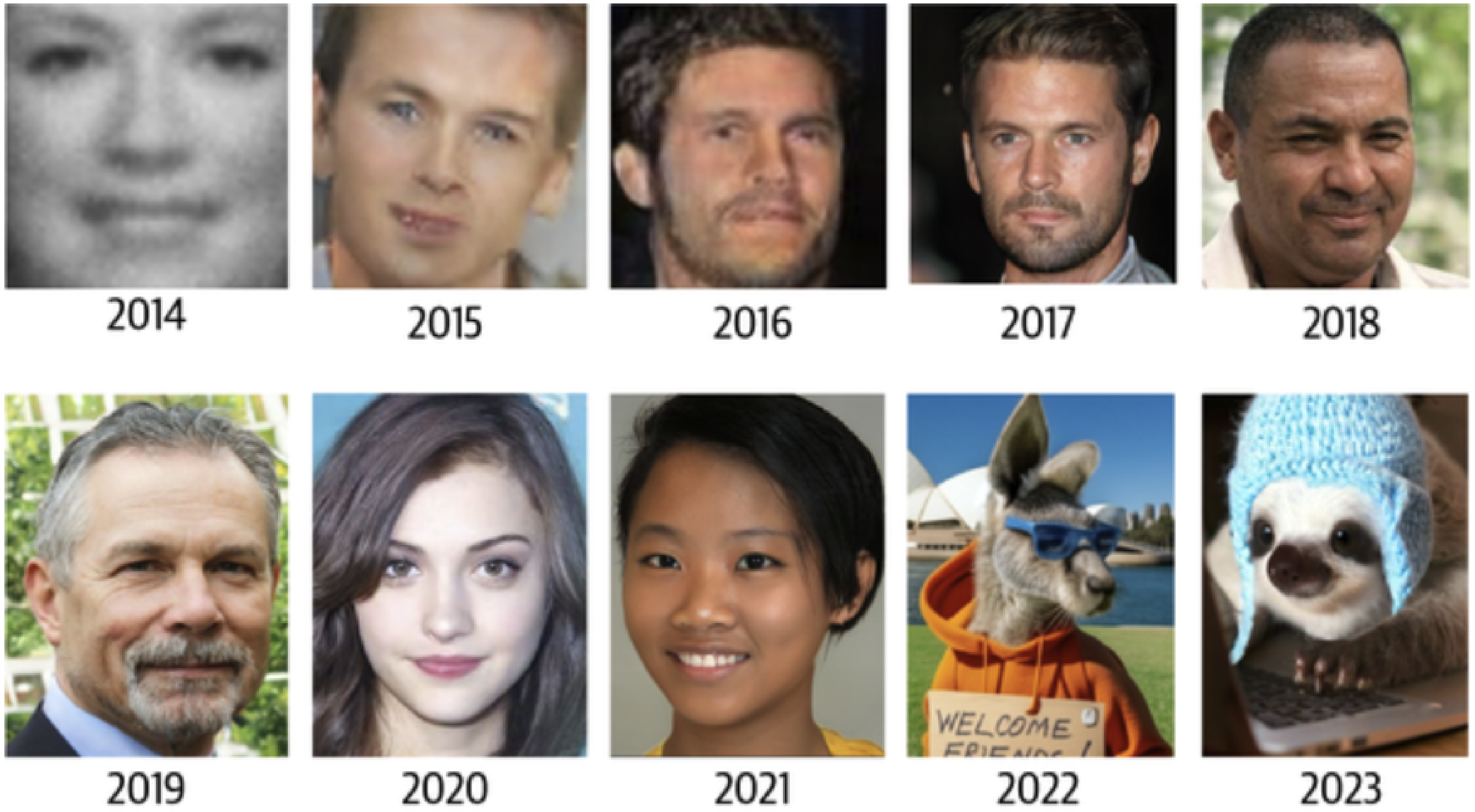
Vision Models

Philipp Koehn

17 April 2025

Image Generation



- How does this work?

Textbook

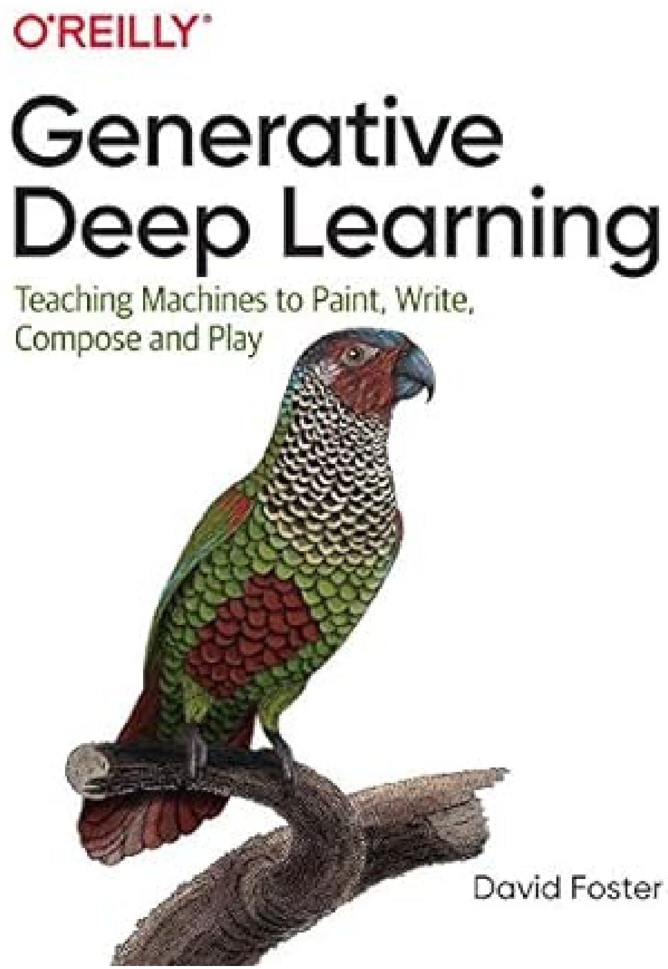
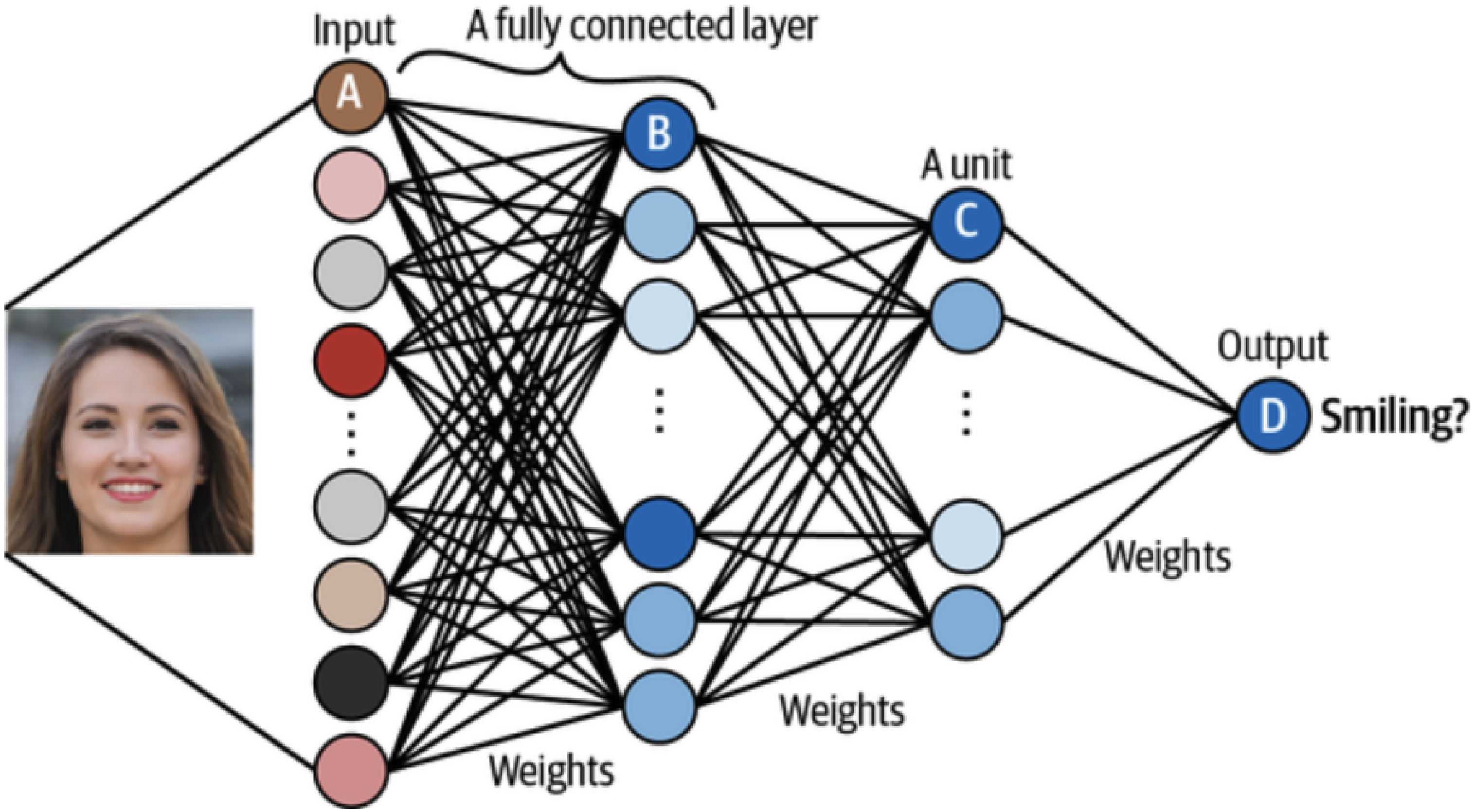

O'REILLY
Generative
Deep Learning
Teaching Machines to Paint, Write,
Compose and Play
David Foster

image classification

Image Classification

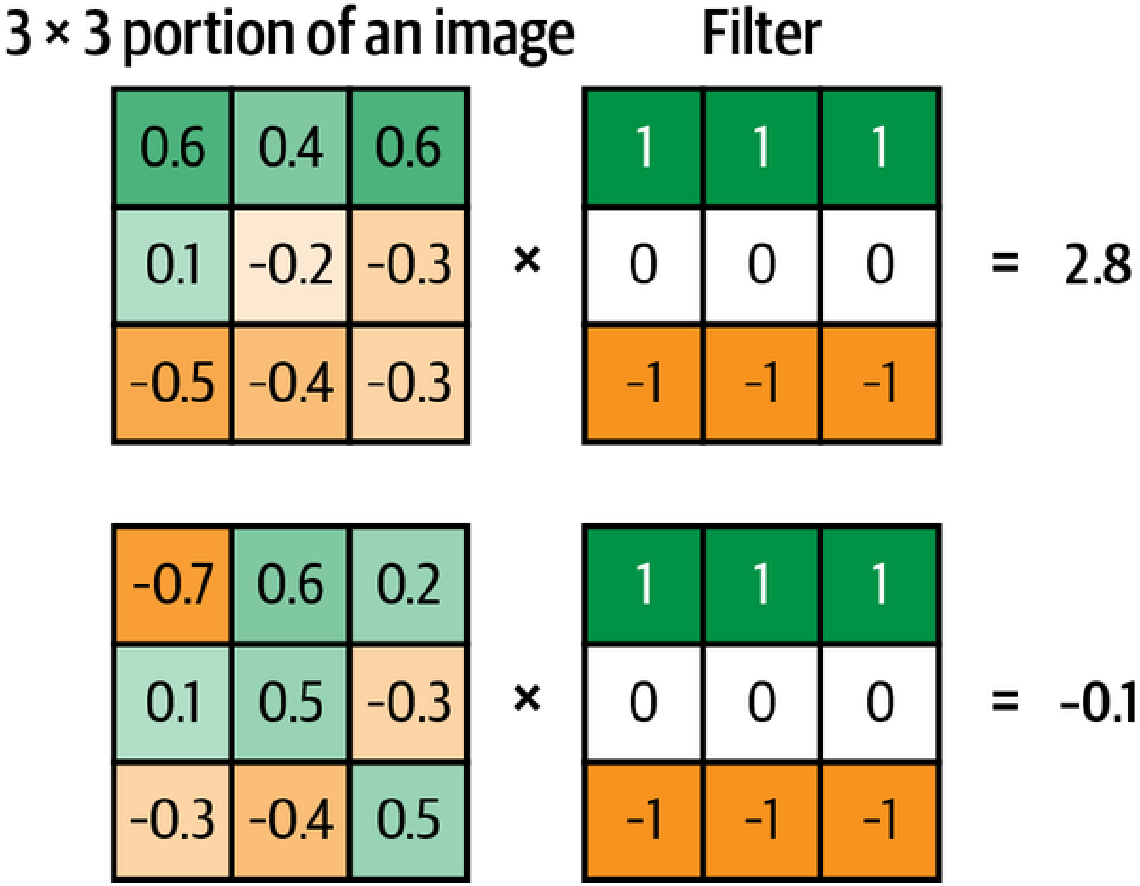


- Input: Image, Output: Class

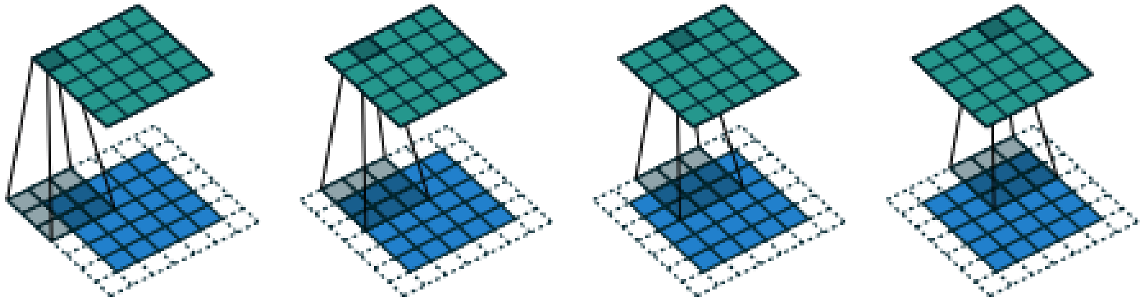
CIFAR-10

- 60,000 images
- 32x32 resolution
- 10 classes

Simple Convolution

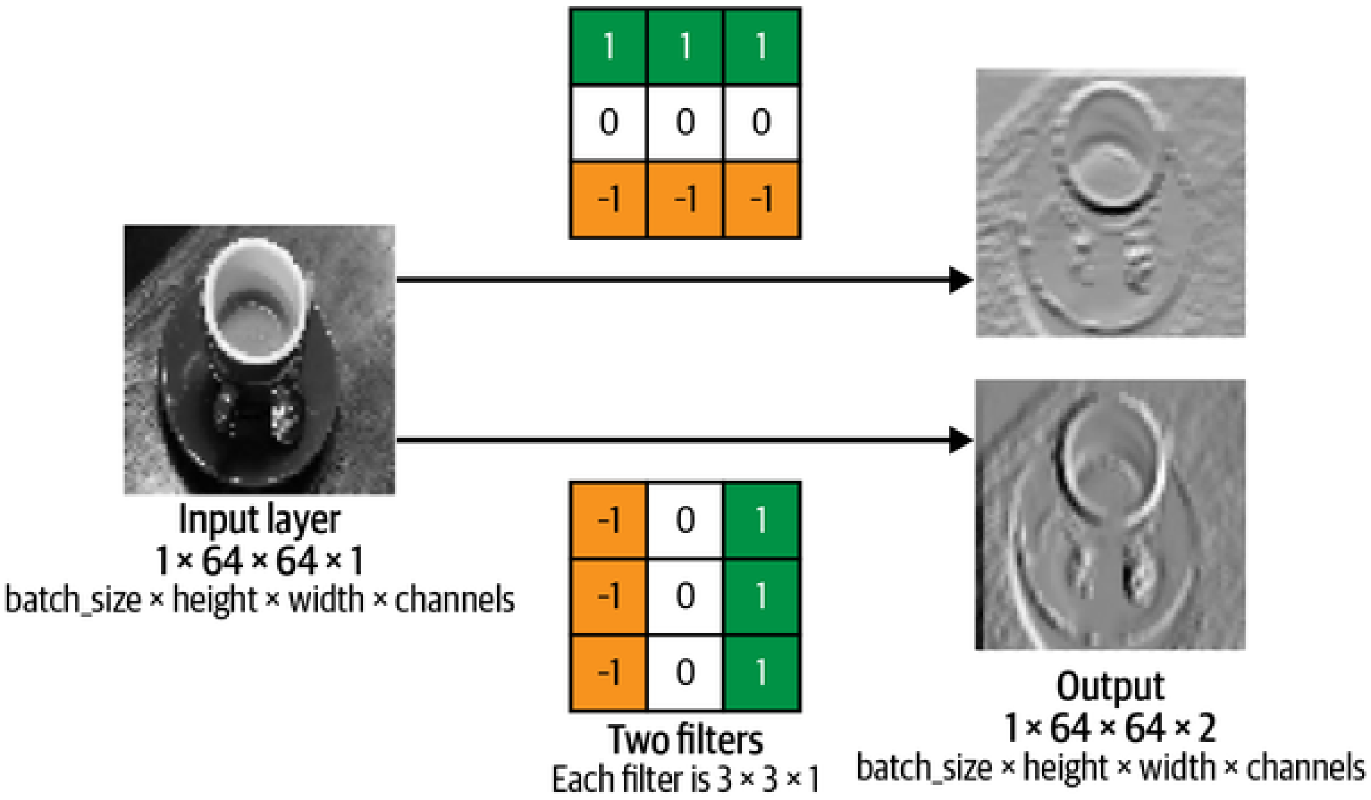


Processing Image with Convolution



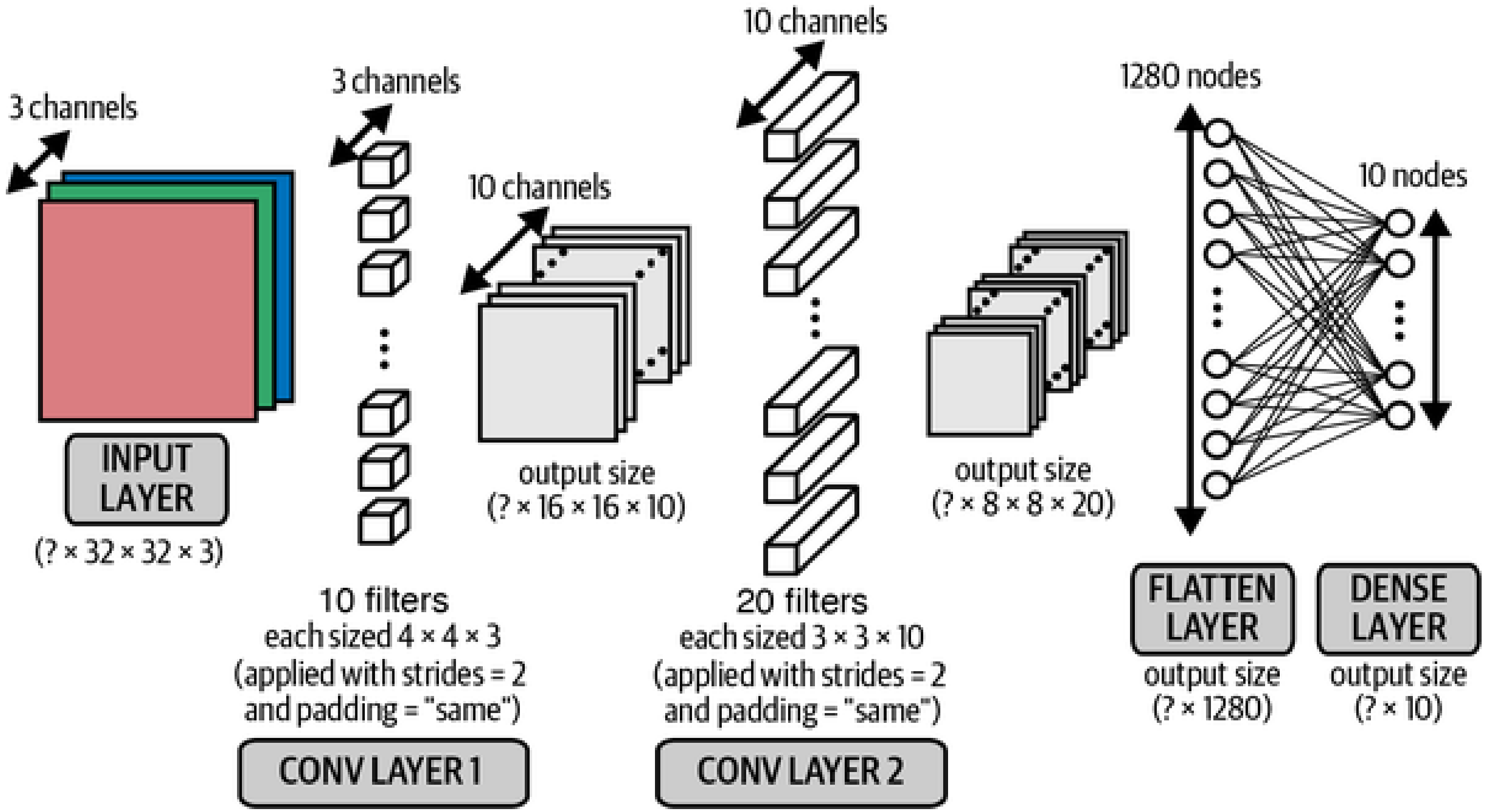
- Passing a 3x3 kernel over a 5x5 image
- Using padding to preserve size of representation

