

Overview of Generative AI and Foundation Models

Chen Change Loy

Outline

Part I

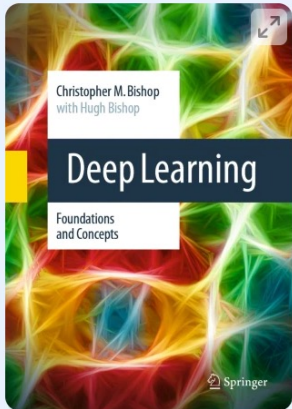
- Key Generative Model Families
 - Variational Autoencoders
 - Autoregressive Models
 - Generative Adversarial Models
 - Diffusion Models
- Rise of Foundation Models
 - Scaling Law
 - Transformers
 - CLIP
 - Unified Multimodal Models
 - Vision Foundation Models

Part II

- Text Generation
- Visual Generation
- How These Models Work in Practice

Reference Books

[Home](#) > Textbook



Deep Learning

Foundations and Concepts

Textbook | © 2024

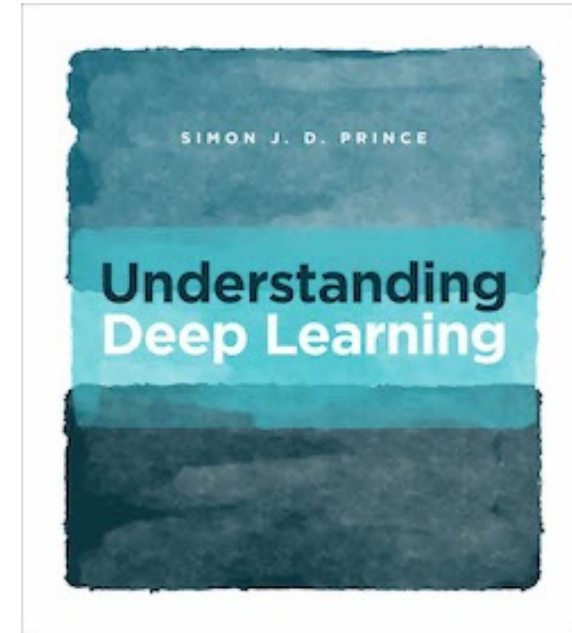
✓ Access provided by Nanyang Technological University Library

[Download book PDF](#) 

Overview

Authors: [Christopher M. Bishop](#), [Hugh Bishop](#)

- Foundational and conceptual approach emphasizes real-world practical value of techniques for a wide range of learners
- Companion volume to the author's standard reference text Pattern Recognition and Machine Learning
- To reinforce key ideas, end-of-chapter exercises of varying difficulty are included to promote active learning



Understanding Deep Learning

by Simon J.D. Prince

To be published by MIT Press Dec 5th 2023.

<https://udlbook.github.io/udlbook/>

Observations

- Algorithmic and model architecture improvements have provided substantial improvements
- Self-supervised learning removes the bottleneck of labeled data
- Increasing scale—compute, data, and model size—consistently delivers better results. Scaling has become a ‘universal recipe’ across modalities, including language, vision, protein, and multimodal tasks.
- Emergent capabilities at scale
 - Reasoning, generalization, in-context learning, cross-modal alignment, and tool-use emerge only when models surpass certain scale thresholds
 - Larger models learn more abstract patterns, enabling surprising transfer across tasks

Key Generative Model Families

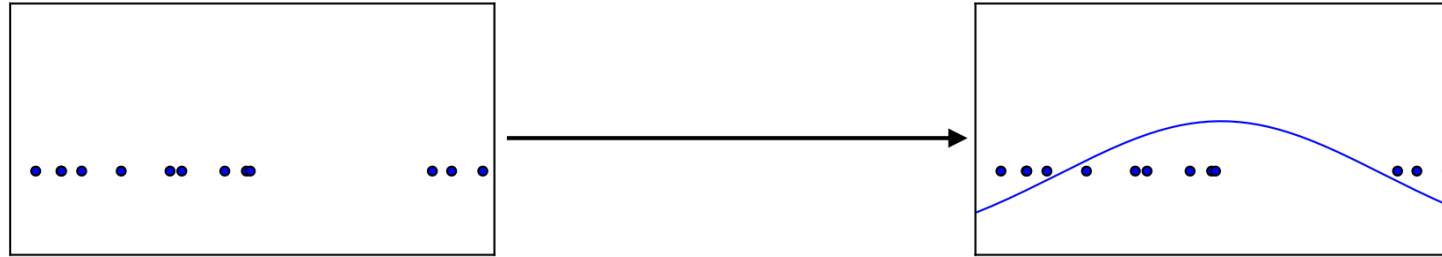
Core Conceptual Difference

- Discriminative models: learn boundaries
 - Model $p(y|x)$ → classification, detection, segmentation
 - Goal: given input → predict label
- Generative models: learn distribution
 - Model $p(x)$ or $p(x|c)$ → synthesis, editing
 - Goal: understanding and create new samples following data distribution



Generative Modeling

Density estimation – a core problem in unsupervised learning



Sample generation



Training samples – $p_{data}(x)$

Model samples – $p_{model}(x)$
(ideally $p_{model}(x) = p_{data}(x)$)

Generative Models

- Variational Autoencoders
 - Probabilistic latent space
- Autoregressive Models
 - Model the joint distribution by predicting the next element conditioned on all previous elements
 - LLMs: Next-token generative training from massive corpora
- Generative Adversarial Networks
 - Adversarial training
- Diffusion Models
 - Noise–denoise paradigm. State-of-the-art fidelity

