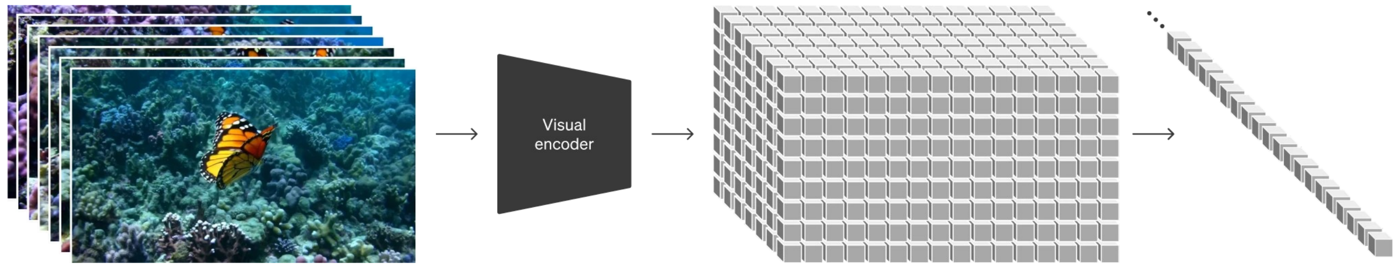
Open AI announced Sora, a video generation model, signaling the next stage of diffusion generative AI…

Prompt: *The camera follows behind a white vintage SUV with a black roof rack as it speeds up a steep dirt road surrounded by pine trees on a steep mountain slope, dust kicks up from it's tires, the sunlight shines on the SUV as it speeds along the dirt road, casting a warm glow over the scene. The dirt road curves gently into the distance, with no other cars or vehicles in sight. The trees on either side of the road are redwoods, with patches of greenery scattered throughout. The car is seen from the rear following the curve with ease, making it seem as if it is on a rugged drive through the rugged terrain. The dirt road itself is surrounded by steep hills and mountains, with a clear blue sky above with wispy clouds.*

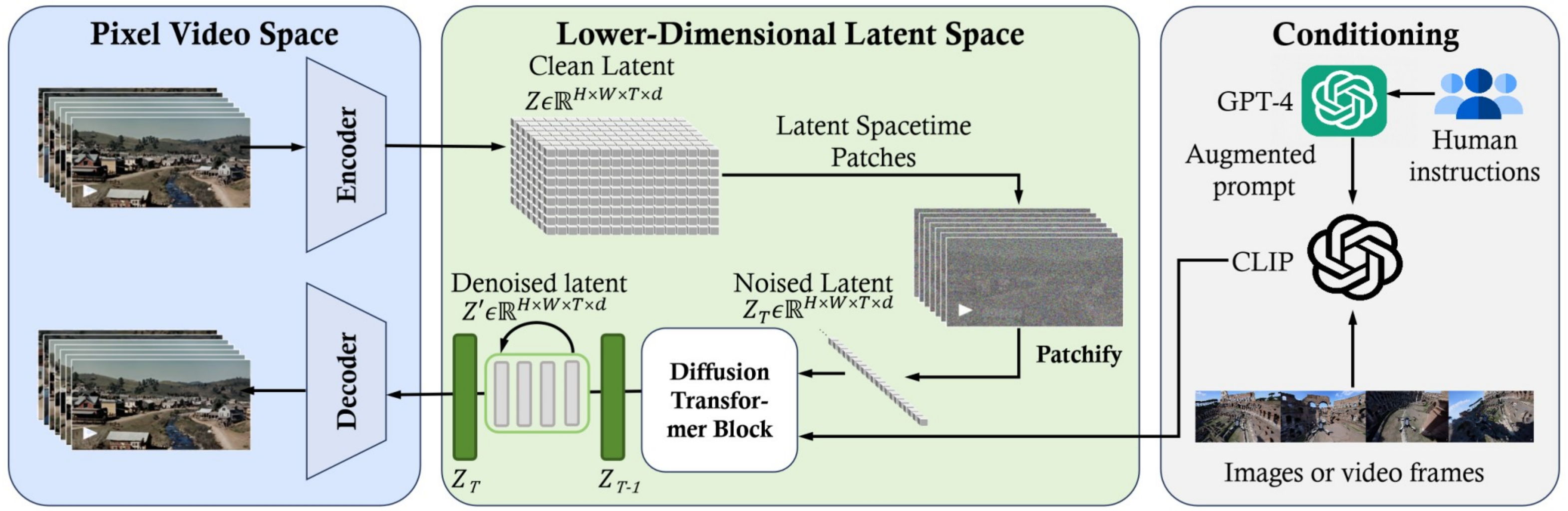
Video Diffusion Models

Current Models (mid 2024)