Edge Detection

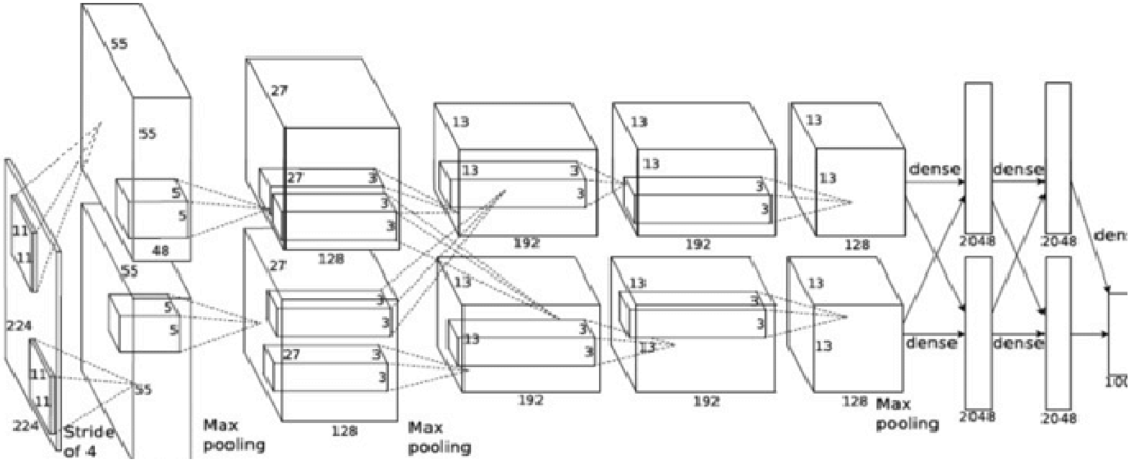


- Detecting large local color shifts: edges in image
- Two kernels: 2-dimensional vector for each point

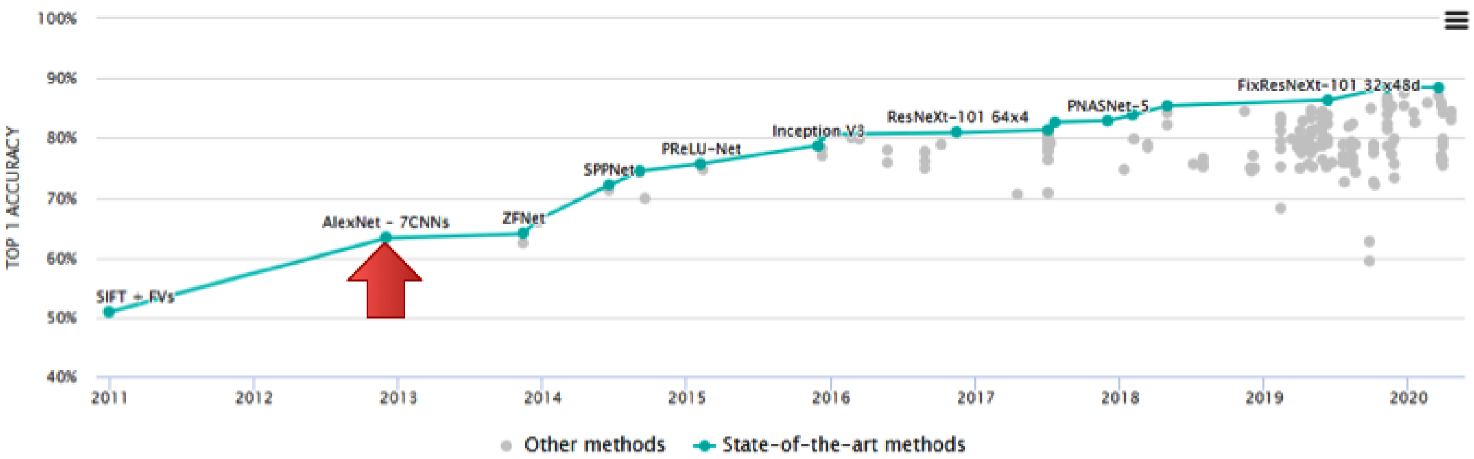
Convolutional Neural Network



ImageNet: 2012

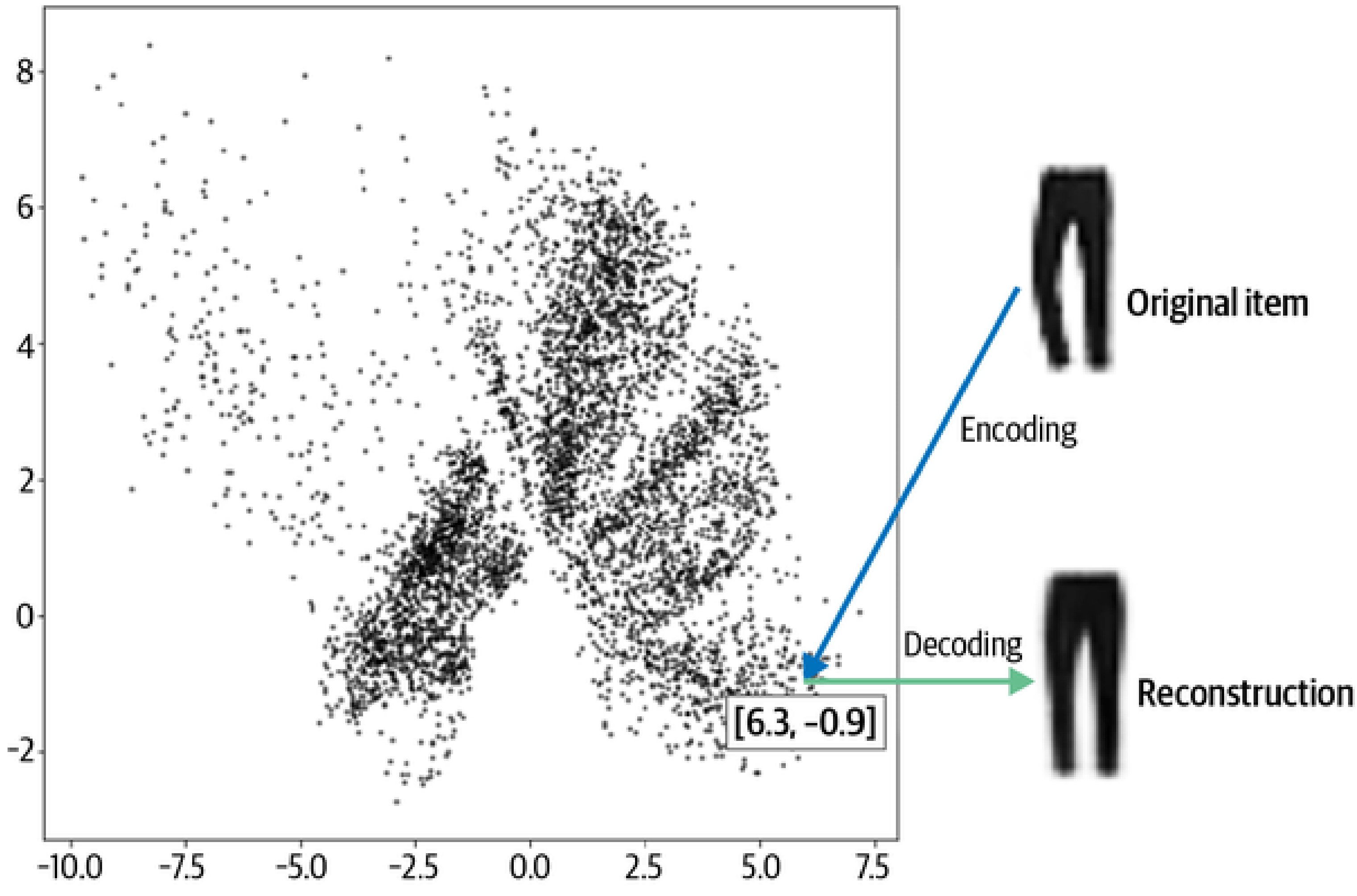


- ImageNet: Large scale competition for image classification (1.2 million labeled training examples, 1000 categories)
- Breakthrough for deep learning in 2012: large gains with CNN (AlexNet)