Autoencoders

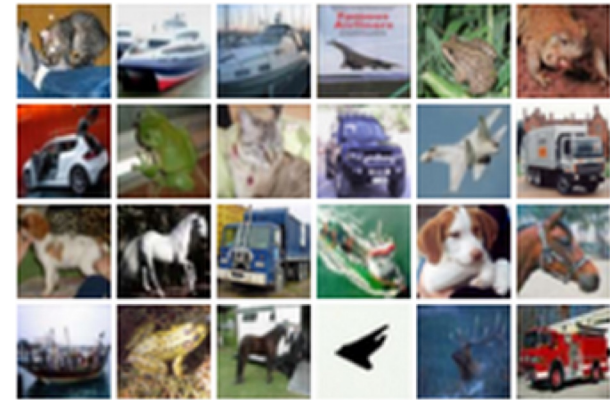
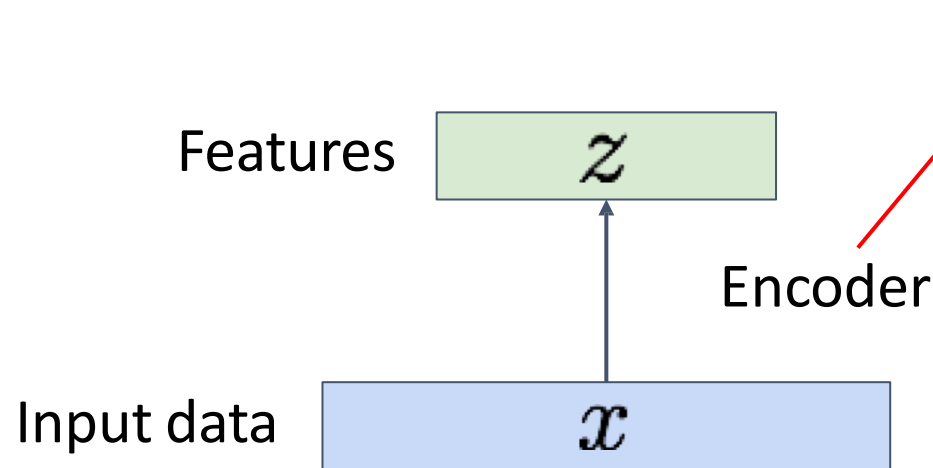
Unsupervised method for learning feature vectors from raw data x , without any labels

Features should extract useful information (maybe object identities, properties, scene type, etc) that we can use for downstream tasks

Originally: Linear + nonlinearity (sigmoid)

Later: Deep, fully-connected

Later: ReLU CNN, Transformers



Input Data

Autoencoders

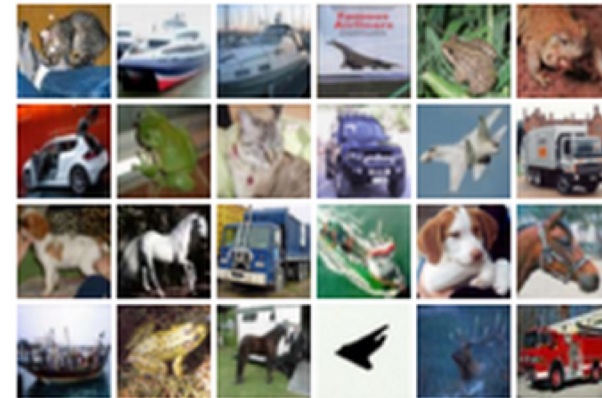
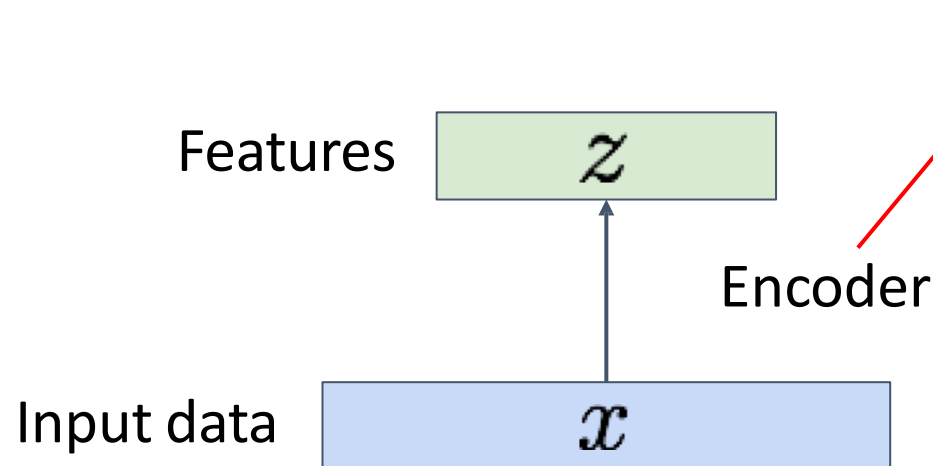
Problem: How can we learn this feature transform from raw data?

Features should extract useful information (maybe object identities, properties, scene type, etc) that we can use for downstream tasks
But we can't observe features!

Originally: Linear + nonlinearity (sigmoid)

Later: Deep, fully-connected

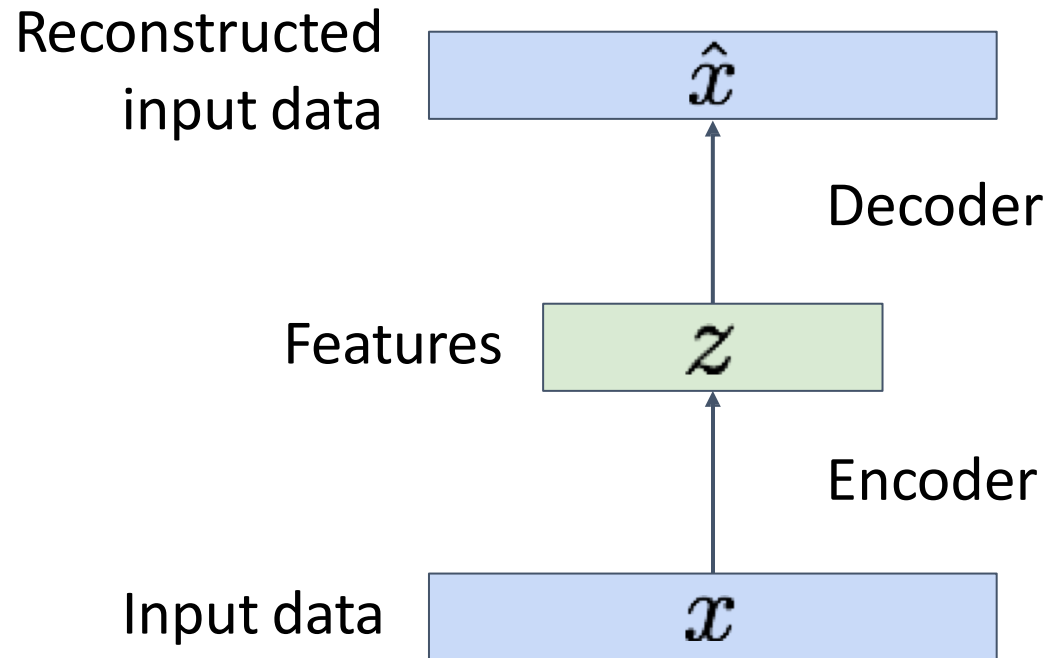
Later: ReLU CNN, Transformers



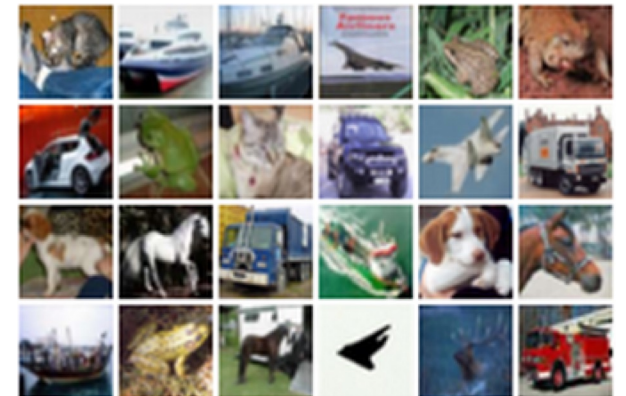
Input Data

Autoencoders

- **Problem:** How can we learn this feature transform from raw data?
 - **Idea:** Use the features to reconstruct the input data with a **decoder**
- “Autoencoding” = encoding itself