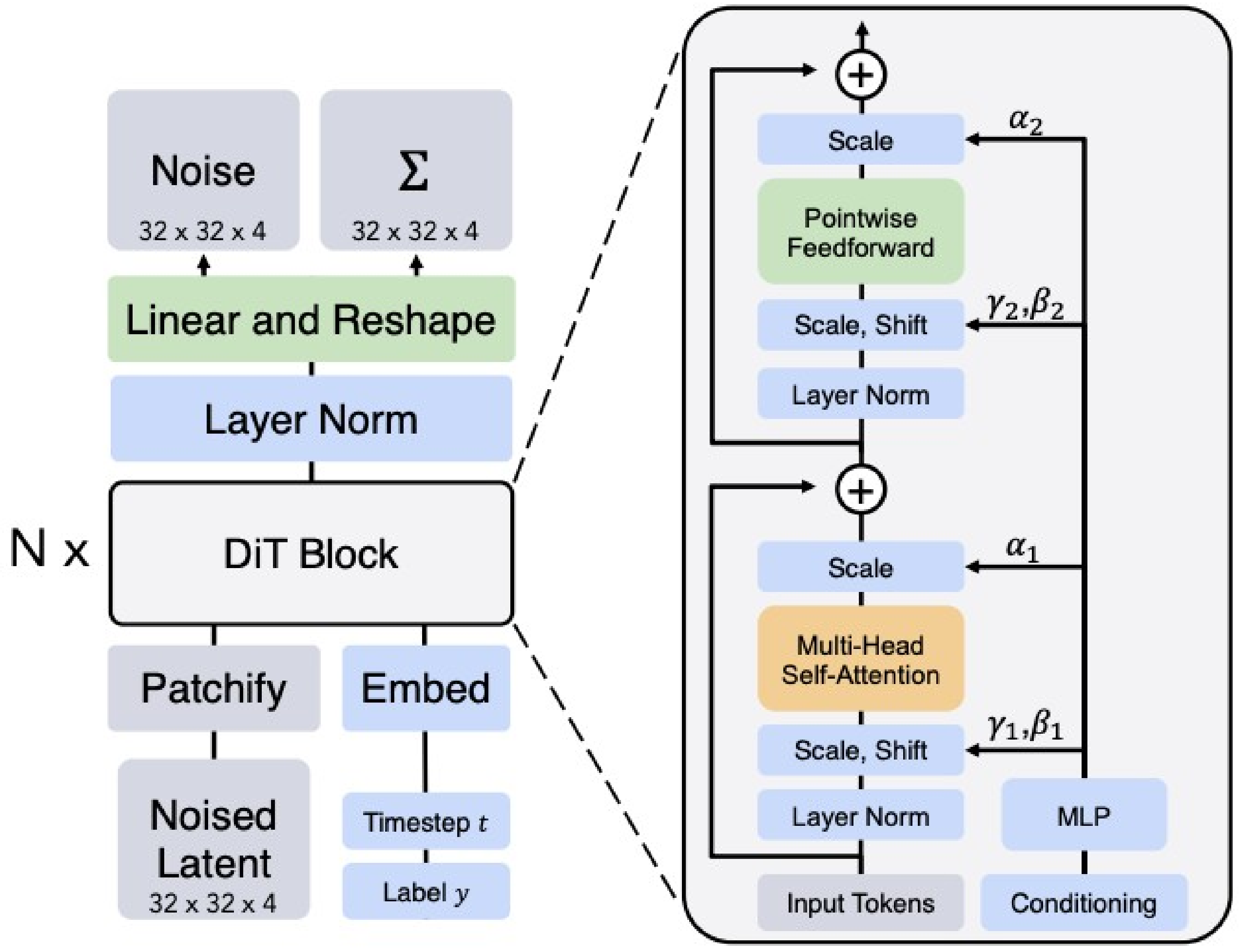
- Sora
- OpenSora
- Runway Gen-2
- Pika



Sora/OpenSora



Diffusion Transformer



Latent Diffusion Transformer

DiT Block with adaLN-Zero

Working with Diffusion Models

Hugging Face Diffusers

Similar to LLMs, Hugging Face provides an API to interact with open-source diffusion model.