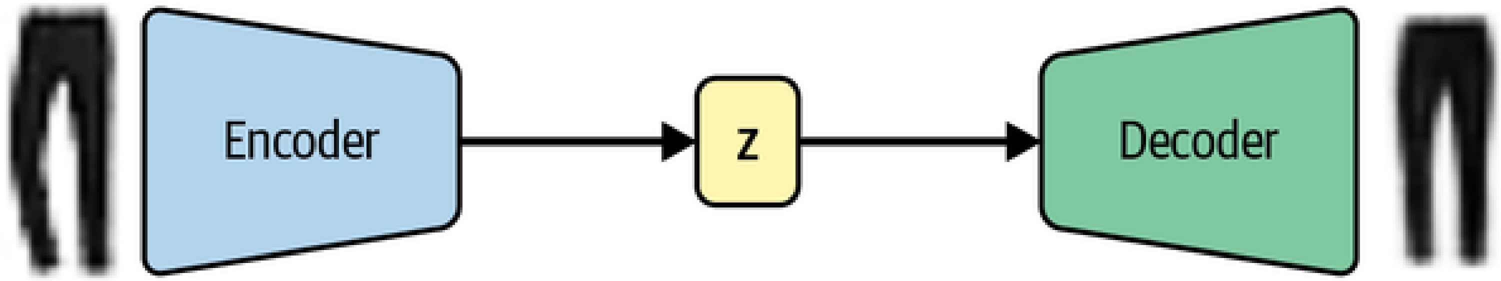


autoencoders

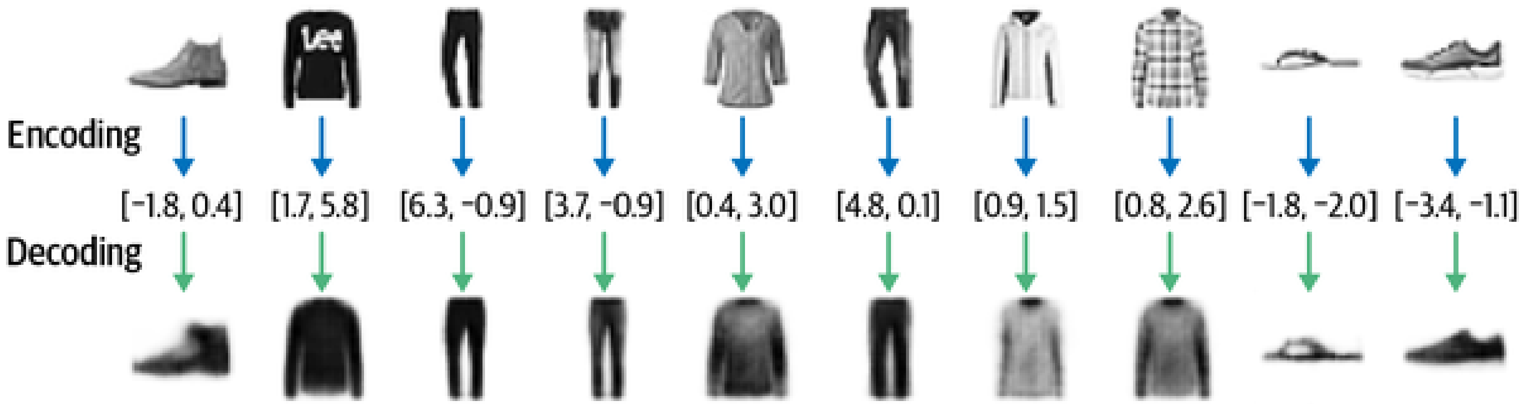
Autoencoders



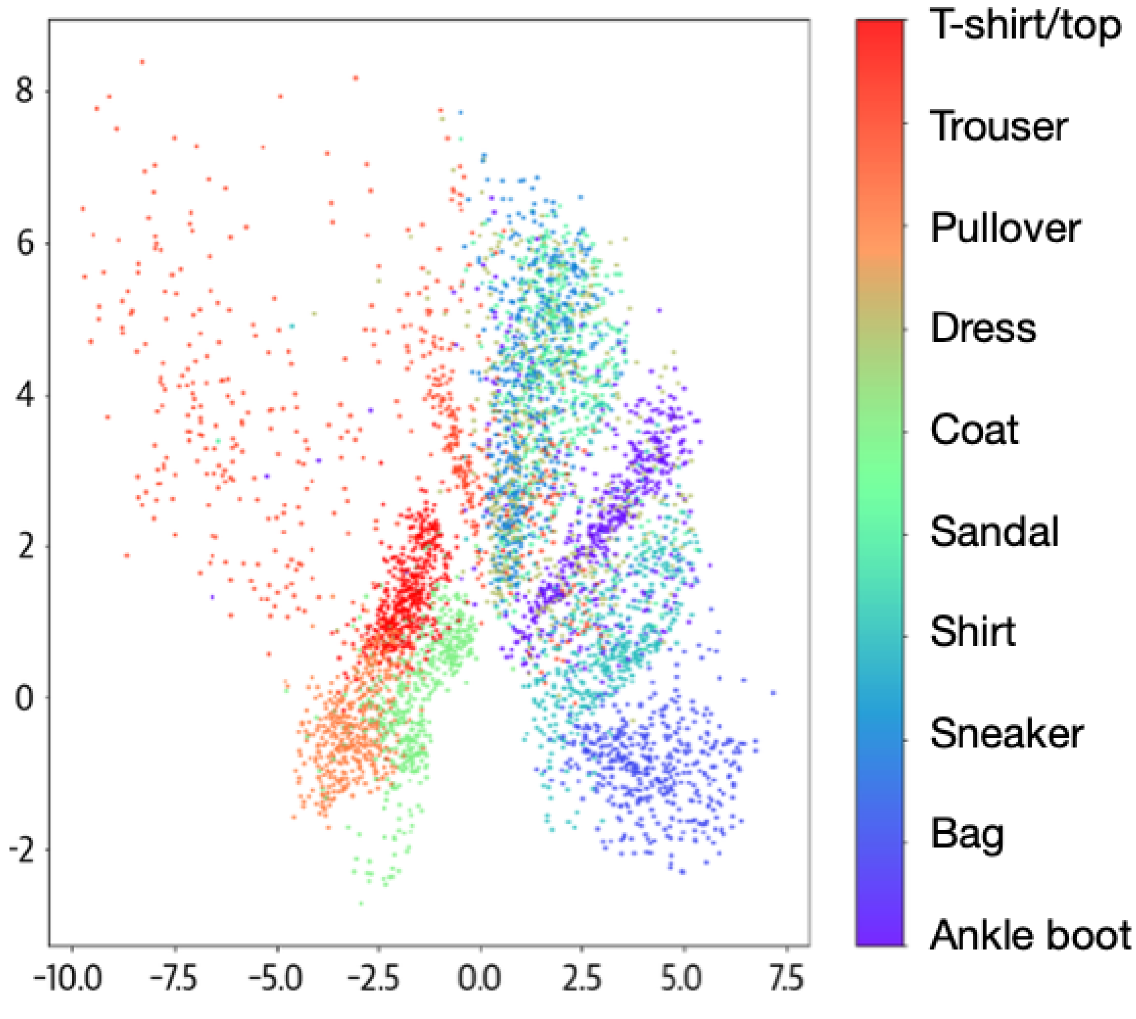
Encoder-Decoder Model



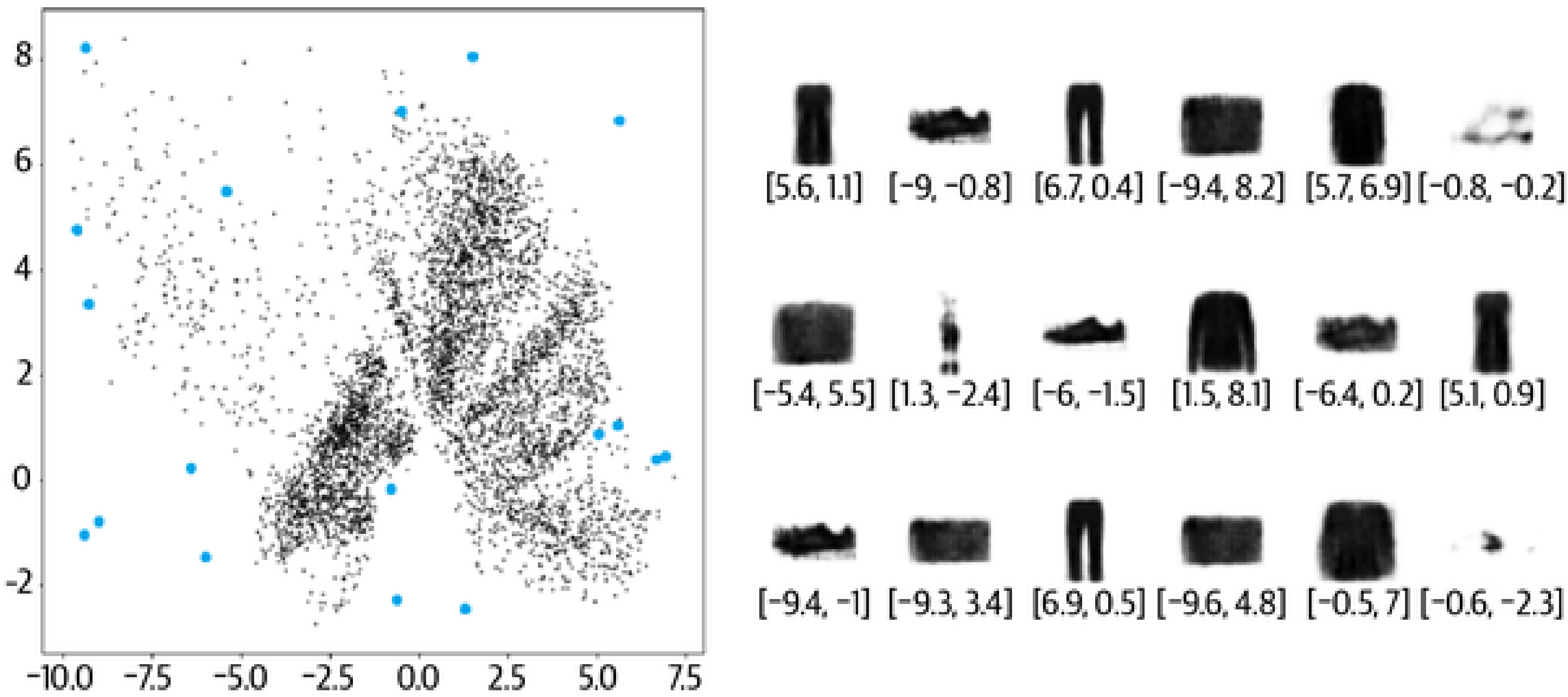
Example: Image to 2-dimensional vector



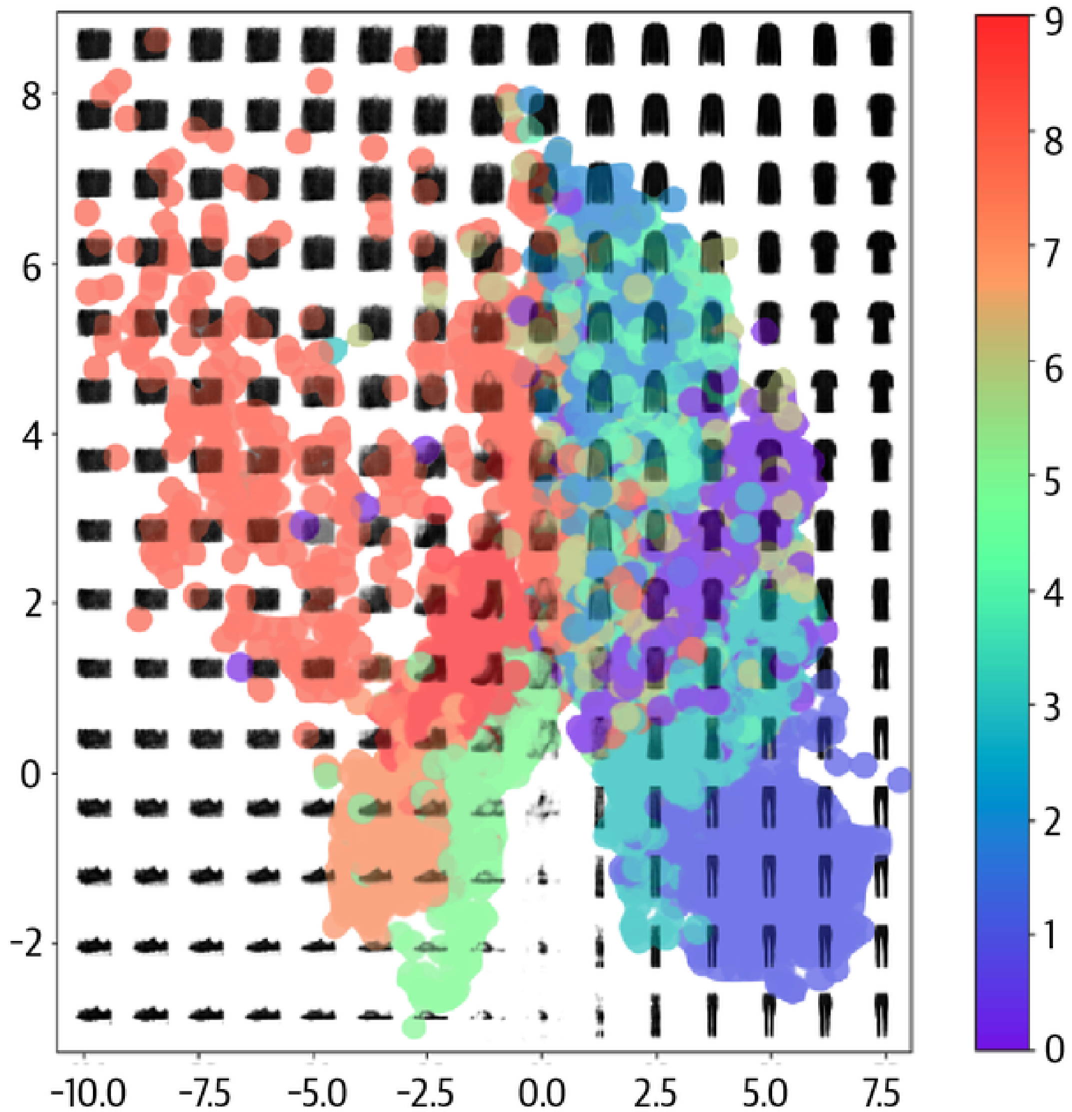
Visualization of Embeddings Space



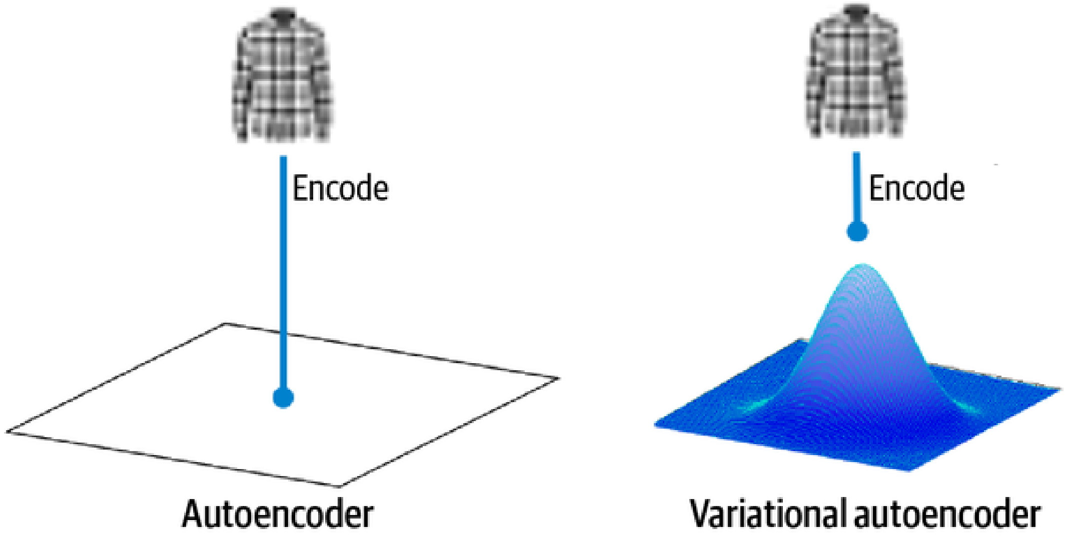
Generating Novel Items of Clothing



Generations from Full Embeddings Space



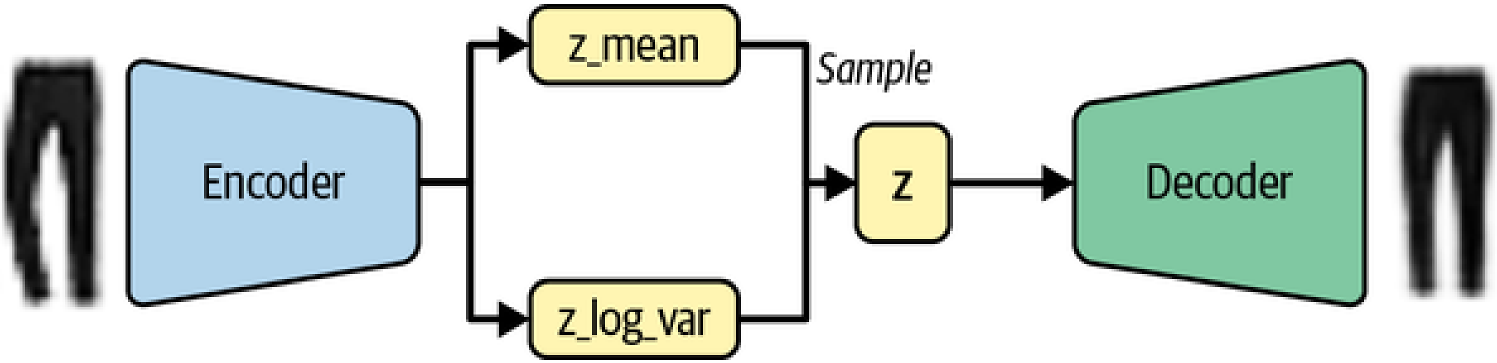
Variational Autoencoder



- Encoder predicts a normal distribution, specifically the **mean** μ and **variance** σ^2

$$f\left(x \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Variational Autoencoder



- Encoder (as before)
- Predicts mean `z_mean` and variance `z_log_var` of the distribution (note: predict 2-dimensional point $\rightarrow$ 2 means and 2 variances)
- We randomly draw a point `z` from the distribution
- Decoder predicts from that randomly drawn point `z` (as before)

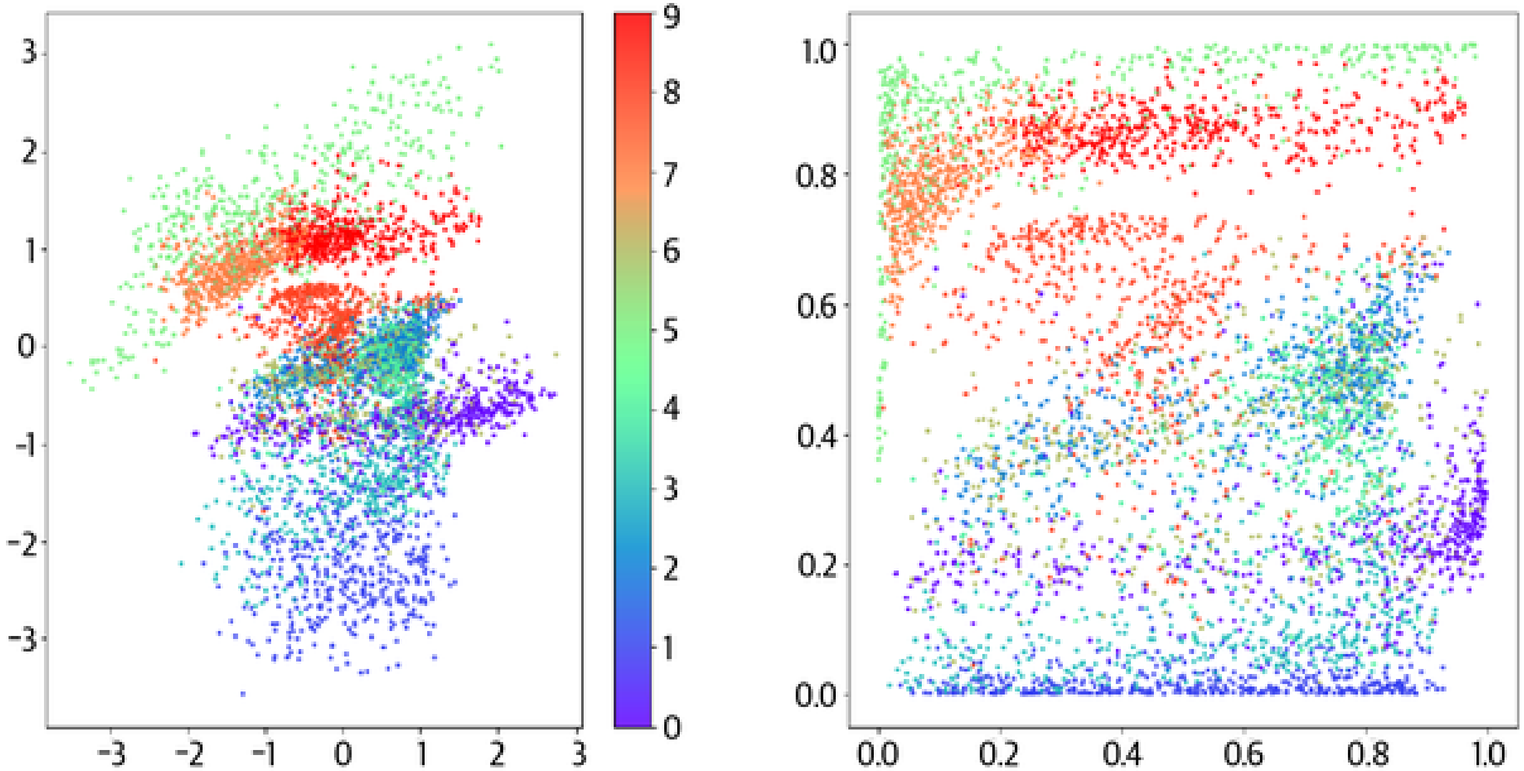
Additional Loss Term for Training

- Add preference that predicted mean and variance are normal distribution with mean 0 and variance 1: $N(0,1)$
- Computed as KL-divergence between predicted values μ and σ^2

$$\mathcal{D}_{KL}\left[N(\mu,\sigma) \| N(0,1)\right] = -\frac{1}{2}\sum\left(1 + \log\left(\sigma^2\right) - \mu^2 - \sigma^2\right)$$

What is it Good For?

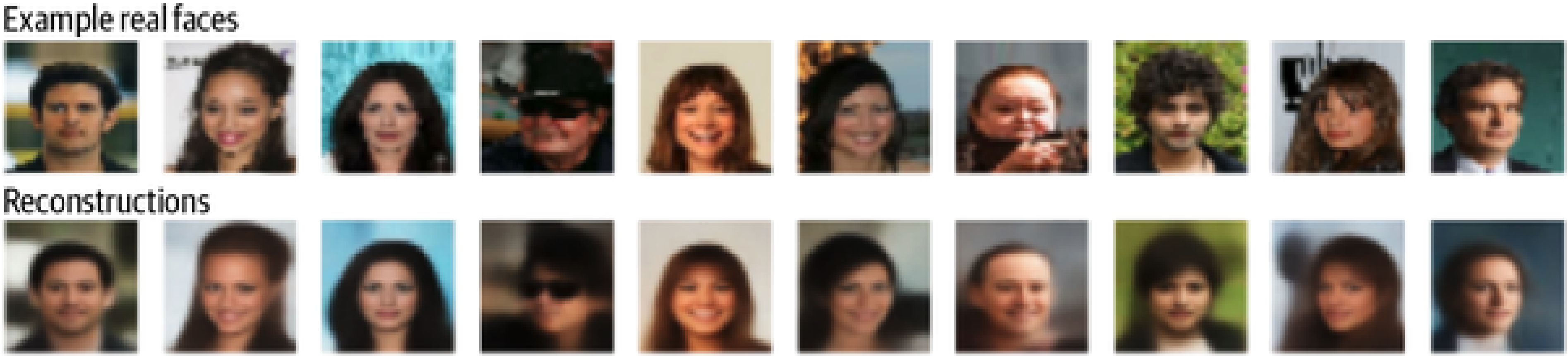
- General principle: adding noise is good (here: random sampling of point z)
- Better use of embedding space (bias towards center)



Left: `z_mean`, Right: p-values

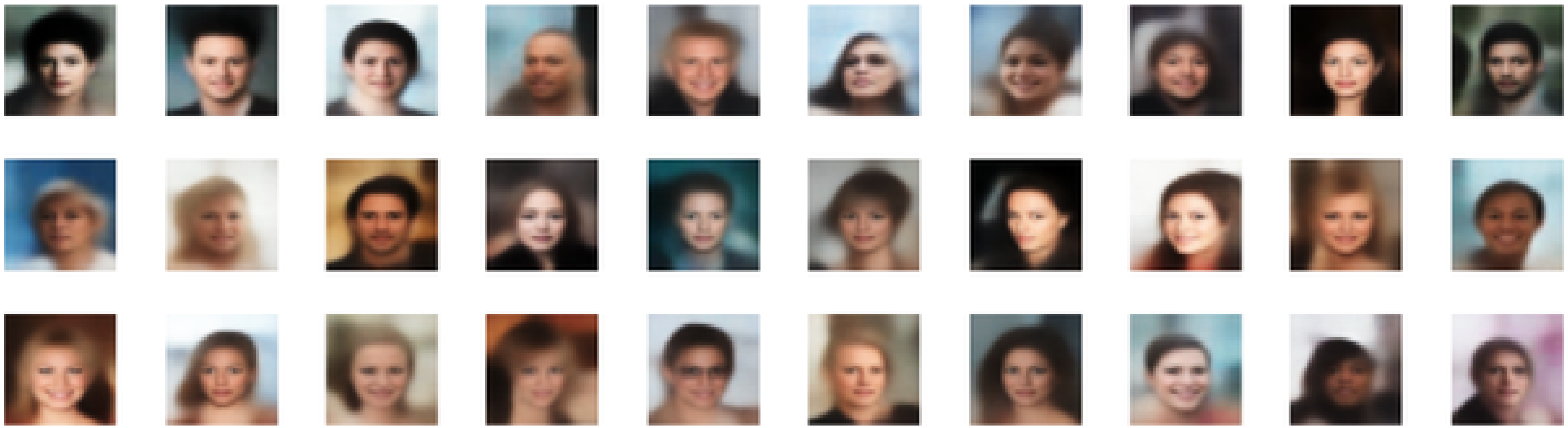
Variational Autoencoder for Images of Faces

- CelebA dataset of over 200,000 color images of celebrity faces
- We need a larger model (multiple convolutional layers, larger embedding sizes)



Generation of Faces

- Sample points (from a standard multivariate normal distribution)
- Decode the sampled points
- Plot the images



CelebA Dataset Labeled with Features

Vectors for Features

- We want to obtain a vector for one feature, e.g. `Smiling`

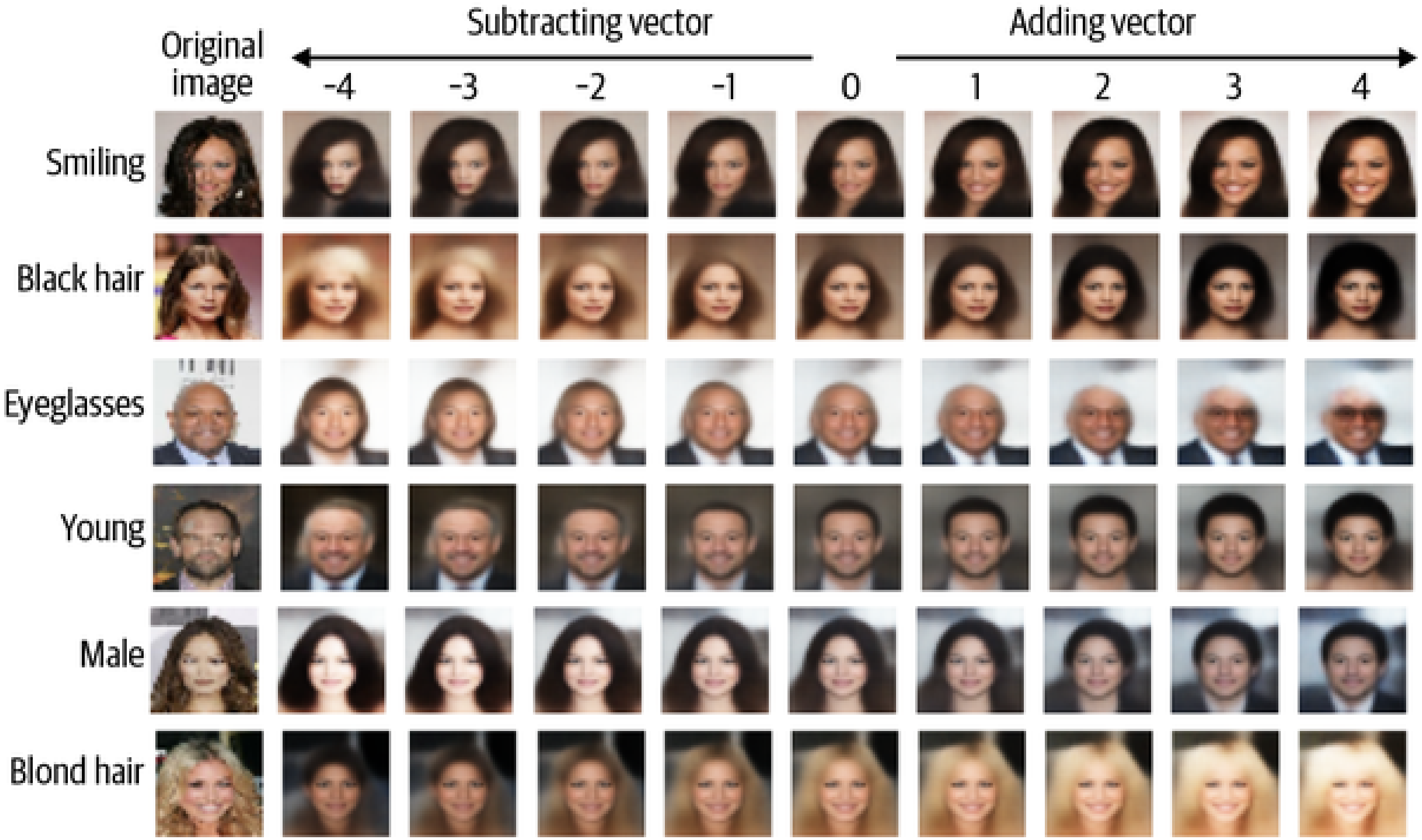
- Recall: encoder predicts vector representation for each image

- Vector that points in the direction of `Smiling`
  - take average of all vectors for images labelled `Smiling`: a_{smiling}
  - take average of all vectors for images not labelled `Smiling`: $a_{\overline{\text{smiling}}}$
  - subtract the two vectors: $v_{\text{smiling}} = a_{\text{smiling}} - a_{\overline{\text{smiling}}}$

Generation with Feature Vector

- Randomly sample point z (or use representation for existing image)

- Subtract and feature vector v_{smiling} with varying factor, generate



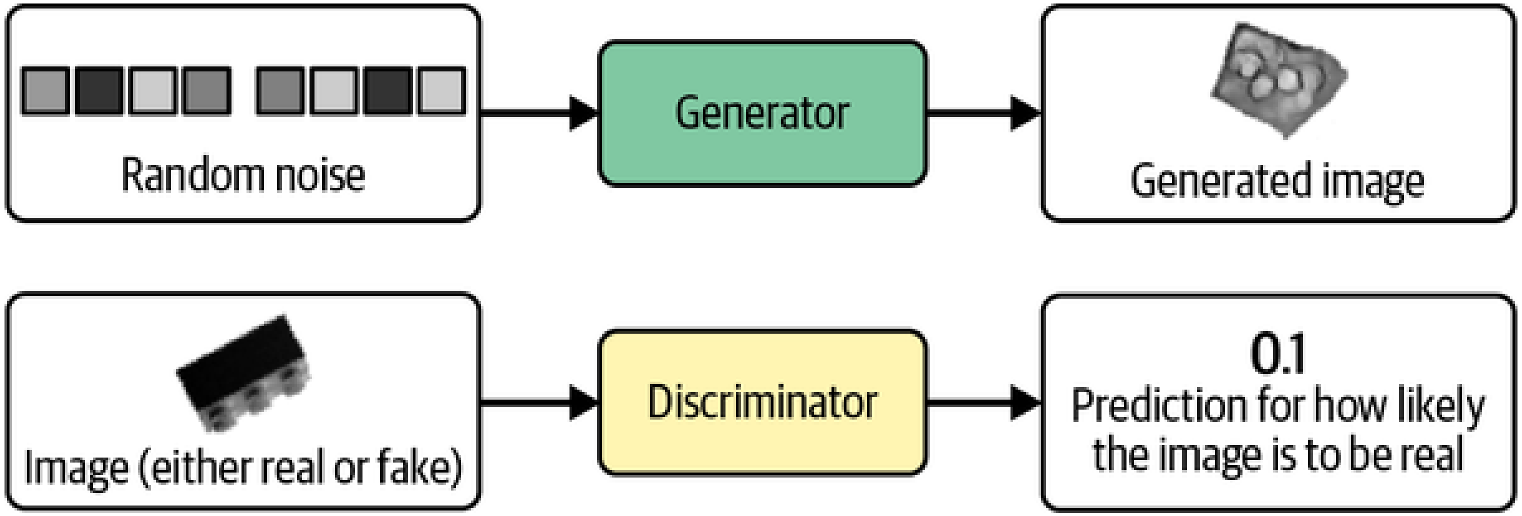
Combine Images

- Take representations of two images: z_A and z_B
- Combine them with weight $\alpha \in [0, 1]$ into new vector $z = \alpha z_A + (1 - \alpha) z_B$