Originally: Linear + nonlinearity (sigmoid) **Later:** Deep, fully-connected **Later:** ReLU CNN (upconv), Transformers

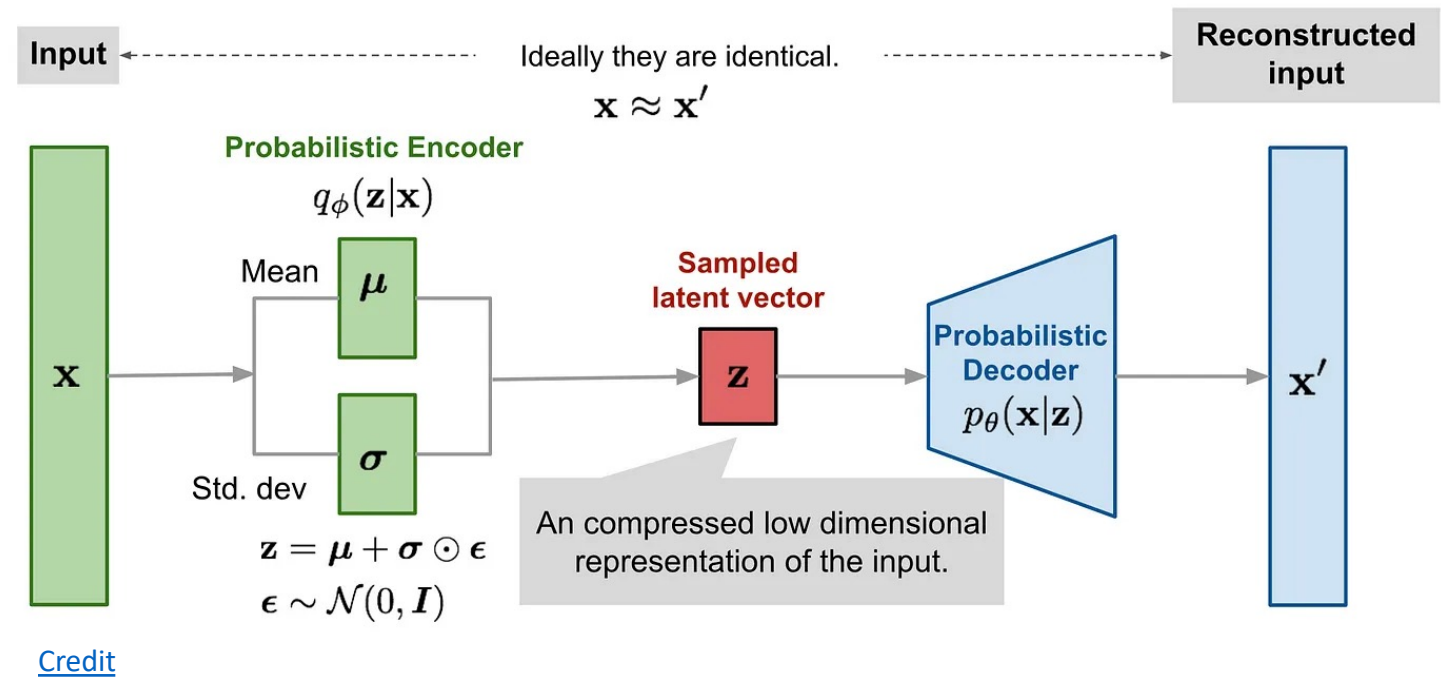


Input Data

Variational Autoencoders

Instead of encoding inputs to fixed points, VAEs encode inputs as **probability distributions over the latent space**.

This probabilistic approach allows VAEs to generate new, diverse outputs by **sampling** from these distributions.



Variational Autoencoders

32x32 CIFAR-10



Labeled Faces in the Wild



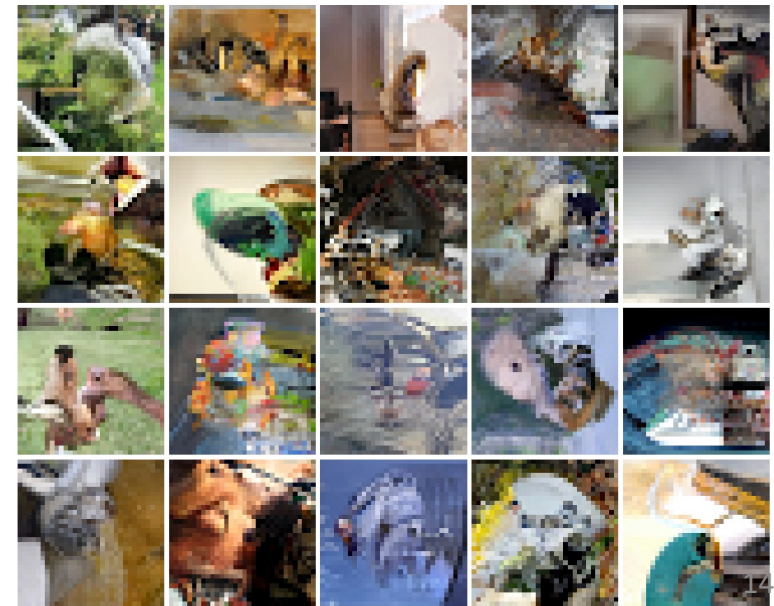
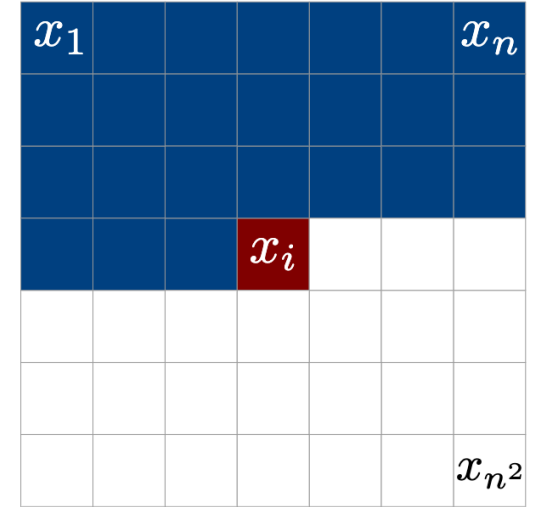
Autoregressive Models

- Model the joint distribution by **predicting the next element conditioned on all previous elements**

$$p(x_1, x_2, \dots, x_T) = \prod_{t=1}^T p(x_t \mid x_{<t})$$

Early Vision Example: PixelRNN

- PixelRNN (van den Oord et al., 2016)
 - Generates images pixel-by-pixel, each pixel conditioned on all pixels above/left.
 - Uses recurrent LSTM layers with masked convolutions to preserve autoregressive order.
 - Captures fine textures and spatial correlations.