Image generation from text:

```
>>> from diffusers import DiffusionPipeline

>>> pipeline = DiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5", use_safetensors=True)
```

Image generation from another image:

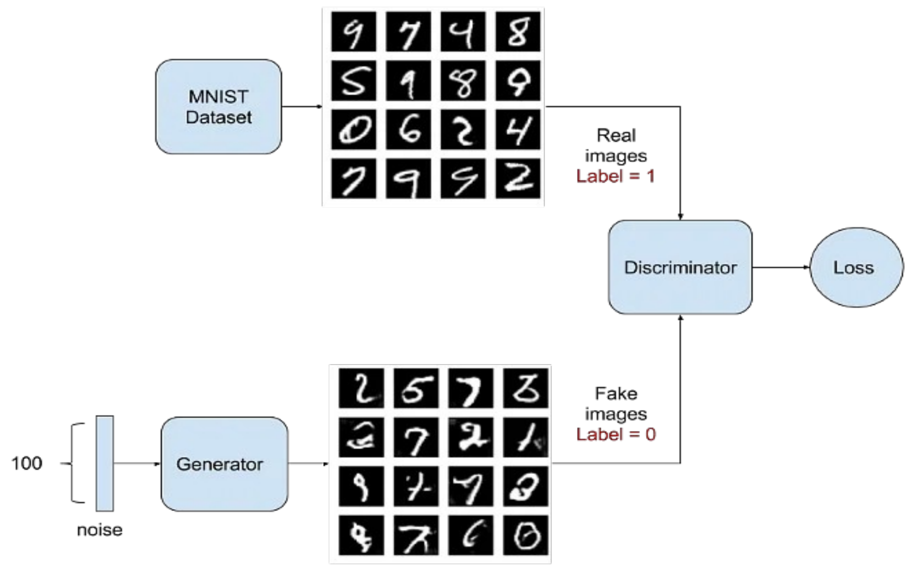
```
import torch
from diffusers import AutoPipelineForImage2Image
from diffusers.utils import load_image, make_image_grid

pipeline = AutoPipelineForImage2Image.from_pretrained(
    "kandinsky-community/kandinsky-2-2-decoder",
    torch_dtype=torch.float16,
    use_safetensors=True,
)
pipeline.enable_model_cpu_offload()
# remove following line if xFormers is not installed or you have PyTorch 2.0 or higher installed
pipeline.enable_xformers_memory_efficient_attention()

# Load an image to pass to the pipeline:
init_image = load_image(
    "https://huggingface.co/datasets/huggingface/documentation-images/resolve/main/diffusers/cat.png"
)

# Pass a prompt and image to the pipeline to generate an image:
prompt = "cat wizard, gandalf, lord of the rings, detailed, fantasy, cute, adorable, Pixar, Disney, 8k"
image = pipeline(prompt, image=init_image).images[0]
make_image_grid([init_image, image], rows=1, cols=2)
```

```
>>> pipeline
StableDiffusionPipeline {
  "_class_name": "StableDiffusionPipeline",
  "_diffusers_version": "0.21.4",
  ...,
  "scheduler": [
    "diffusers",
    "PNDMScheduler"
  ],
  ...,
  "unet": [
    "diffusers",
    "UNet2DConditionModel"
  ],
  "vae": [
    "diffusers",
    "AutoencoderKL"
  ]
}
```

Diffusion and Image Generation – Wrap Up

In this module we have introduce the generative models that focus on image generation:

- Generative Adversarial Networks

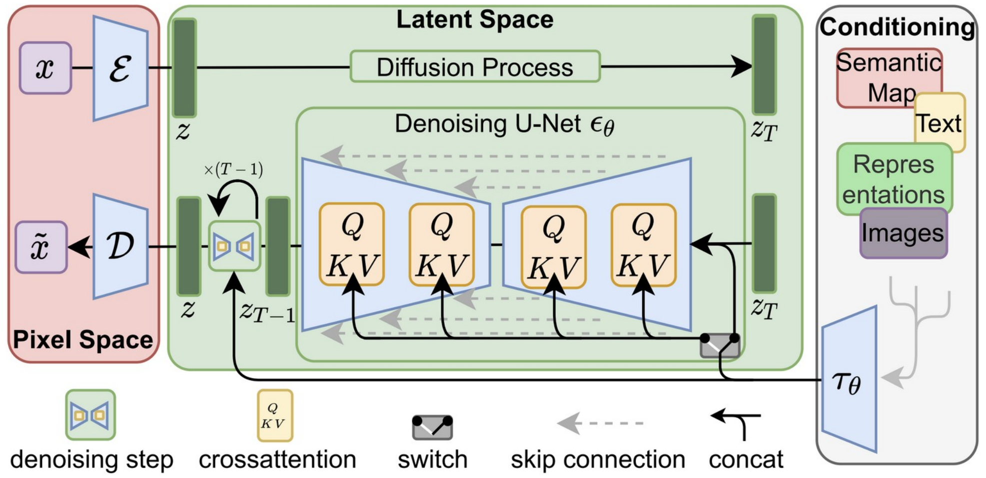


- Diffusion Models

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$$

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

- Latent Diffusion Models



Thank you!