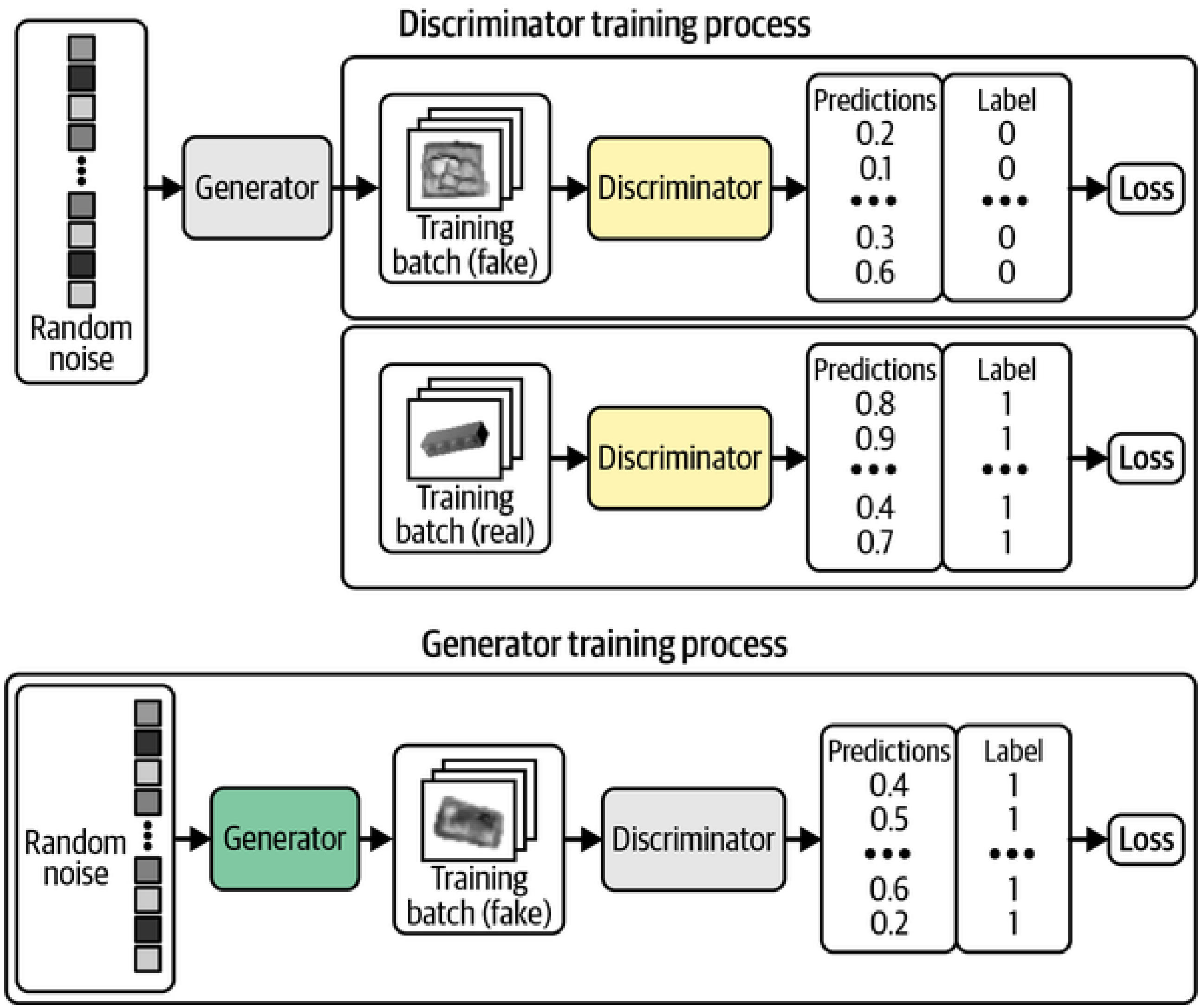
generative adversarial training

Discriminator

- The story so far: we can generate novel images
- New task: detect if image is generated or real



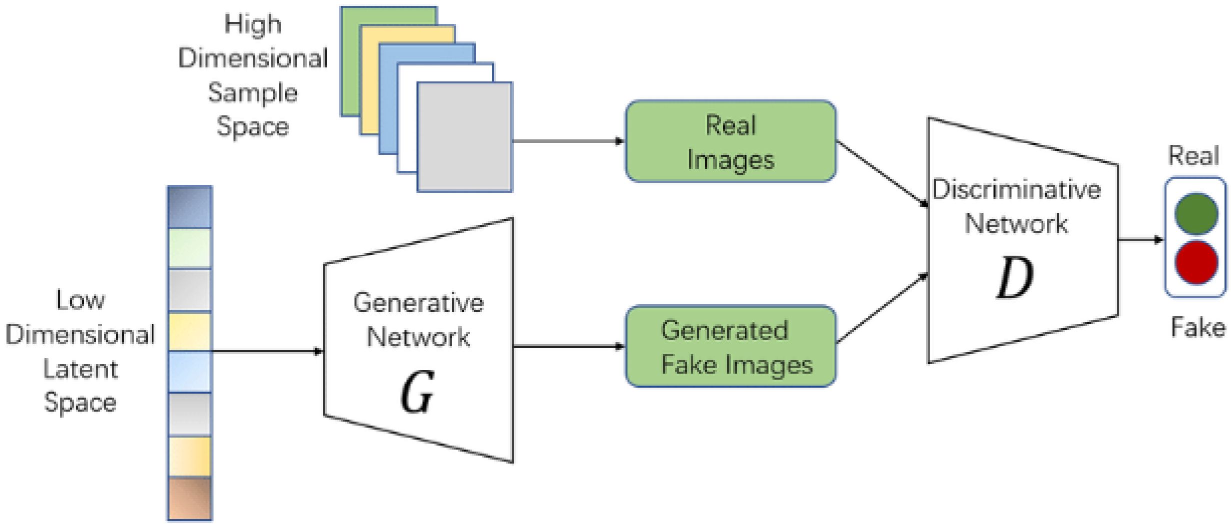
Training


Discriminator training process
Random noise
Generator
Training batch (fake)
Discriminator
Predictions
0.2
0.1
0.3
0.6
Label
0
0
0
0
Loss
Training batch (real)
Discriminator
Predictions
0.8
0.9
0.4
0.7
Label
1
1
1
1
Loss
Generator training process
Random noise
Generator
Training batch (fake)
Discriminator
Predictions
0.4
0.5
0.6
0.2
Label
1
1
1
1
Loss

Training Stages

- Train initial generator
- Iterate
  - train discriminator
  - train generator with additional discriminator loss
- Note: when training generator, leave discriminator parameters fixed
  (and vice versa)

GAN Training Objective



- **Goal:** Train a generator G to produce realistic samples that fool a discriminator D.

-

- **Training:** G and D play a two-player minimax game:

$$\min_G \max_D V(G, D)$$

GAN Training Objective

- **Discriminator Objective:**
  - Learns to distinguish real data from fake data.

$$\max_D \; \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

- **Generator Objective:**
  - Learns to generate samples that fool the discriminator.

$$\min_G \; \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

  - Often optimized via:

$$\max_G \; \mathbb{E}_{z \sim p_z}[\log D(G(z))] \quad \text{(non-saturating loss)}$$

Common Problems

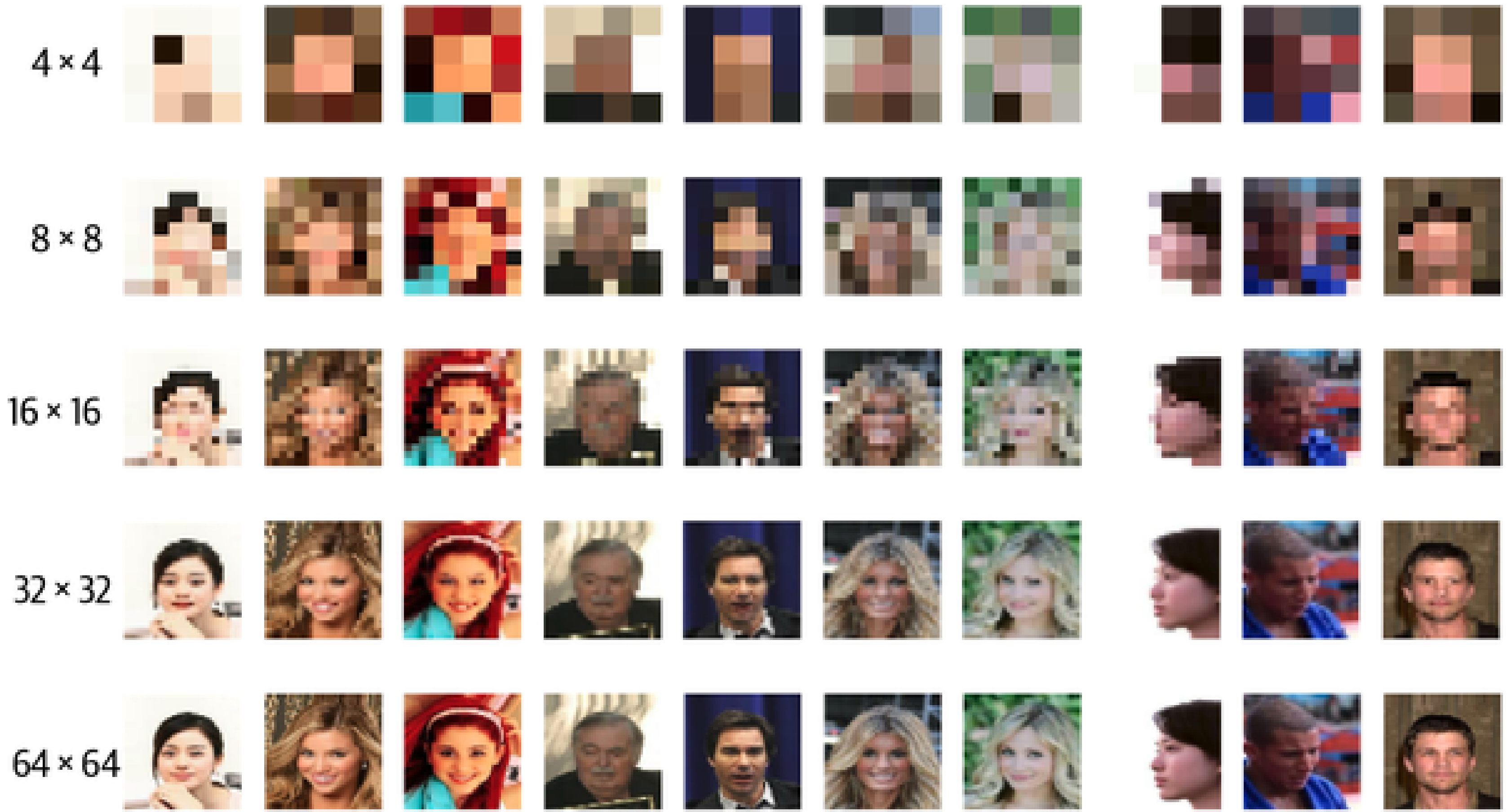
- Discriminator overpowers generator
- Generator overpowers discriminator
  (for instance mode collapse: generates unique image that fools discriminator)
- Training loss of generator is not informative
  (mainly battles discriminator and not fitting the training data)
- Many hyperparameters

Extensions

- Wasserstein GAN
- Lipschitz Constraint
- Gradient Penalty Loss
- Conditional GAN

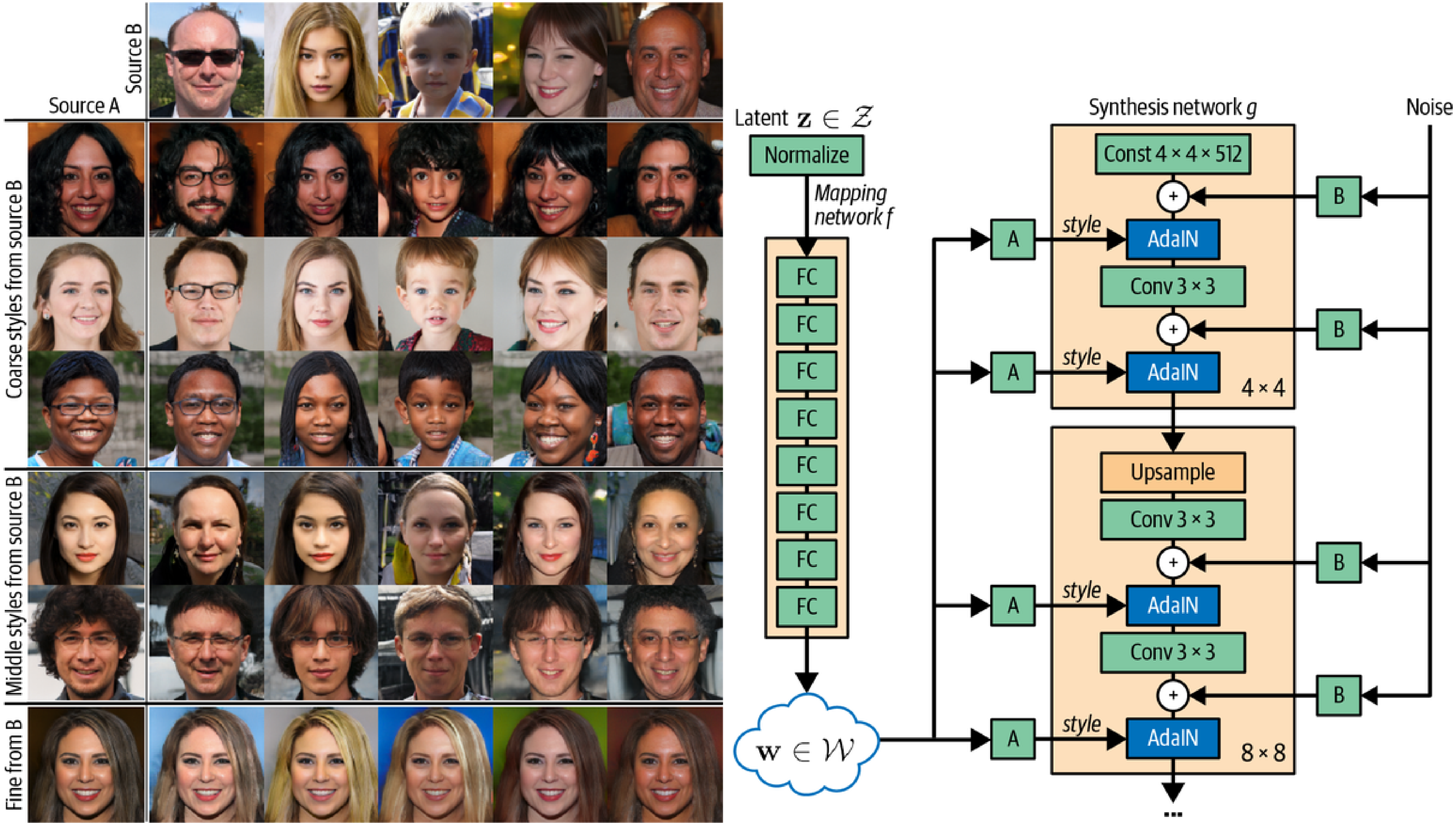
advanced gan

ProGAN: Progressively Upsizing the Image



- First train a model for lower resolution images
- Upscale the image in stages

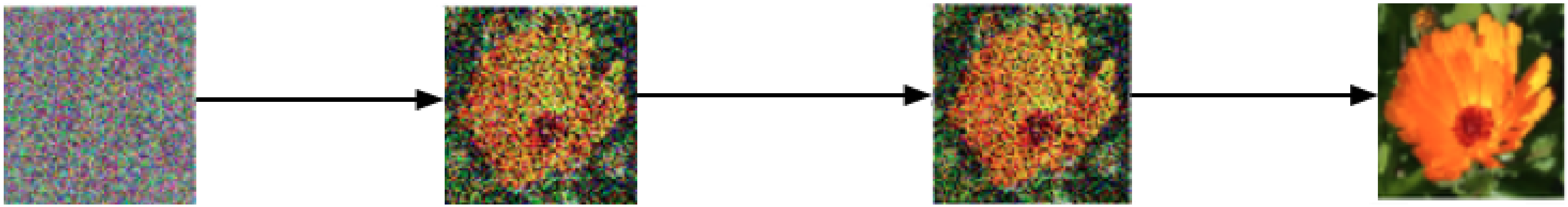
StyleGAN: Mixing Image Styles



diffusion models

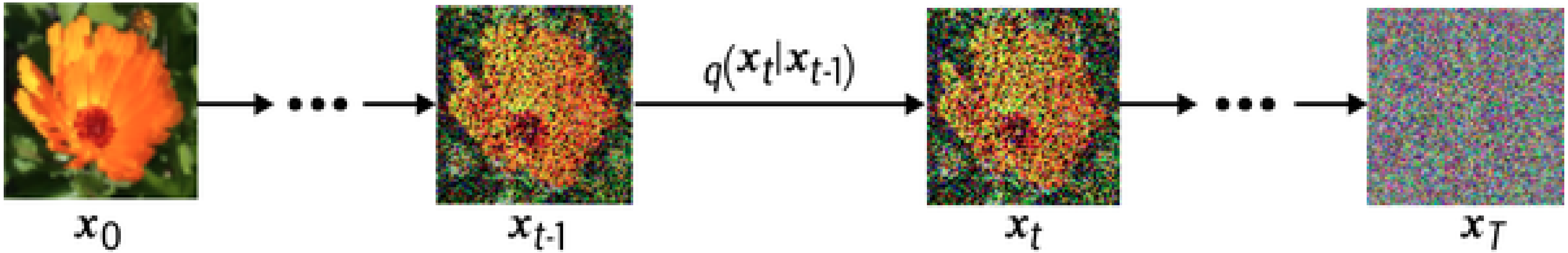
Diffusion Models

- Key idea
  - create noise image
  - generate image from noise
  - repeat this for, say, 20 steps



Create Training Data

- Training data for this process can be creating by adding noise



- This is done in stages, each time adding Gaussian noise ϵ_t
(mean 0 and unit variance, but then scaling to effective variance β_t below)

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\,\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\,\epsilon_{t-1}$$