Autoregressive Models

LLMs apply the same autoregressive principle but operate on **tokens** instead of pixels.

Autoregressive (look at prefix, predict next word):

The typhoon signal ____

The typhoon signal was ____

The typhoon signal was upgraded ____

The typhoon signal was upgraded to ____

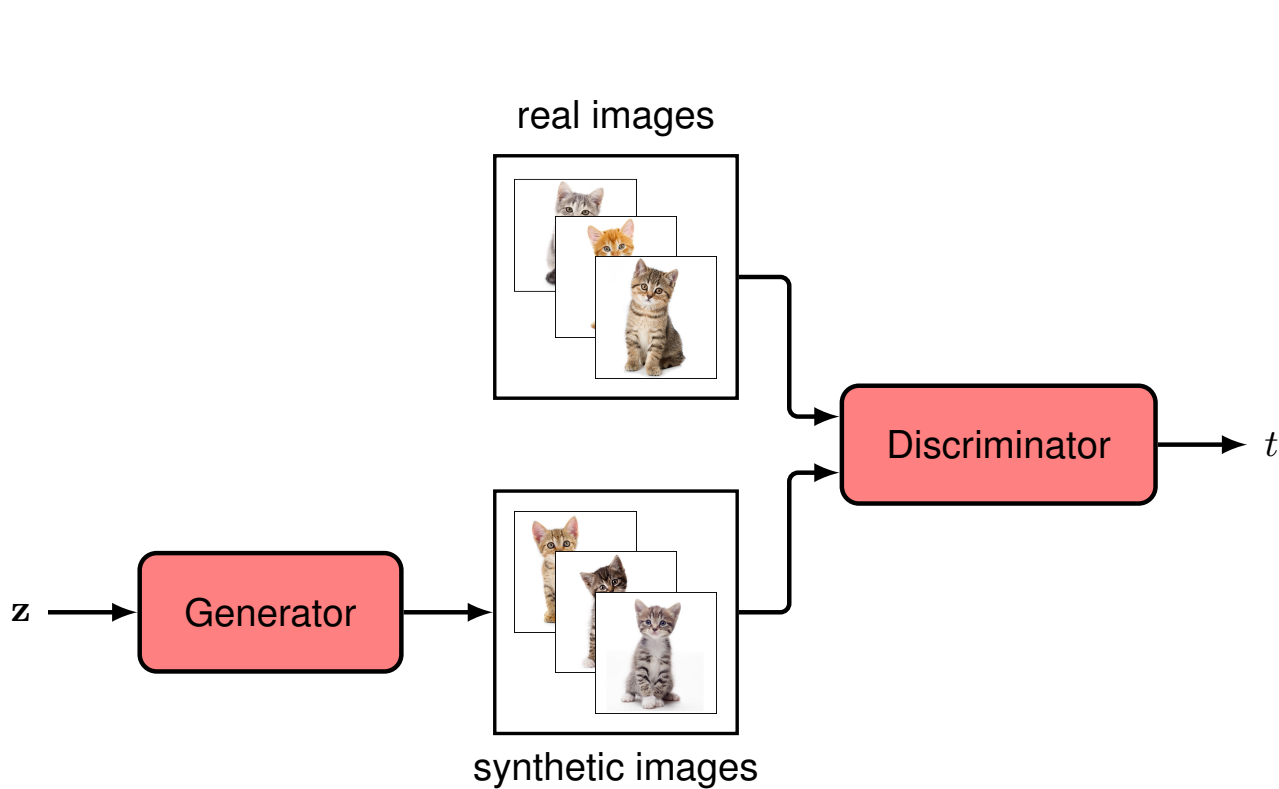
The typhoon signal was upgraded to No. ____

The typhoon signal was upgraded to No. 8

Key components:

- Tokenizer converts text into discrete tokens.
- Transformer decoder predicts the next token given the prefix.

Generative Adversarial Networks (GAN)

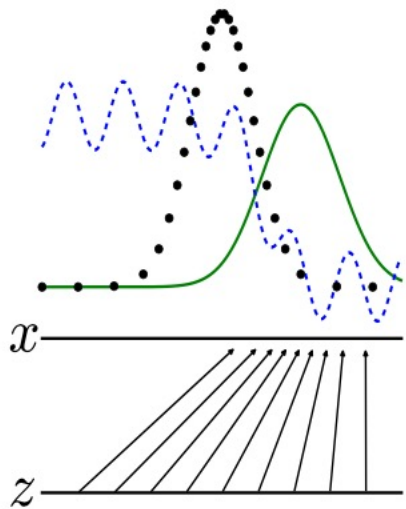


z is some random noise (Gaussian/Uniform). Can be thought as the latent representation of the data

- **Generative model G :**
 - Captures data distribution
 - Fool $D(G(z))$
 - Generate an image $G(z)$ such that $D(G(z))$ is wrong (i.e. $D(G(z)) = 1$)
- **Discriminative model D :**
 - Distinguishes between real and fake samples
 - $D(x) = 1$ when x is a real image, and otherwise $D(x) = 0$

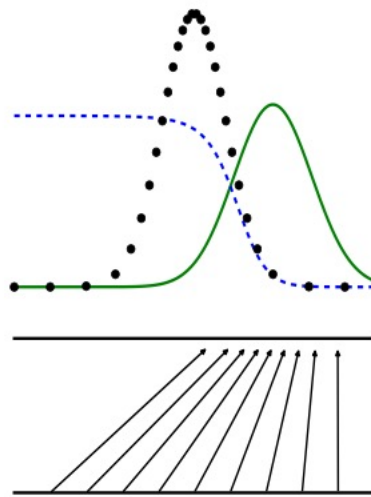
Generative Adversarial Networks (GAN)