- Or, written as a probability distribution from one image to the next

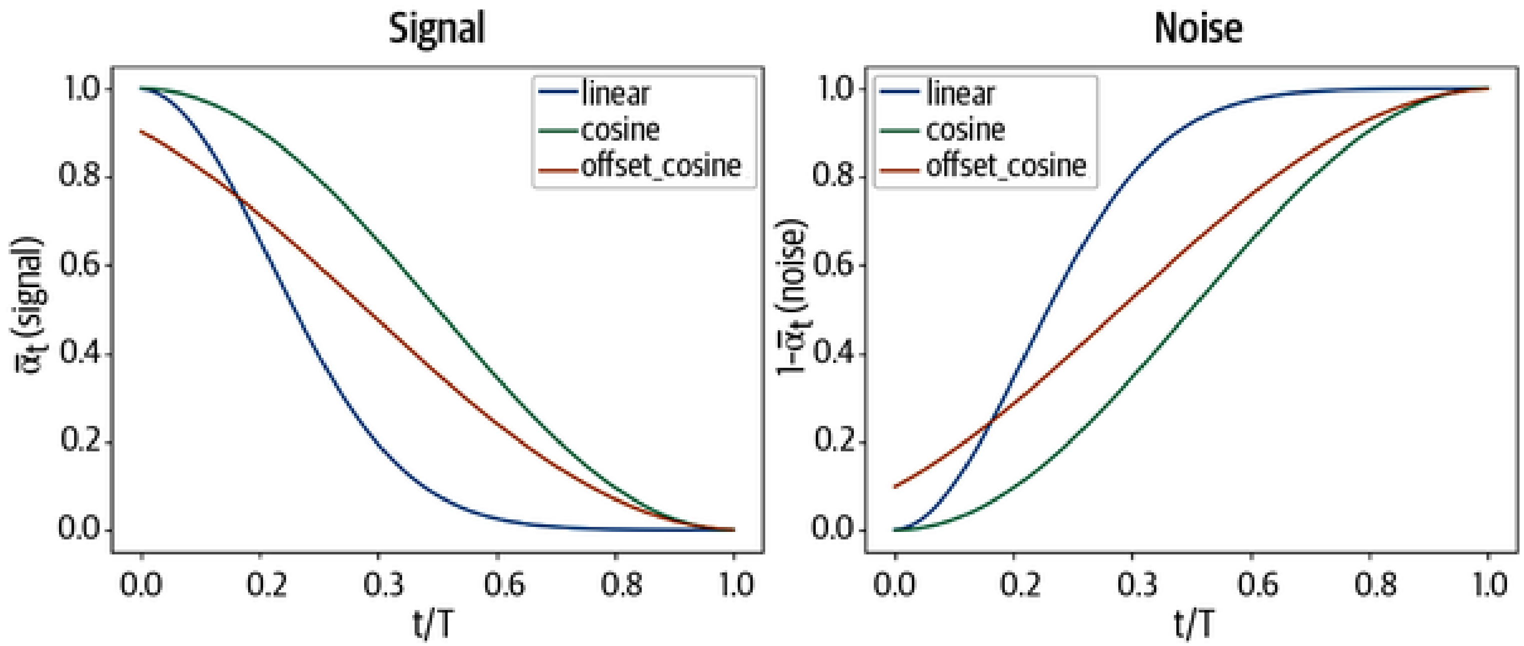
$$q\left(\boldsymbol{x}_t \,|\, \boldsymbol{x}_{t-1}\right) = \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{1-\beta_t}\,\boldsymbol{x}_{t-1}, \beta_t\,\mathbf{I}\right)$$

- After a large number of steps, this becomes indistinguishable from a noise image

Diffusion Schemes

- The variance β_t is changed throughout the process
  - small changes to initial, original image
  - larger changes towards the end to ensure randomness

- Common options: linear, cosine, offset cosine

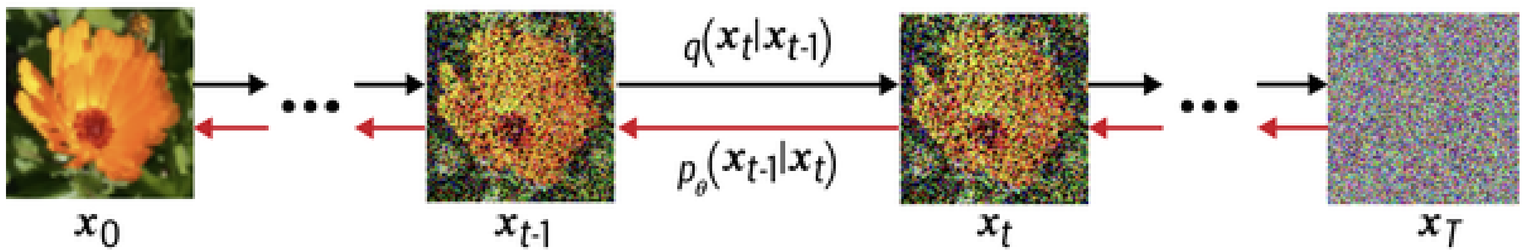


Effect of Diffusion Schemes

- Cosine diffusion scheme makes less changes initially

Goal: Learn How to Reverse this Process

- Learn a model p_θ that maps a noisy image x_t back to a less noisy image x_{t-1}



Refinements

- We actually learn a model that maps back x_t to the original image x_o
- The accumated noise can be computed

$$q\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0\right) = \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\overline{\alpha}_t}\,\boldsymbol{x}_0, \left(1-\overline{\alpha}_t\right)\mathbf{I}\right)$$

- The model predicts the noise $\epsilon_\theta(x_t)$

Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Flow-Matching

- **Diffusion Models**
  - Add noise step-by-step to data to create a noisy prior.
  - Learn to reverse this noising process to generate samples.
  - Requires many steps (e.g., 50–1000) and stochastic sampling.

- **Flow Matching (FM)**
  - Learns a continuous transformation (or flow) from noise to data in one shot.
  - Instead of reversing diffusion, directly fits a vector field that ”guides” particles.
  - No stochastic sampling: it's deterministic and much faster at inference.

- **Key Differences**
  - **Training:** FM minimizes a supervised loss; Diffusion minimizes denoising loss.
  - **Sampling:** FM uses ODE solvers; Diffusion uses stochastic reverse steps.
  - **Efficiency:** FM needs fewer steps and is faster at generation.

Flow Matching vs. Diffusion Models

- **Diffusion Models**
  - Forward process adds noise:

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

  - Model learns to predict noise:

$$\hat{\epsilon}_\theta(x_t, t) \approx \epsilon$$

  - Sampling via reverse stochastic process (e.g., DDPM, DDIM).

Flow Matching vs. Diffusion Models

- **Flow Matching**
  - Define a target flow (velocity) between data x_0 and noise x_1:

$$v_t(x) = \frac{dx_t}{dt}, \quad x_0 \rightarrow x_1$$

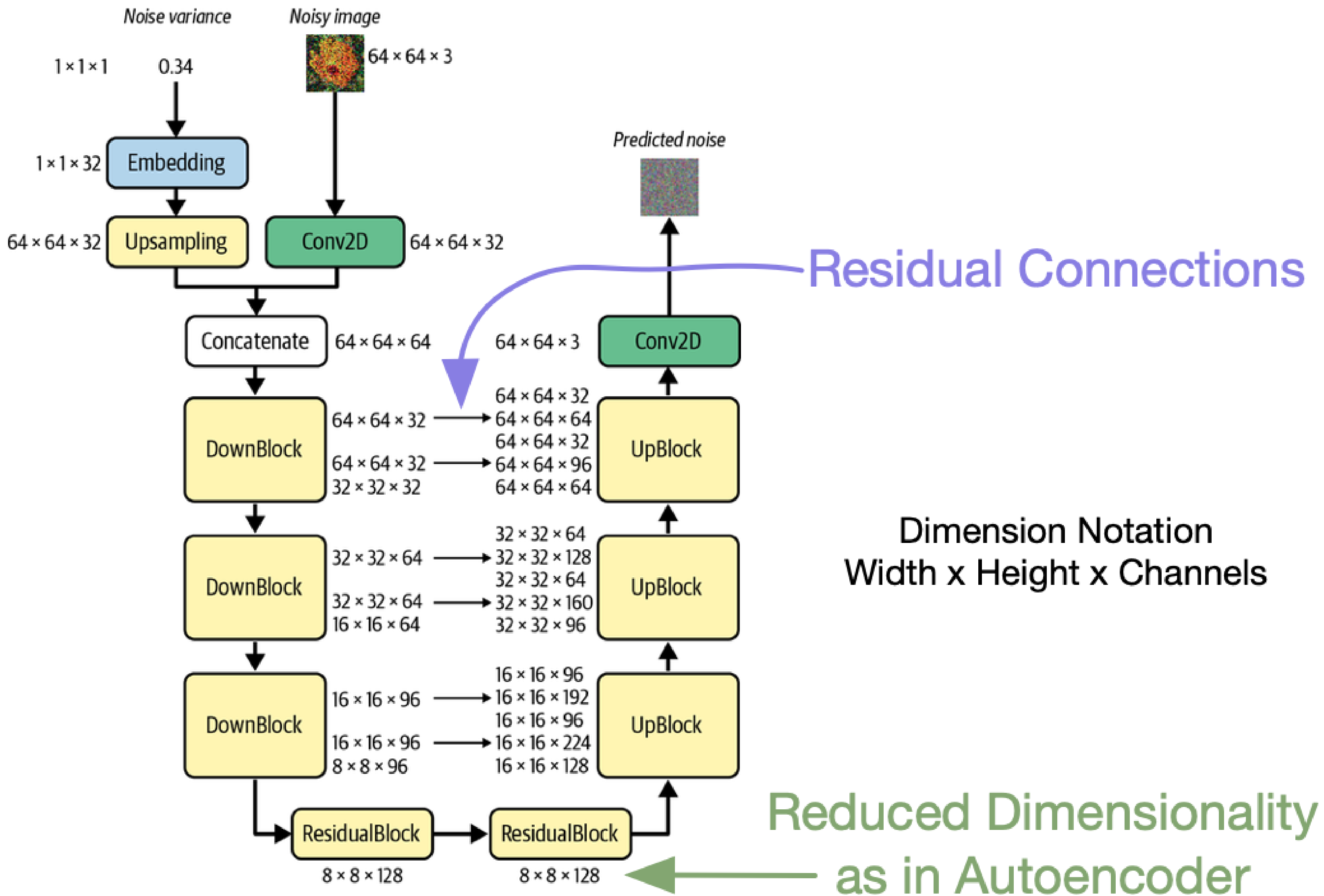
  - Learn a neural network $\hat{v}_\theta(x,t)$ to match this flow:

$$\hat{v}_\theta(x,t) \approx v_t(x)$$

  - Sample using an ODE solver:

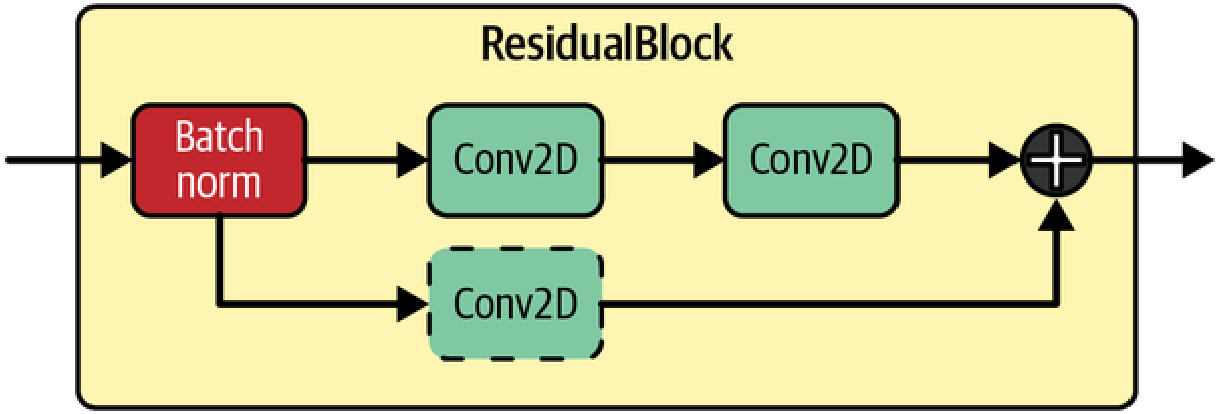
$$\frac{dx_t}{dt} = \hat{v}_\theta(x_t, t)$$

U-Net Model Overview



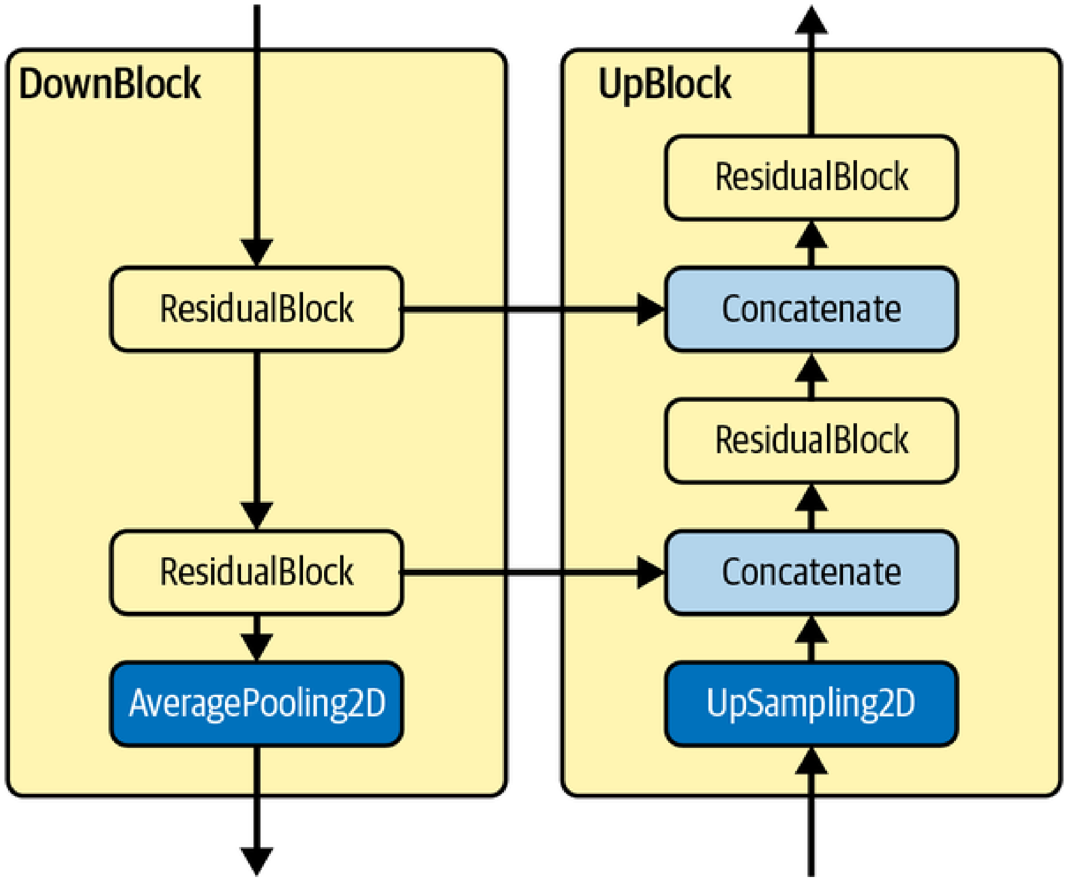
Dimension Notation
Width x Height x Channels

Residual Block

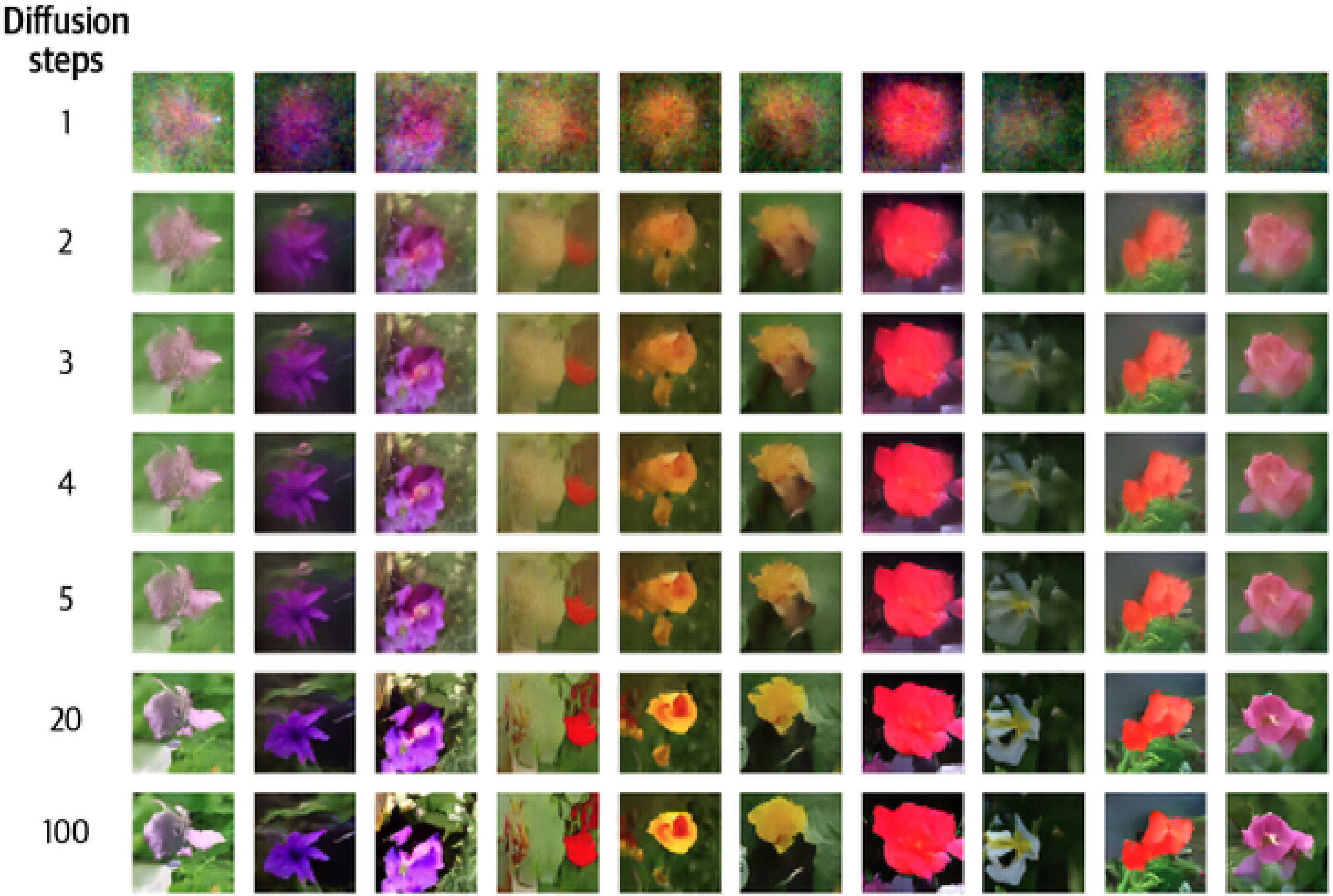


(may add convolution with kernel size 1 to residual connection to generate tensor with the right number of channels)

DownBlock and UpBlock



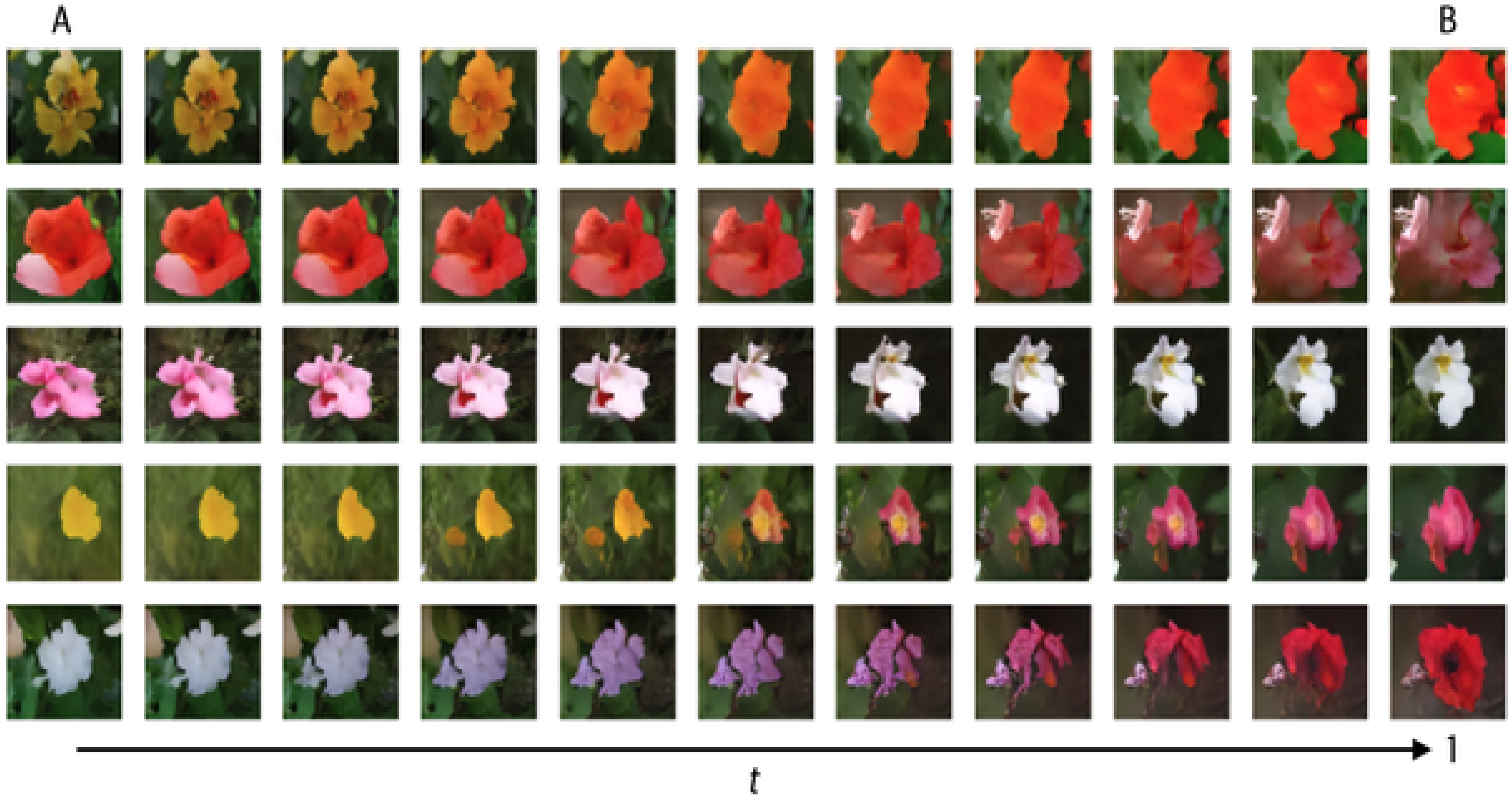
Generation in Multiple Steps



Interpolation

- Generation is deterministic given noise tensor a and b

$\Rightarrow$ We can interpolate between different noise images

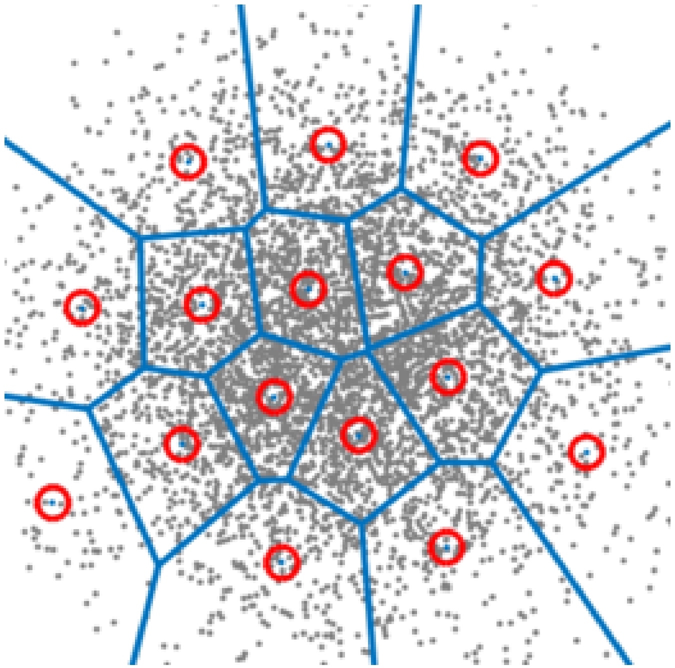


Common Failures

VQ-GAN: Vector Quantization

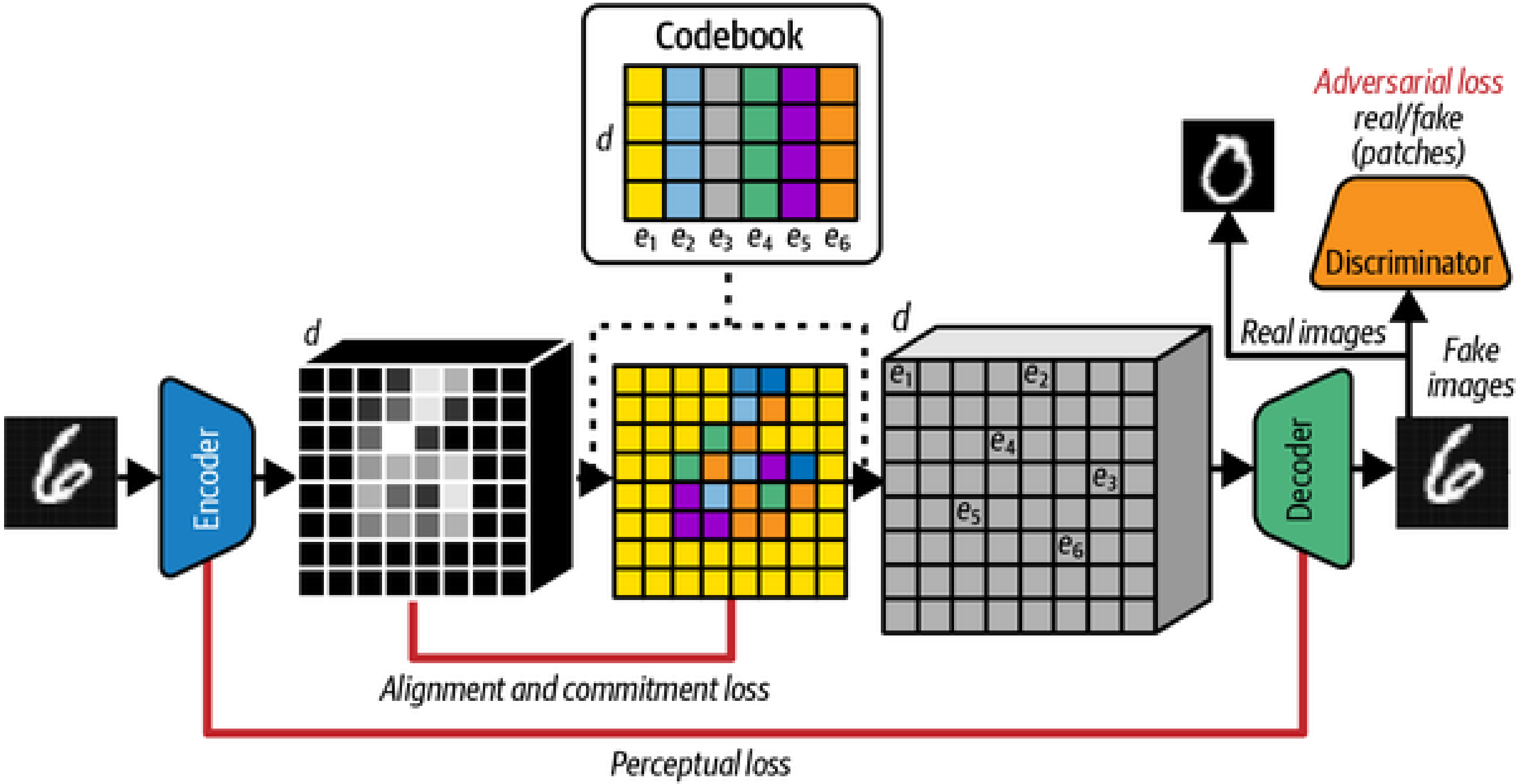
- Input: high dimensional vector

- K-Means Clustering



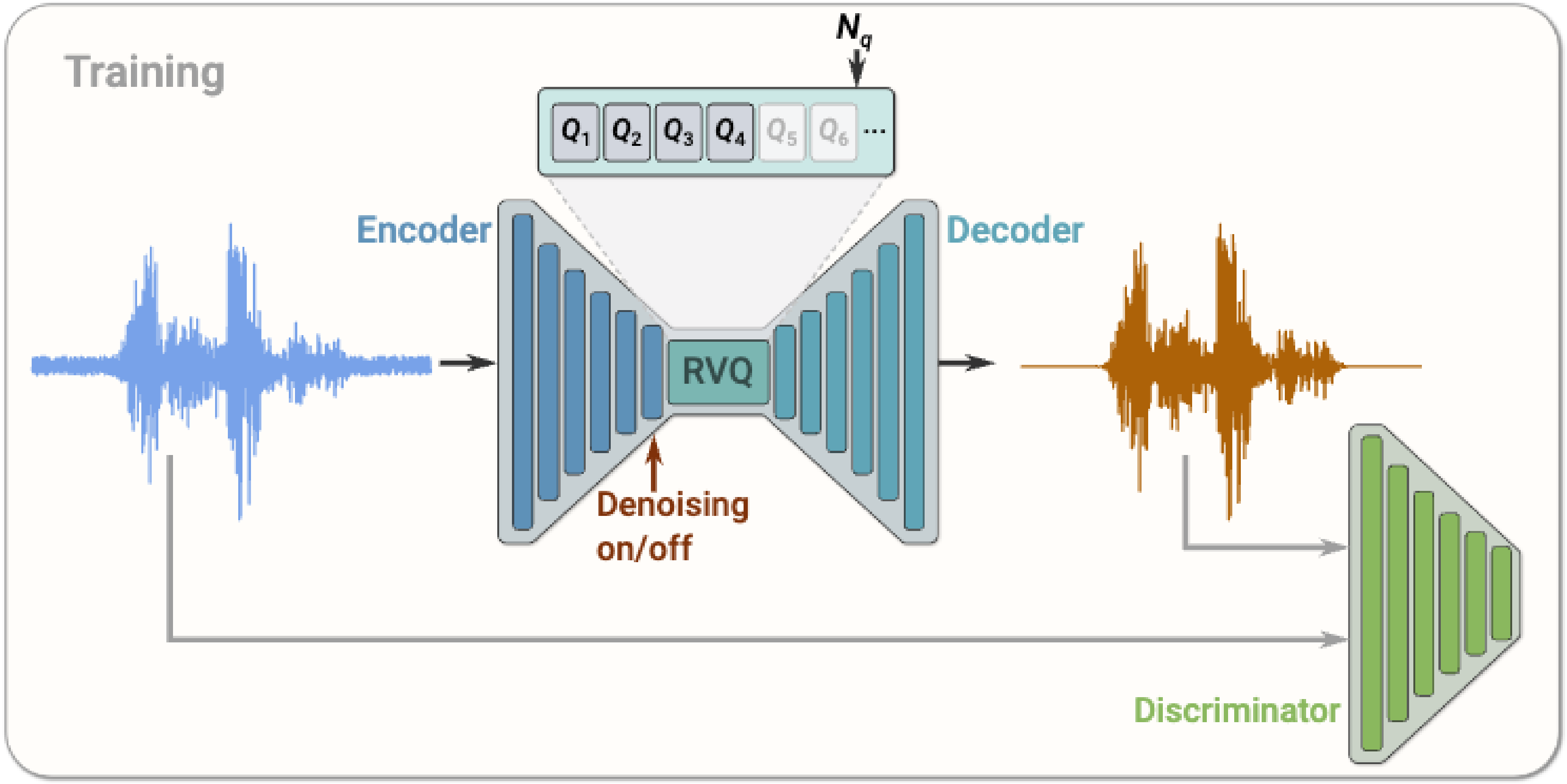
- Centroid vectors form a codebook, each vector is replaced with cluster ID

Discrete Latent Space



- Vectors are replaced by their nearest centroid

Discretization: RVQ and FSQ



RVQ

- **RVQ: Residual Vector Quantization**
  - Quantizes input vector using a sequence of codebooks.
  - Each stage encodes the residual from the previous stage:

$$r_0 = x, \quad q_i = \text{Quantize}(r_{i-1}, \mathcal{C}_i), \quad r_i = r_{i-1} - q_i$$

  - Final quantized output:

$$\hat{x} = \sum_{i=1}^{N} q_i$$

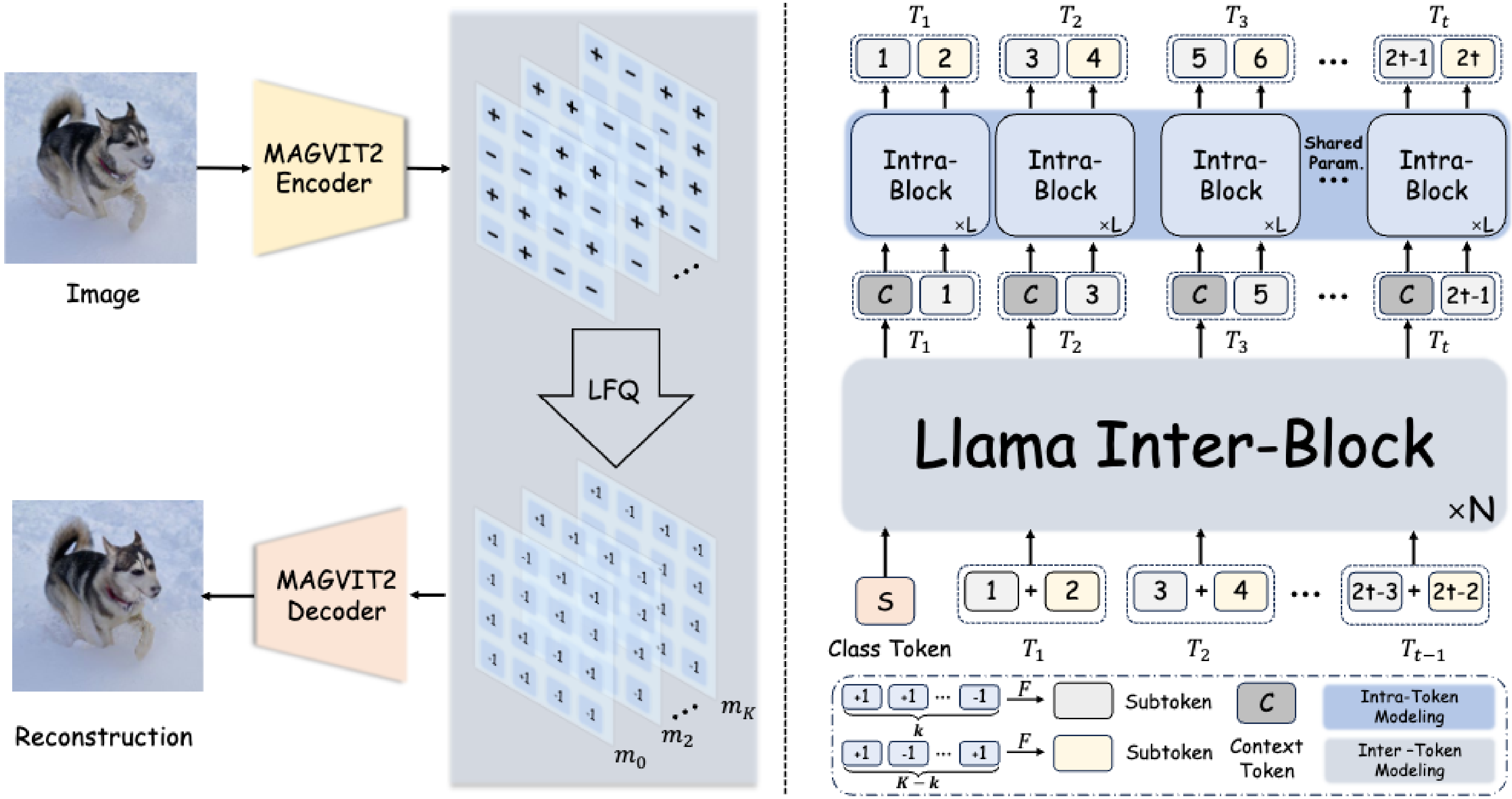
LFQ

- **LFQ: Lookup-Free Quantizer**
  - Avoids embedding lookups by directly binarizing latents.
  - Latent space is a Cartesian product of 1D variables:

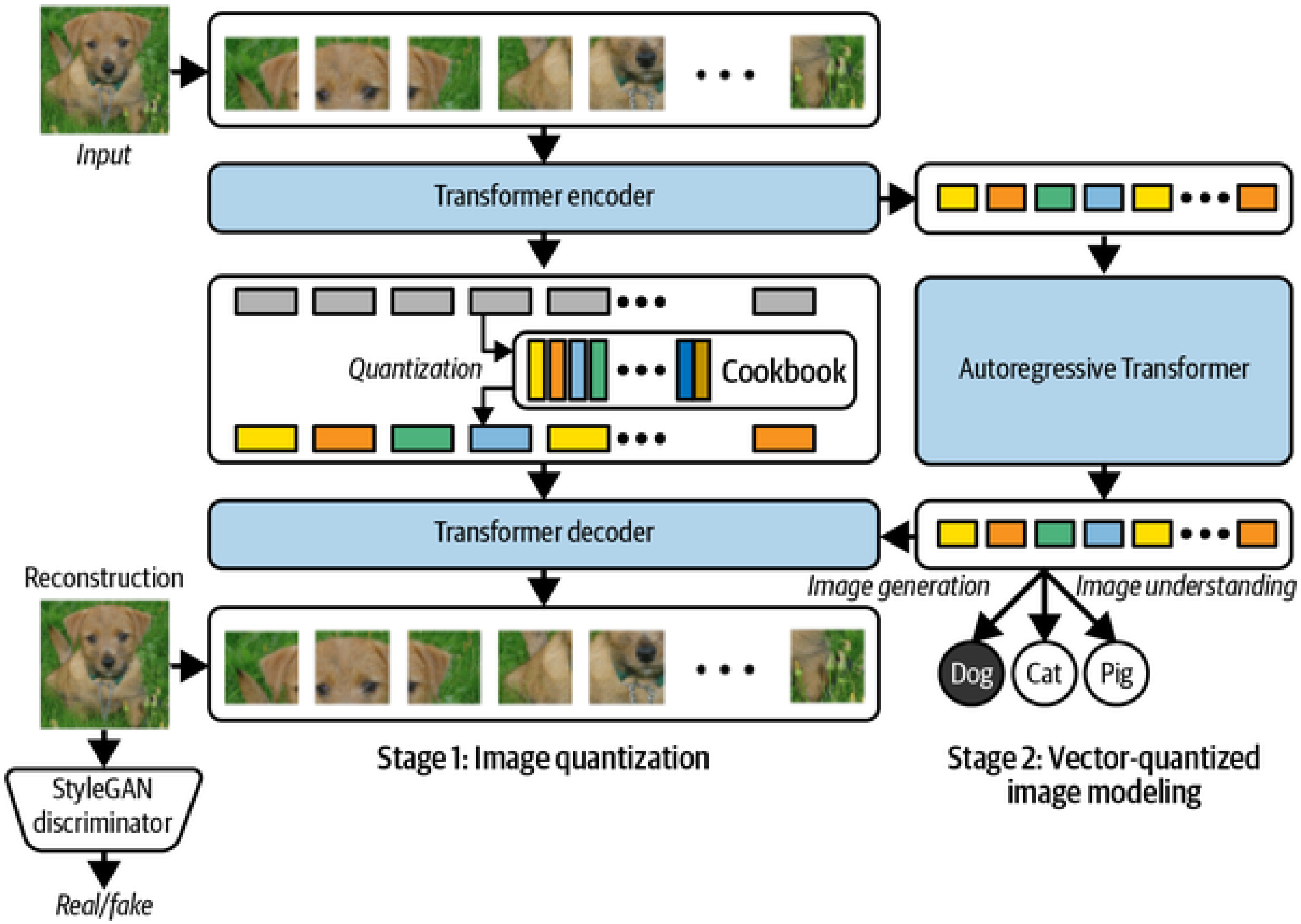
$$\hat{z}_i = \text{sign}(z_i) = -\mathbb{1}\{z_i \leq 0\} + \mathbb{1}\{z_i > 0\}$$

  - The codebook is implicit: $\hat{\mathcal{C}} = \{-1, 1\}^K$, with $|\hat{\mathcal{C}}| = 2^K$
  - Fast, memory-efficient, and no need to learn or store embeddings.