- data generating distribution
- generative distribution
- - - - - discriminative distribution



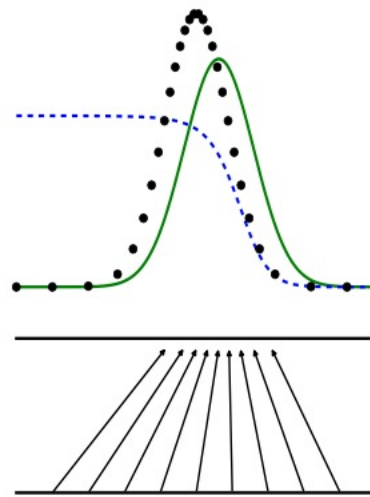
(a)

An adversarial pair near convergence



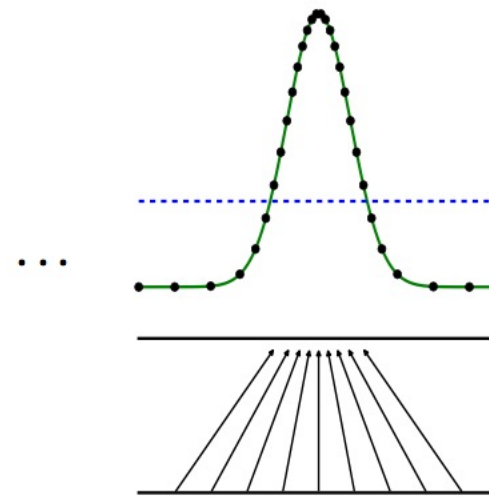
(b)

D is trained to discriminate samples from data



(c)

After an update to G, gradient of D has guided G(z) to flow to regions that are more likely to be classified as data



(d)

After several steps of training, G and D will reach a point at which both cannot improve because $p_g = p_{data}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = 0.5$.

Generative Adversarial Networks (GAN)

- D and G play the following two-player minimax game with value function $V(G, D)$

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

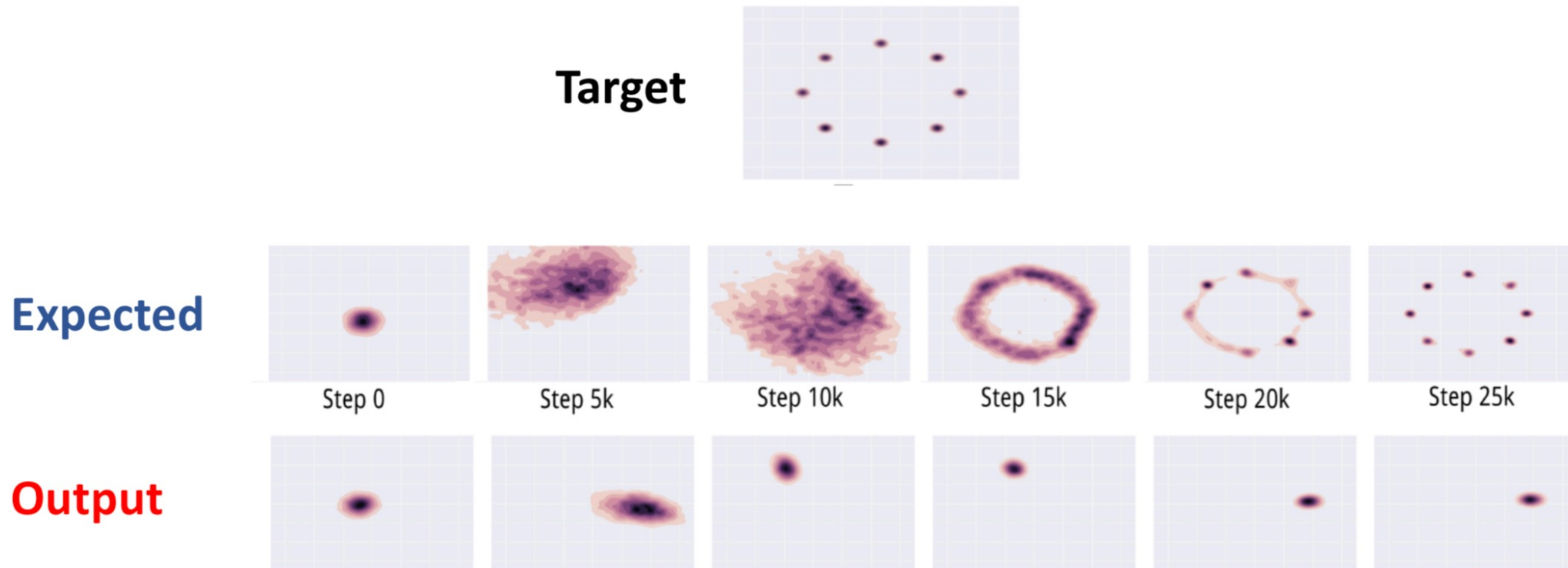
The diagram illustrates the architecture and data flow of a GAN. At the top, a red-bordered box describes the Generator: 'Generator: input = random vector $\mathbf{z}$ ' and 'Output: $G(\mathbf{z})$ as synthetic image'. Below this, the minimax equation is shown. The term $D(\mathbf{x})$ in the equation has a blue box around the D , with a blue arrow pointing down to a blue-bordered box at the bottom. This box describes the Discriminator: 'Discriminator: input = generated/real images' and 'Output: $D(x) = 1$ if real image'. The term $D(G(\mathbf{z}))$ in the equation has a blue box around the D and a red box around the G . A blue arrow points down from the blue D to the bottom Discriminator box, and a red arrow points from the red G to the top Generator box.

Generator: input = random vector $\mathbf{z}$
Output: $G(\mathbf{z})$ as synthetic image

Discriminator: input = generated/real images
Output: $D(x) = 1$ if real image

Generative Adversarial Networks (GAN)

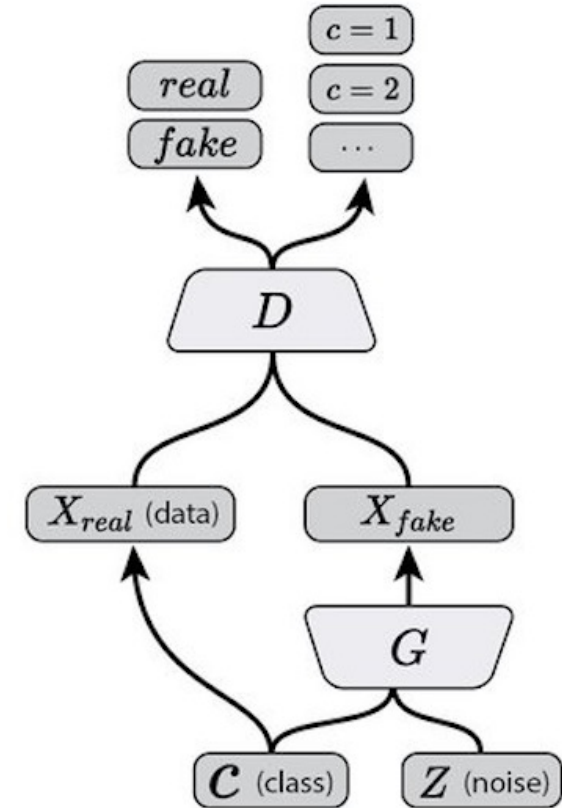
Mode Collapse: Generator fails to output diverse samples