MagVIT2: VIT with FSQ-based quantization



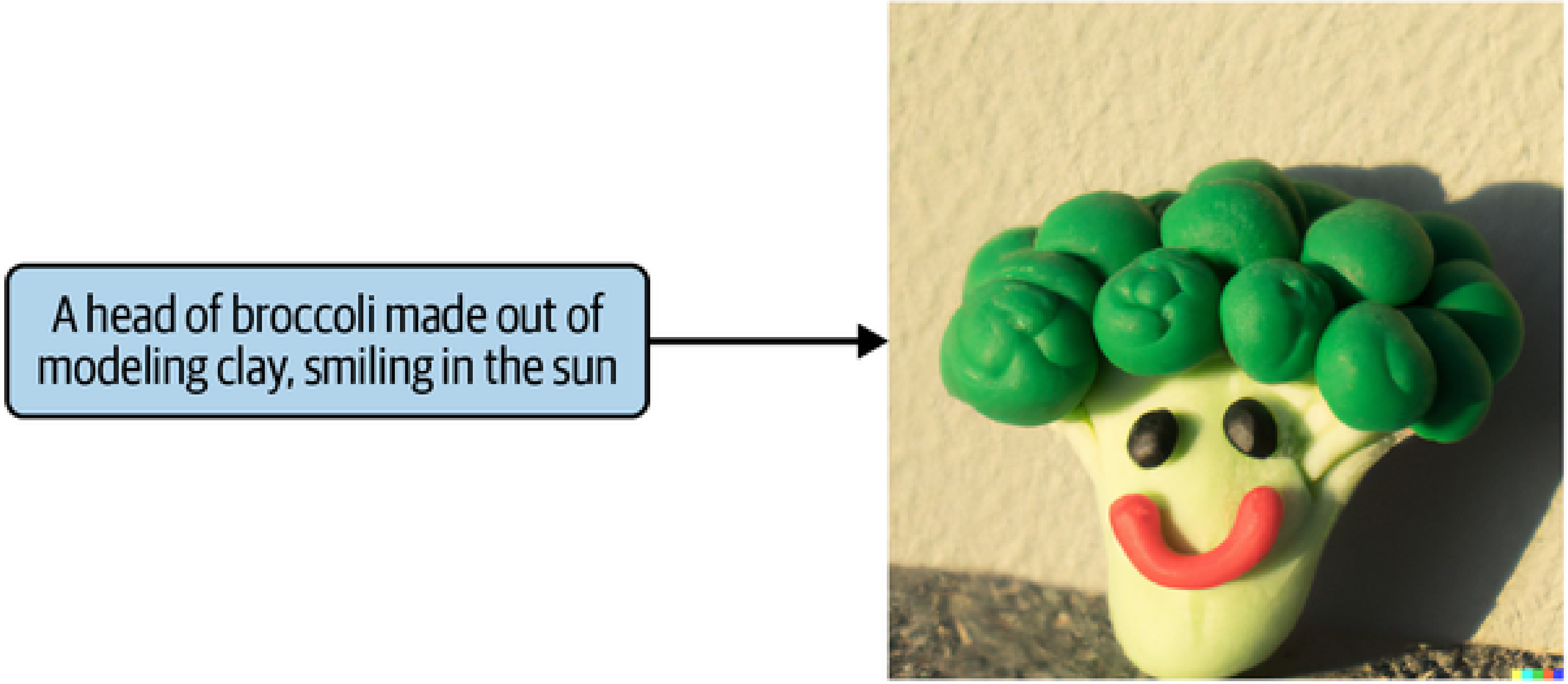
ViT VQ-GAN: Video Transformer



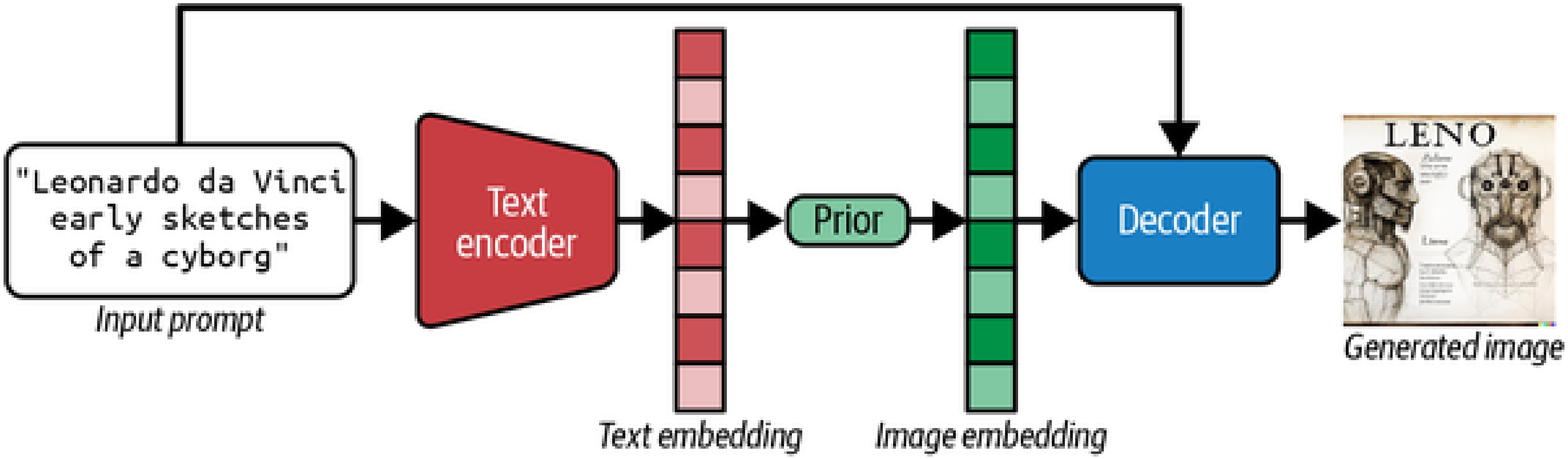
- Instead of using ConvNNs, Transformer model predicts sequence of patches
- This particular model also uses vector quantization

text to image generation

Text to Image

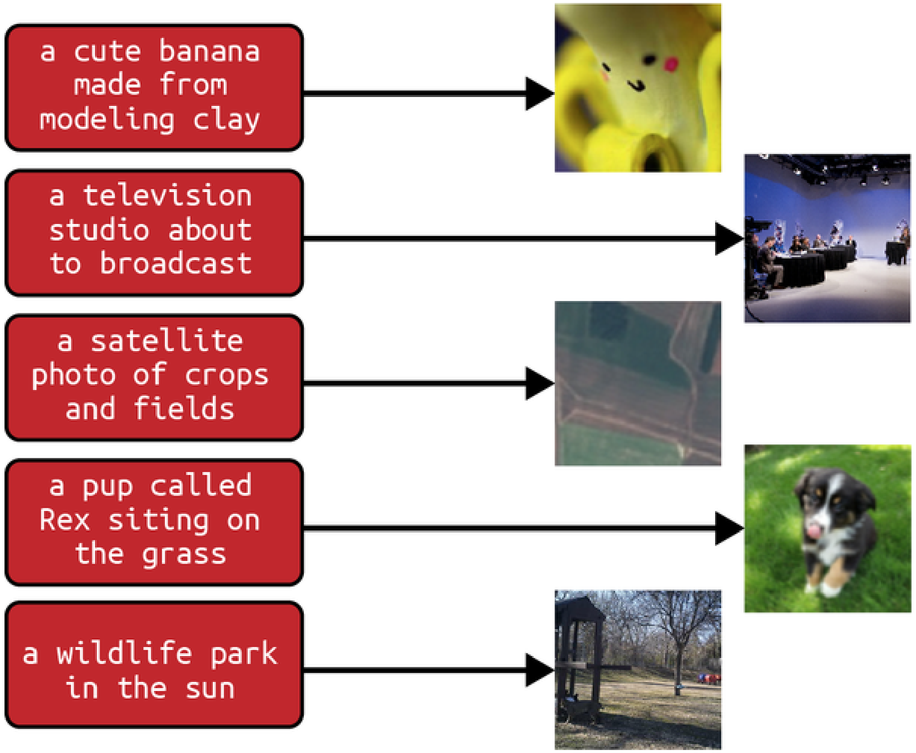


DALL.E 2



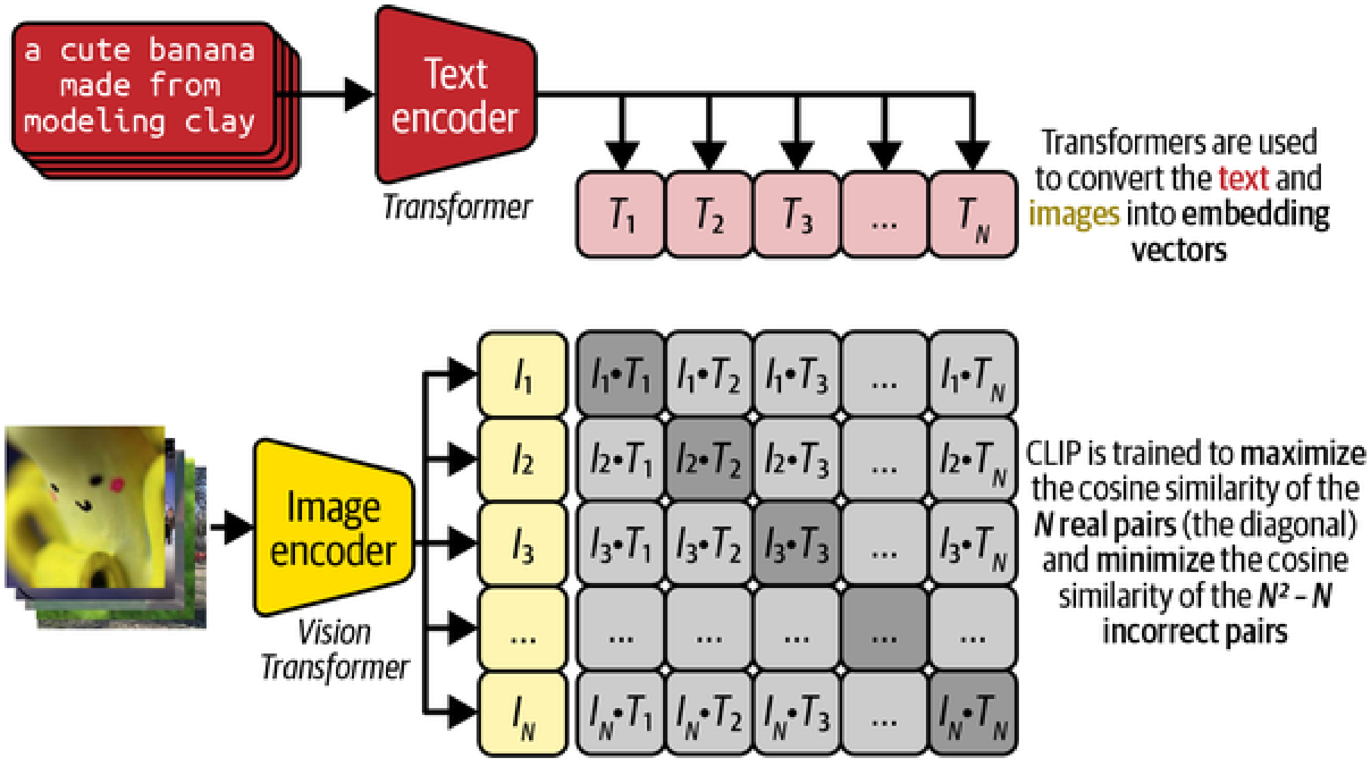
- Text is encoded as a prior to the image generation process
- Key training step: Contrastive Language-Image Pre-training (CLIP)

CLIP: Mapping Between Text and Images



- Given pairs of images and text (scraped from the Internet)
- Learn representations of text (using Transformer models)
- Learn representations of images (using ViT-VQ GAN)
- Learn mapping between them

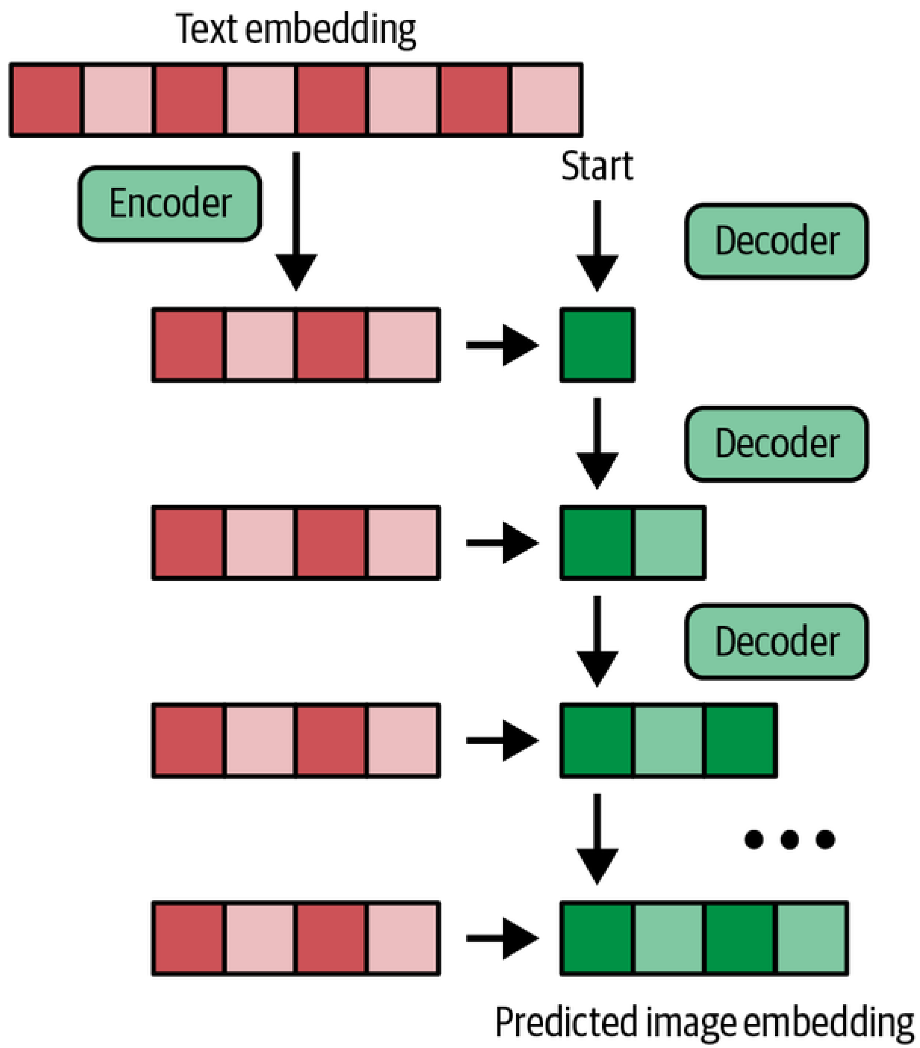
Contrastive Learning



- Representation of text and image as a single vector, mapped to same size
- Training: minimize cosine similarity of real pairs
- Note: this is not a generative model

Generation

- Image decoder is a Transformer model
- Predicts patches of the image at each step
- Generation is also conditioned on the text representation (the prior)
- Alternatively: diffusion decoder



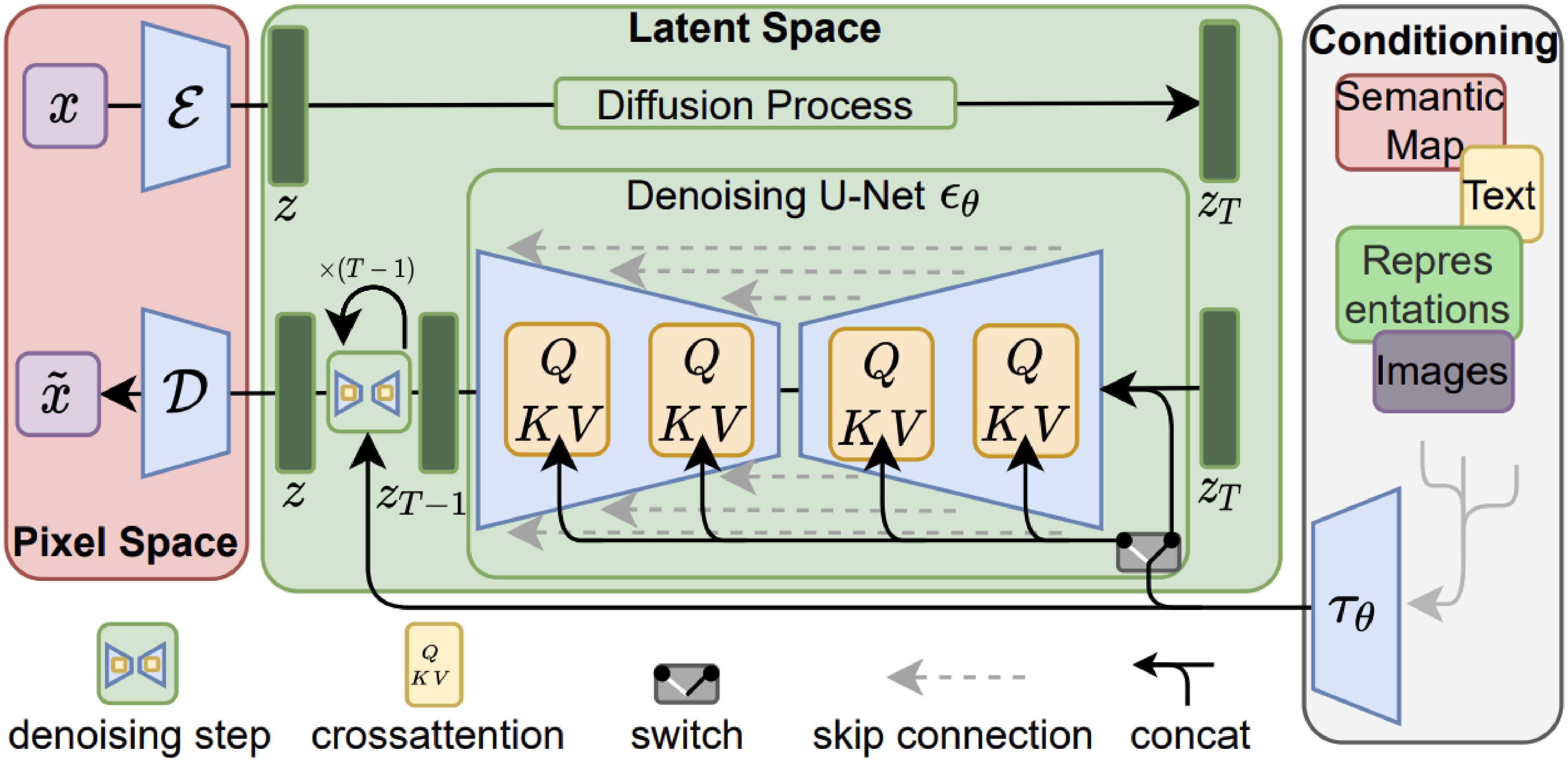
stable diffusion

Robach et al. (2022):
High-Resolution Image Synthesis with Latent Diffusion Models

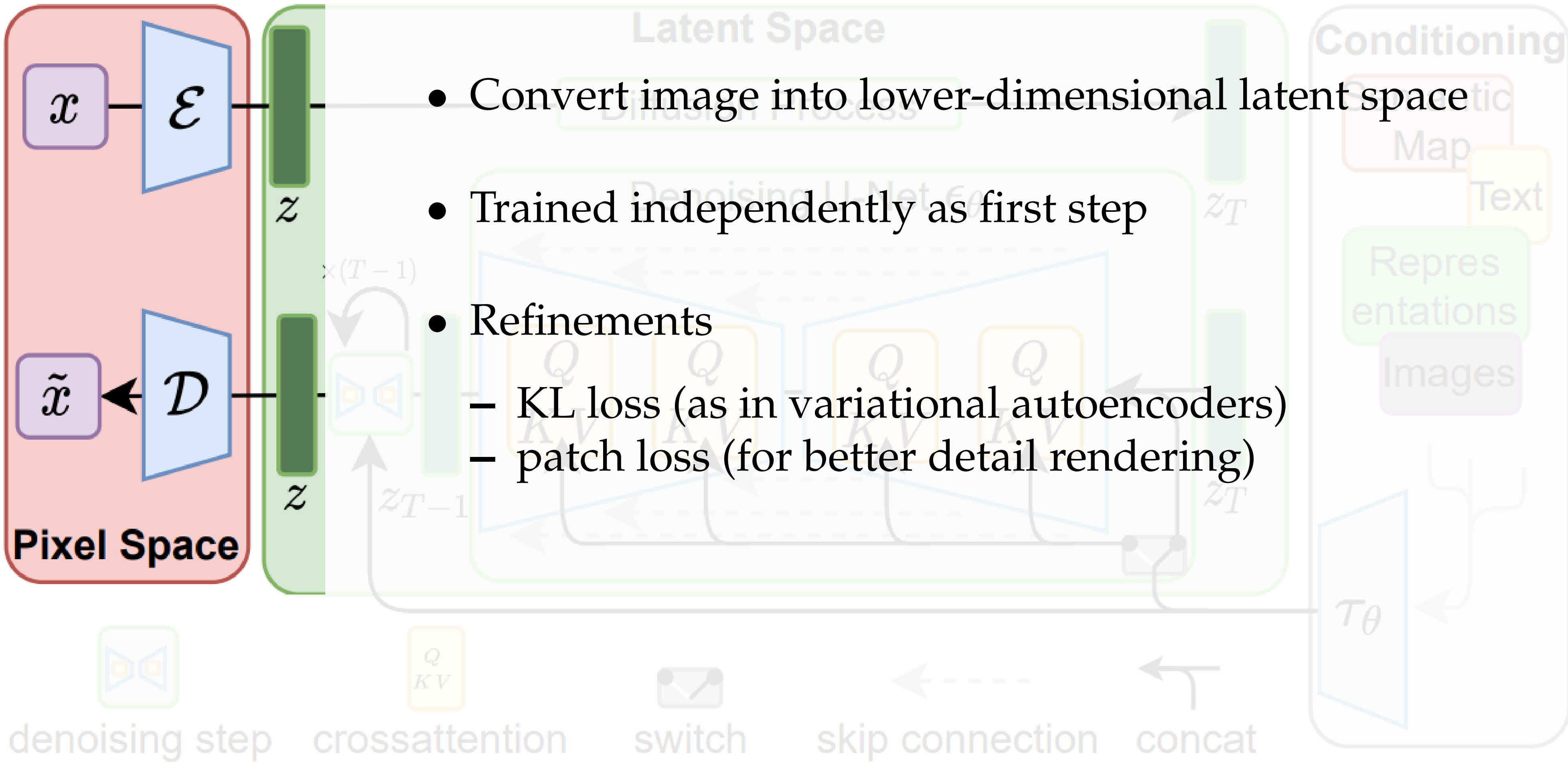
Stable Diffusion

Generated from prompt "a photograph of an astronaut riding a horse"

Latent Diffusion Model

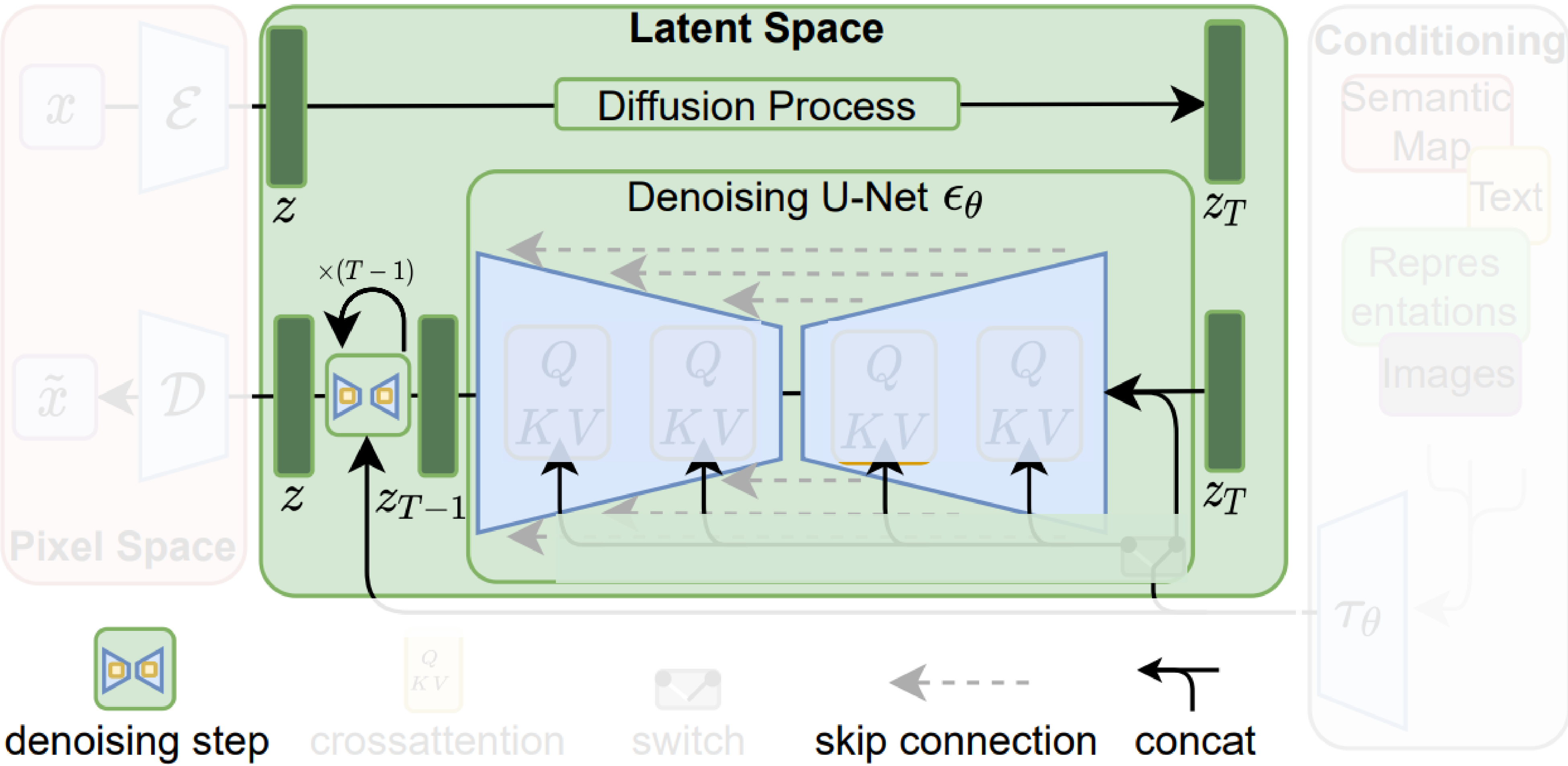


Autoencoder



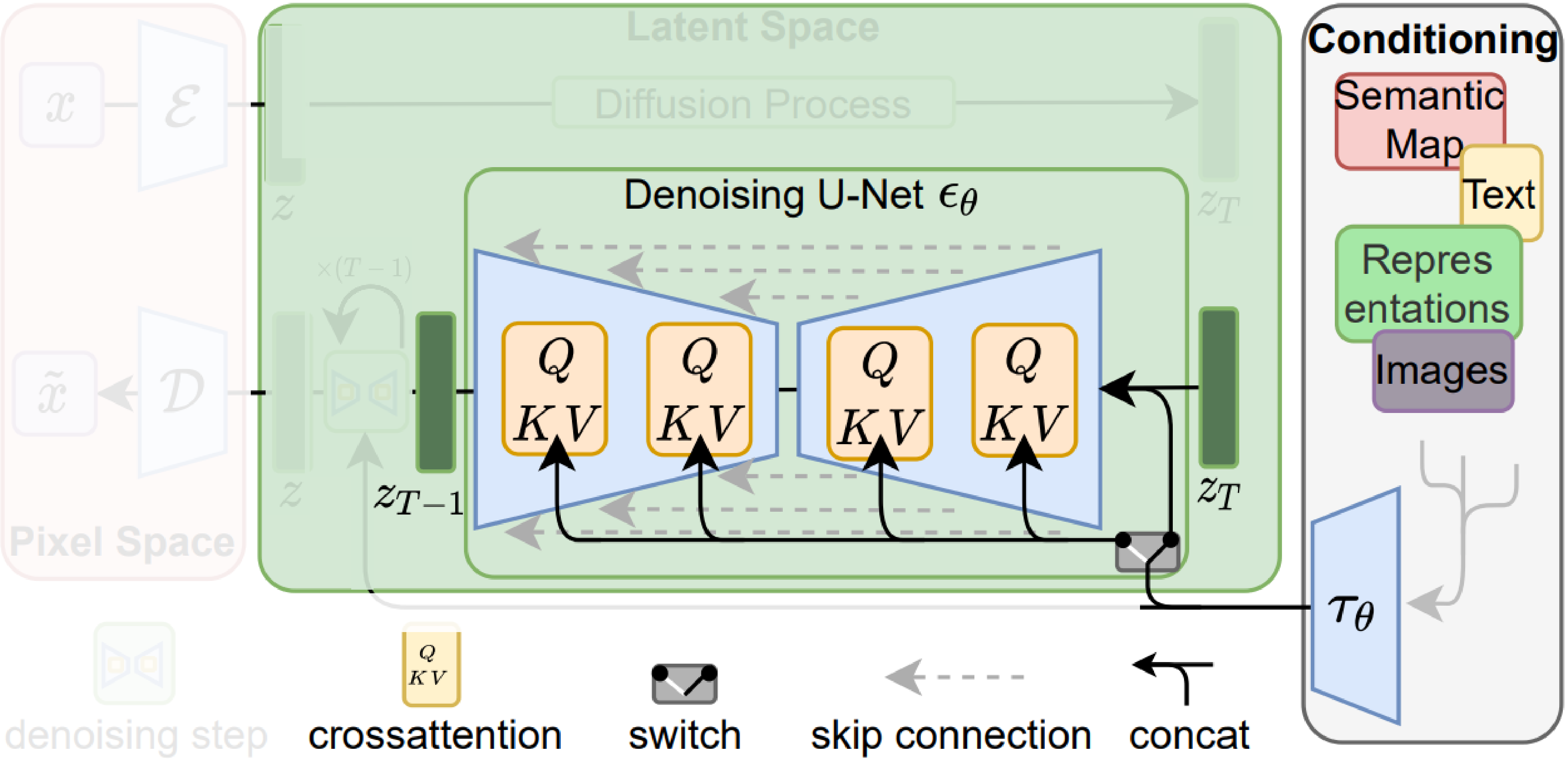
- Convert image into lower-dimensional latent space
- Trained independently as first step
- Refinements
  - KL loss (as in variational autoencoders)
  - patch loss (for better detail rendering)

Diffusion Model



Operates in latent space — otherwise the same U-Net from before

Conditioning on External Information



Attention (Query, Key, Value) to representations of semantic maps, text, images

Conditioning on External Information

- External input y is converted into an intermediate representation $\tau_\theta \in \mathbb{R}^{M \times d_r}$

- For use in the attention model, the intermediate representations of the U-Net are also flattened to $\varphi_i(z_t) \in \mathbb{R}^{N \times d_\epsilon^i}$

- Attention is $$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V$$

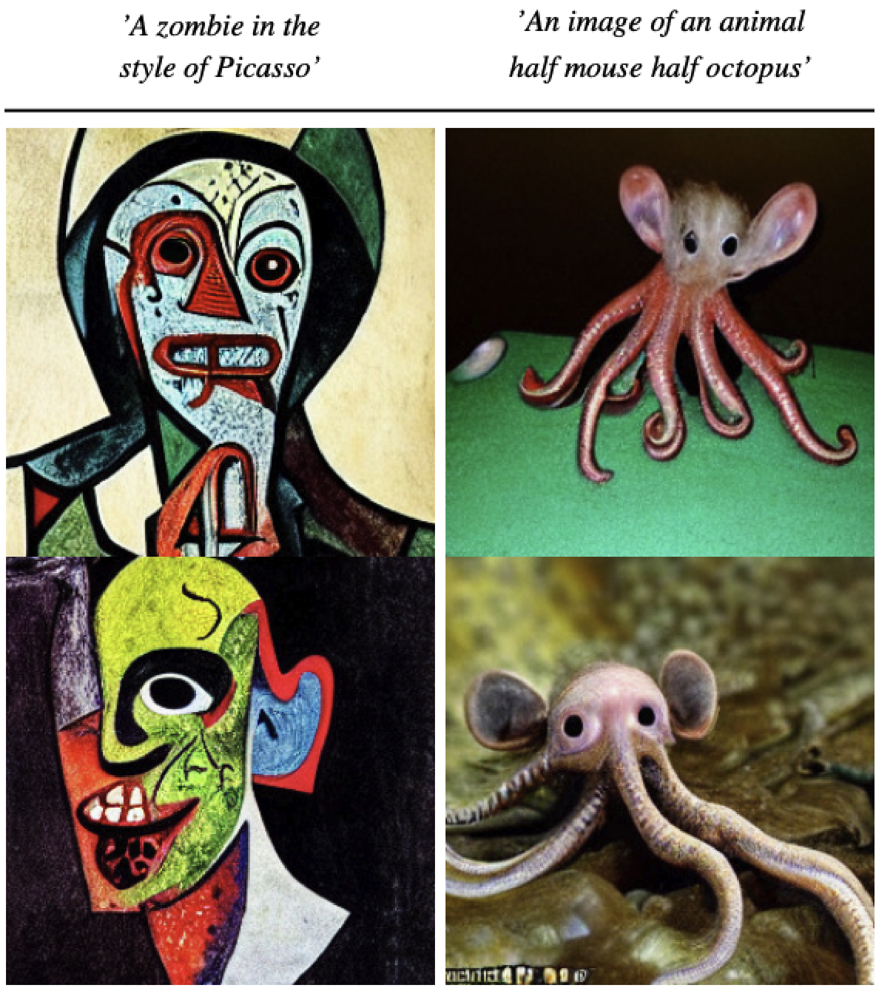
  with $$Q = W_Q^{(i)} \cdot \varphi_i(z_t) \qquad K = W_K^{(i)} \cdot \tau_\theta(y) \qquad V = W_V^{(i)} \cdot \tau_\theta(y)$$

  that map to vectors of size d

- Parameters for τ_θ and ϵ_θ are jointly optimized using diffusion objective

Text to Image

- Text is represented as a sequence of words
- Transformer model generates $\tau_\theta \in \mathbb{R}^{M \times d_r}$
- Trained on language prompts
  - LAION-400M
  - 400 million text-image pairs
  - extracted from web pages with alt-text in HTML image tag
  - filtered in various ways



Semantic Maps

- Trained on Open Images dataset of images with labelled object detection
  - 9.2 million images collected from Flickr
  - bounding boxes with object labels
  - computer-assisted annotation: automatic labels vetted by humans

Inpainting

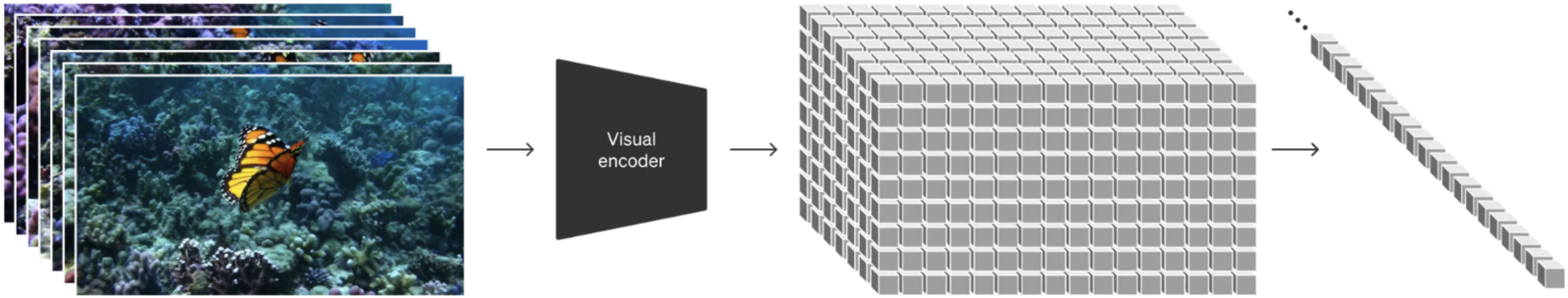
- Automatic generation of training data with synthetic masks
- Training aims to reconstruct the original image

video generation

OpenAI (2024): Video generation models as world simulators (Sora)

https://openai.com/research/video-generation-models-as-world-simulators

Video Representation

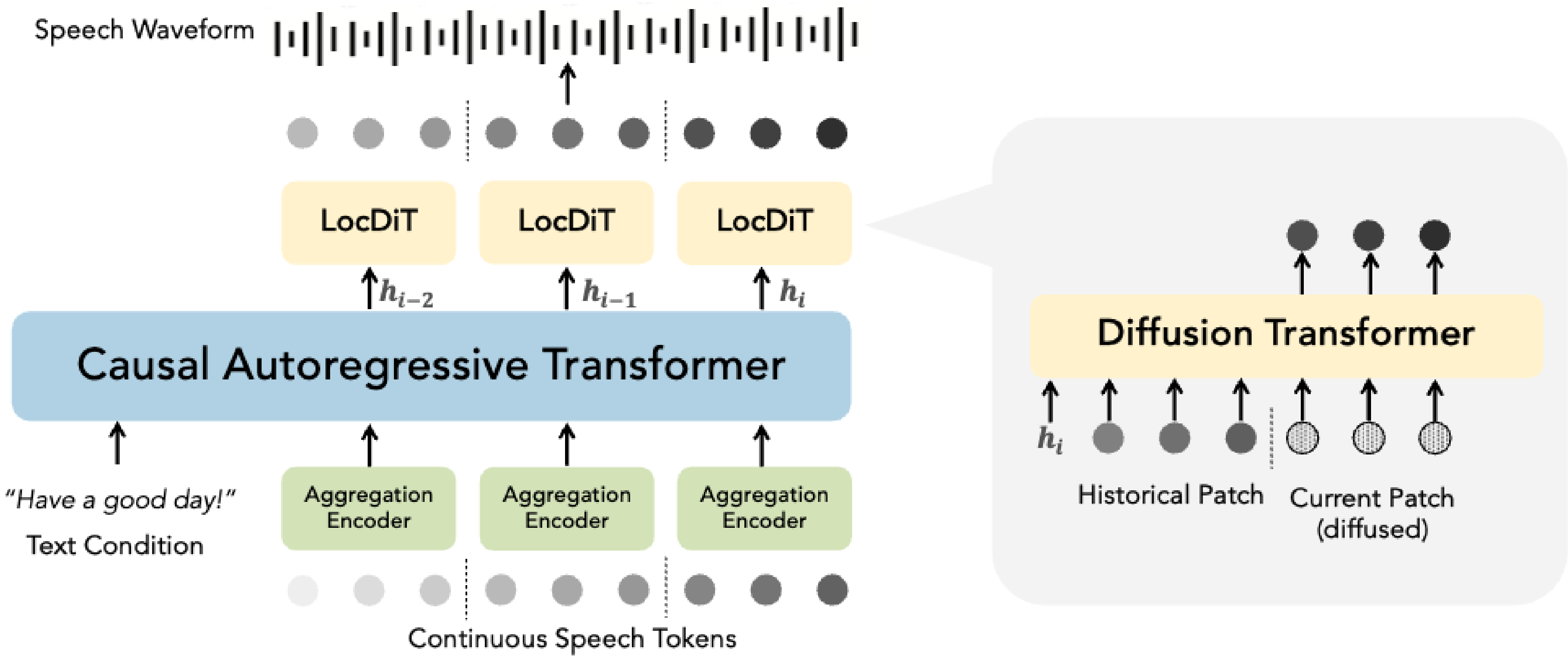


- Up to a full minute of high definition video
- Compress image into lower dimensional latent space
- Decompose representation into spacetime patches

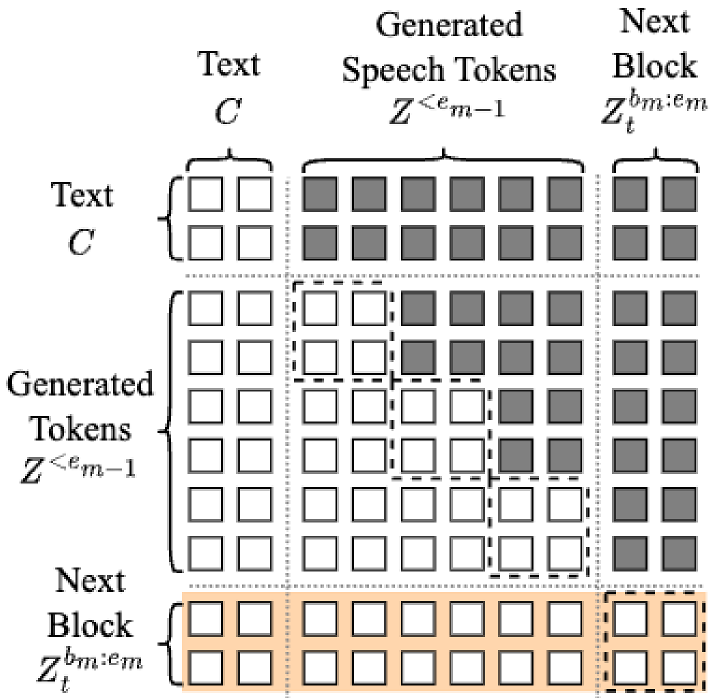
Diffusion Model

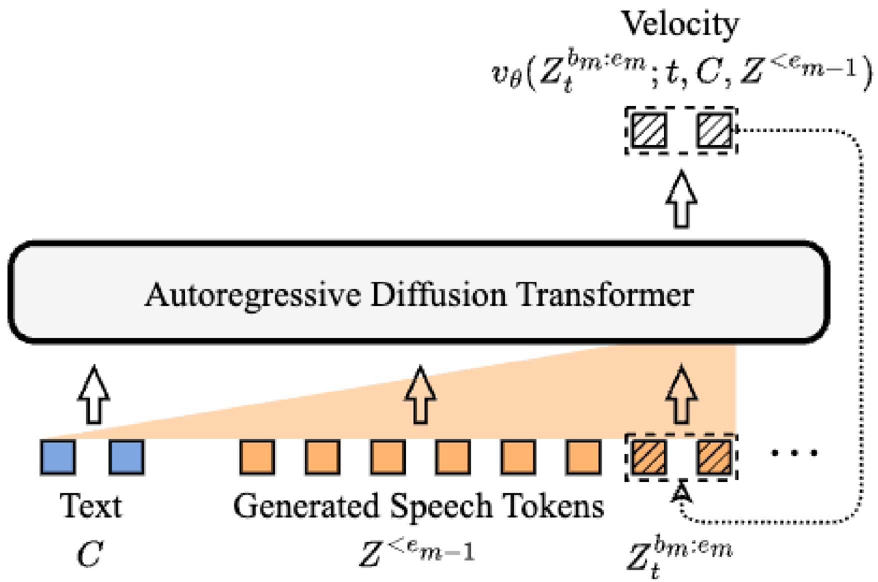
- Given noisy patches and conditioning text prompts
- Predict original clean patches using Transformer model
- Training data generated by re-captioning existing videos
  - first, train captioning model
  - use it to produce highly descriptive text captions for video

Diffusion Transformer Autoregressive (DiTAR) Model



Autoregressive Diffusion Transformer





Beyond Text Prompting

- Prompting with images (maybe images generated by DALL-E)
- Extending existing video (forward or backward)
- Video-to-video editing (video + text prompt $\rightarrow$ new video)
- Connecting videos

questions?