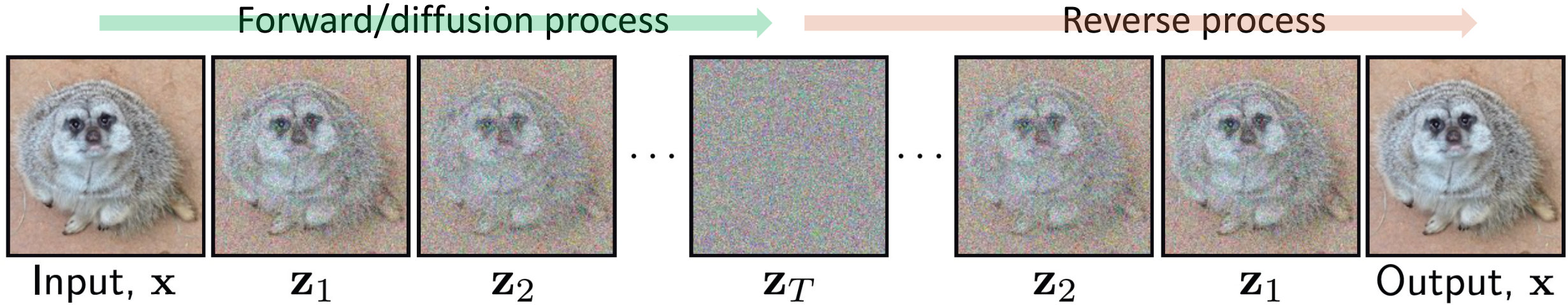
Generative Adversarial Networks (GAN)

Conditional GAN

- Simple modification to the original GAN framework that conditions the model on *additional information* for better multi-modal learning.
- Lends to many practical applications of GANs when we have explicit *supervision* available.



Diffusion Models



A diffusion model consists of an forward process and a reverse process.

The forward process takes a data sample x and maps it through a series of intermediate latent variables $z_1 \dots z_T$.

The reverse process reverses this process; it starts with z_T and maps back through $z_{T-1} \dots z_1$ until it finally (re-) creates a data point x .



SDXL + FreeU



SDXL + FreeU



SDXL + FreeU

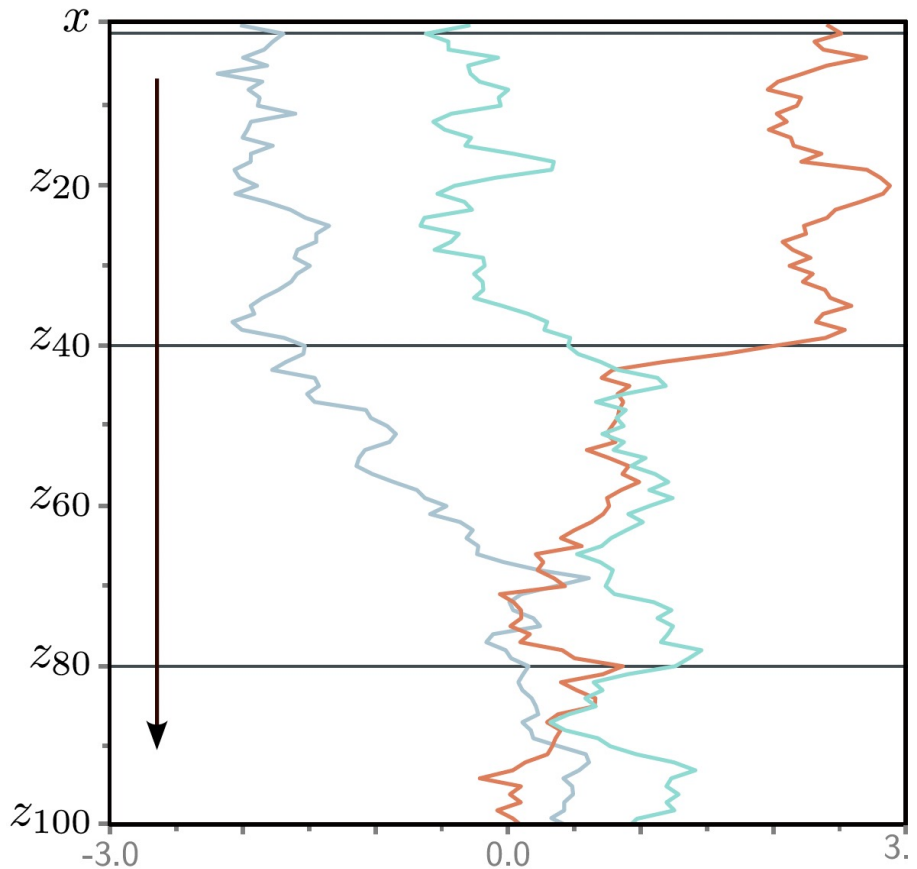


SDXL + FreeU



Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the colorful lights. Many pedestrians walk about.

Diffusion Models: Forward Process



The *diffusion* or *forward* process maps a data example $\mathbf{x}$ through a series of intermediate variables $\mathbf{z}_1, \dots, \mathbf{z}_T$ with the same size as $\mathbf{x}$ according to:

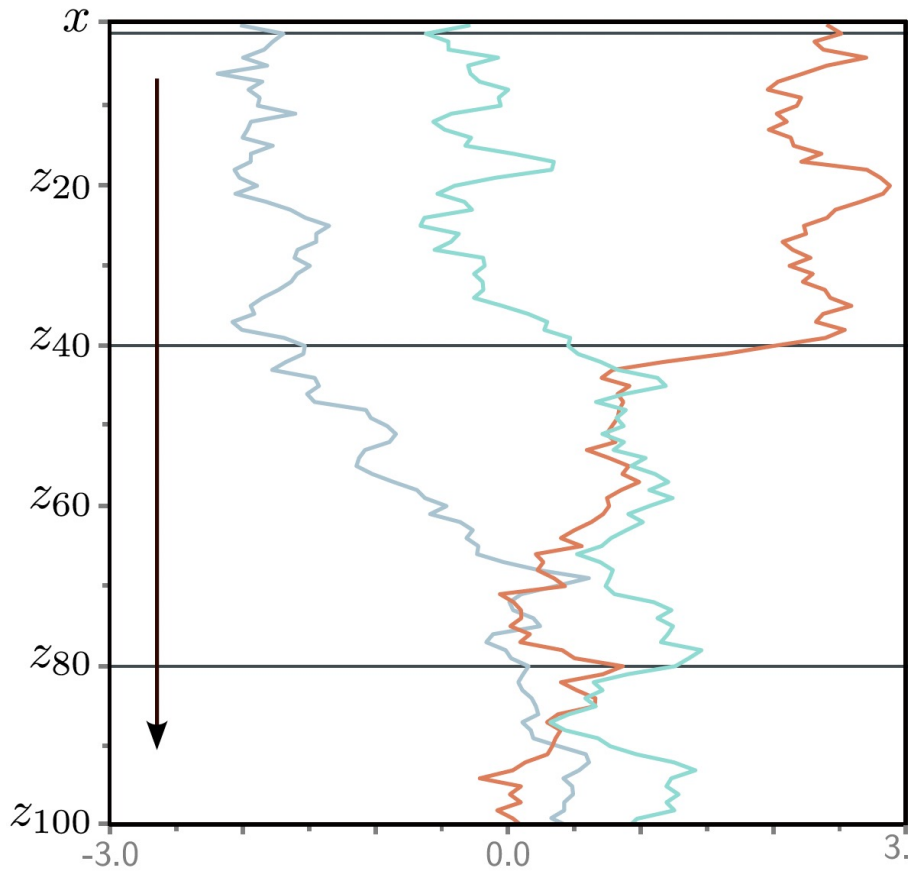
$$\begin{aligned}\mathbf{z}_1 &= \sqrt{1 - \beta_1} \cdot \mathbf{x} + \sqrt{\beta_1} \cdot \epsilon_1 \\ \mathbf{z}_t &= \sqrt{1 - \beta_t} \cdot \mathbf{z}_{t-1} + \sqrt{\beta_t} \cdot \epsilon_t \quad \forall t \in 2, \dots, T,\end{aligned}$$

where ϵ_t is noise drawn from a standard normal distribution.

The first term attenuates the data plus any noise added so far, and the second adds more noise.

The hyperparameters $\beta_t \in [0, 1]$ determine how quickly the noise is blended and are collectively known as the *noise schedule*.

Diffusion Models: Forward Process



The *diffusion* or *forward* process maps a data example $\mathbf{x}$ through a series of intermediate variables $\mathbf{z}_1, \dots, \mathbf{z}_T$ with the same size as $\mathbf{x}$ according to:

$$\begin{aligned}\mathbf{z}_1 &= \sqrt{1 - \beta_1} \cdot \mathbf{x} + \sqrt{\beta_1} \cdot \epsilon_1 \\ \mathbf{z}_t &= \sqrt{1 - \beta_t} \cdot \mathbf{z}_{t-1} + \sqrt{\beta_t} \cdot \epsilon_t \quad \forall t \in 2, \dots, T,\end{aligned}$$

This process is pre-specified and gradually mixes the data with noise until only noise remains.

With sufficient steps T , all traces of the original data are removed, and $q(\mathbf{z}_T | \mathbf{x}) = q(\mathbf{z}_T)$ becomes a standard normal distribution.

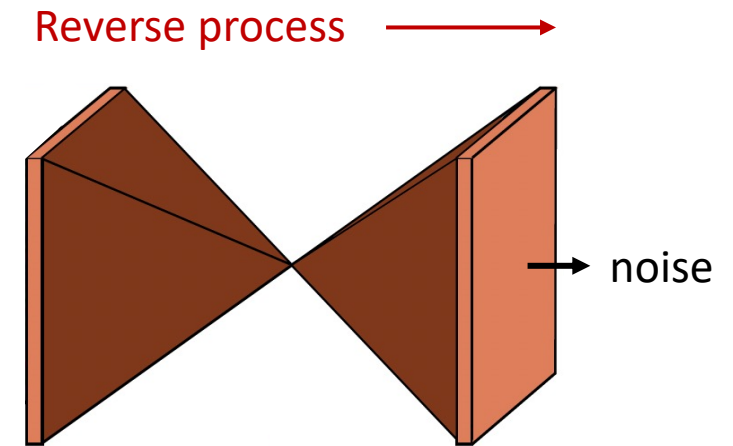
Since this process is prespecified, all the learned parameters are in the reverse process.

Diffusion Models: Reverse Process

The reverse process is learned and passes the data back through the latent variables, removing noise at each stage.

A series of networks are trained to map backward between each adjacent pair of latent variables $\mathbf{z}_t$ and $\mathbf{z}_{t-1}$.

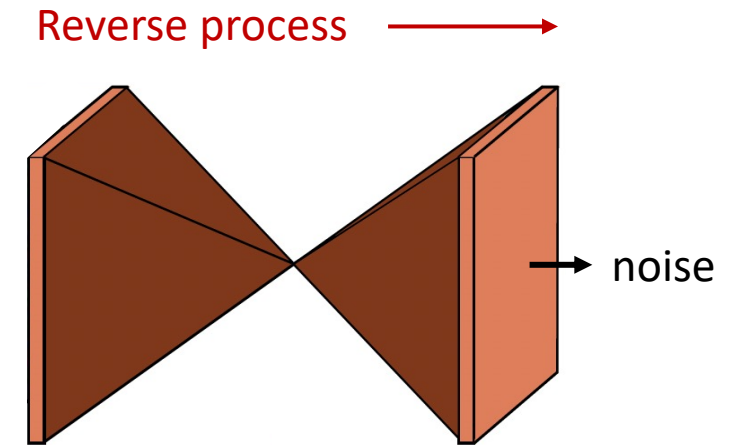
The loss function encourages each network to invert the corresponding forward step.



Diffusion Models: Loss function

The model aims to *predict the noise* that was mixed with the original data example to create the current variable.

$$\begin{aligned} L[\phi_{1...T}] &= \sum_{i=1}^I \sum_{t=1}^T \left\| \mathbf{g}_t[\mathbf{z}_{it}, \phi_t] - \epsilon_{it} \right\|^2 \\ &= \sum_{i=1}^I \sum_{t=1}^T \left\| \mathbf{g}_t \left[\sqrt{\alpha_t} \cdot \mathbf{x}_i + \sqrt{1 - \alpha_t} \cdot \epsilon_{it}, \phi_t \right] - \epsilon_{it} \right\|^2 \end{aligned}$$



We have rewritten $\mathbf{z}_t$ using the diffusion kernel $\mathbf{z}_t = \sqrt{\alpha_t} \cdot \mathbf{x} + \sqrt{1 - \alpha_t} \cdot \epsilon$, in the second line

This leads to straightforward algorithms for both training the model and sampling.

Diffusion Models: Reverse Process

After training, new examples are generated by sampling noise vectors $\mathbf{z}_T$ and passing them through the reverse process.

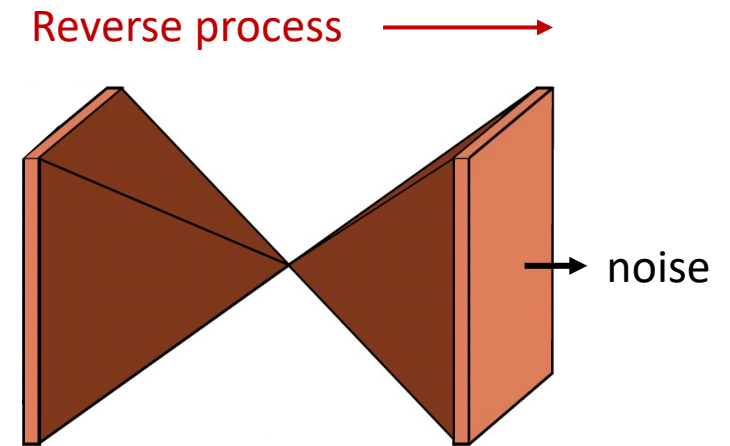
Finally, generate $\mathbf{x}$ from $Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)$

⋮

Sample $\mathbf{z}_{98}$ from $Pr(\mathbf{z}_{98}|\mathbf{z}_{99}, \boldsymbol{\phi}_{99})$ and so on

Sample $\mathbf{z}_{99}$ from $Pr(\mathbf{z}_{99}|\mathbf{z}_{100}, \boldsymbol{\phi}_{100})$,

We start by drawing $\mathbf{z}_{100}$ from $Pr(\mathbf{z}_{100})$.



Diffusion Models - Training

Algorithm - Diffusion model training

Input: Training data $\mathbf{x}$

Output: Model parameters ϕ_t

repeat

```
  for  $i \in \mathcal{B}$  do                                // For every training example index in batch
     $t \sim \text{Uniform}[1, \dots, T]$                 // Sample random timestep
     $\epsilon \sim \text{Norm}[\mathbf{0}, \mathbf{I}]$                 // Sample noise
     $\ell_i = \left\| \mathbf{g}_t \left[ \sqrt{\alpha_t} \mathbf{x}_i + \sqrt{1 - \alpha_t} \epsilon, \phi_t \right] - \epsilon \right\|^2$  // Compute individual loss
```

Accumulate losses for batch and take gradient step

until converged

Advantages

- (i) simple to implement
- (ii) naturally augments the dataset; we can reuse every original data point $\mathbf{x}_i$ as many times as we want at each time step with different noise instantiations ϵ

Diffusion Models - Sampling

Algorithm - Sampling

Input: Model, $\mathbf{g}_t[\bullet, \phi_t]$
Output: Sample, $\mathbf{x}$
 $\mathbf{z}_T \sim \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$ // Sample last latent variable
for $t = T \dots 2$ **do**
 $\hat{\mathbf{z}}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}} \mathbf{g}_t[\mathbf{z}_t, \phi_t]$ // Predict previous latent variable
 $\epsilon \sim \text{Norm}_{\epsilon}[\mathbf{0}, \mathbf{I}]$ // Draw new noise vector
 $\mathbf{z}_{t-1} = \hat{\mathbf{z}}_{t-1} + \sigma_t \epsilon$ // Add noise to previous latent variable
 $\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}} \mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}} \mathbf{g}_1[\mathbf{z}_1, \phi_1]$ // Generate sample from $\mathbf{z}_1$ without noise

To sample $\mathbf{z}_{t-1} \sim \text{Pr}(\mathbf{z}_{t-1} | \mathbf{z}_t, \phi_t)$, take the mean $\hat{\mathbf{z}}_{t-1}$ and add to it random perturbation ϵ with variance sigma σ_t , akin to drawing a sample from a distribution $\text{Norm}_{\mathbf{z}_{t-1}}[\hat{\mathbf{z}}_{t-1}, \sigma_t^2 \mathbf{I}]$

Disadvantages

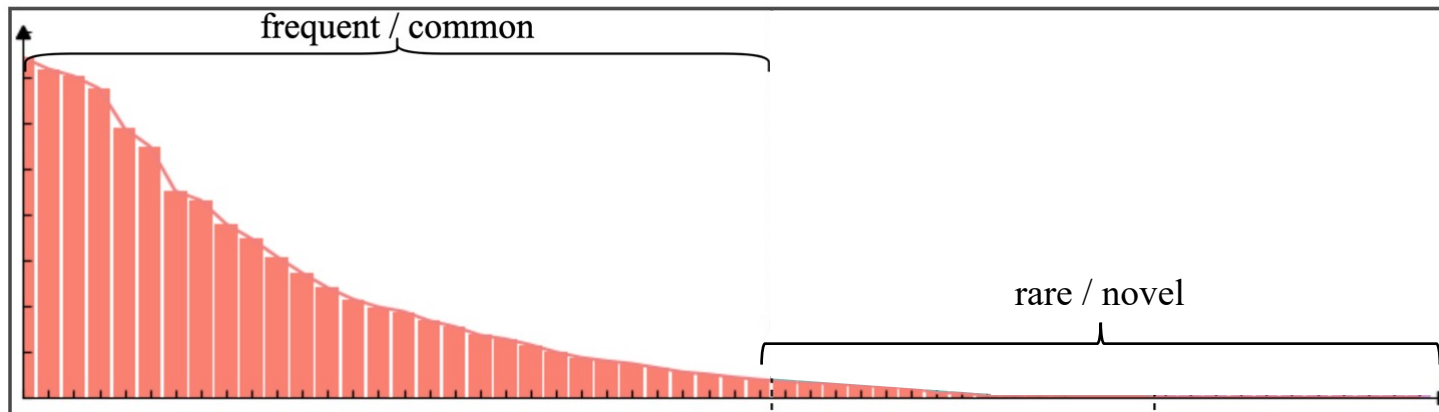
Requires serial processing of many neural networks $\mathbf{g}_t[\mathbf{z}_t, \phi_t]$ and is hence time-consuming!

Summary

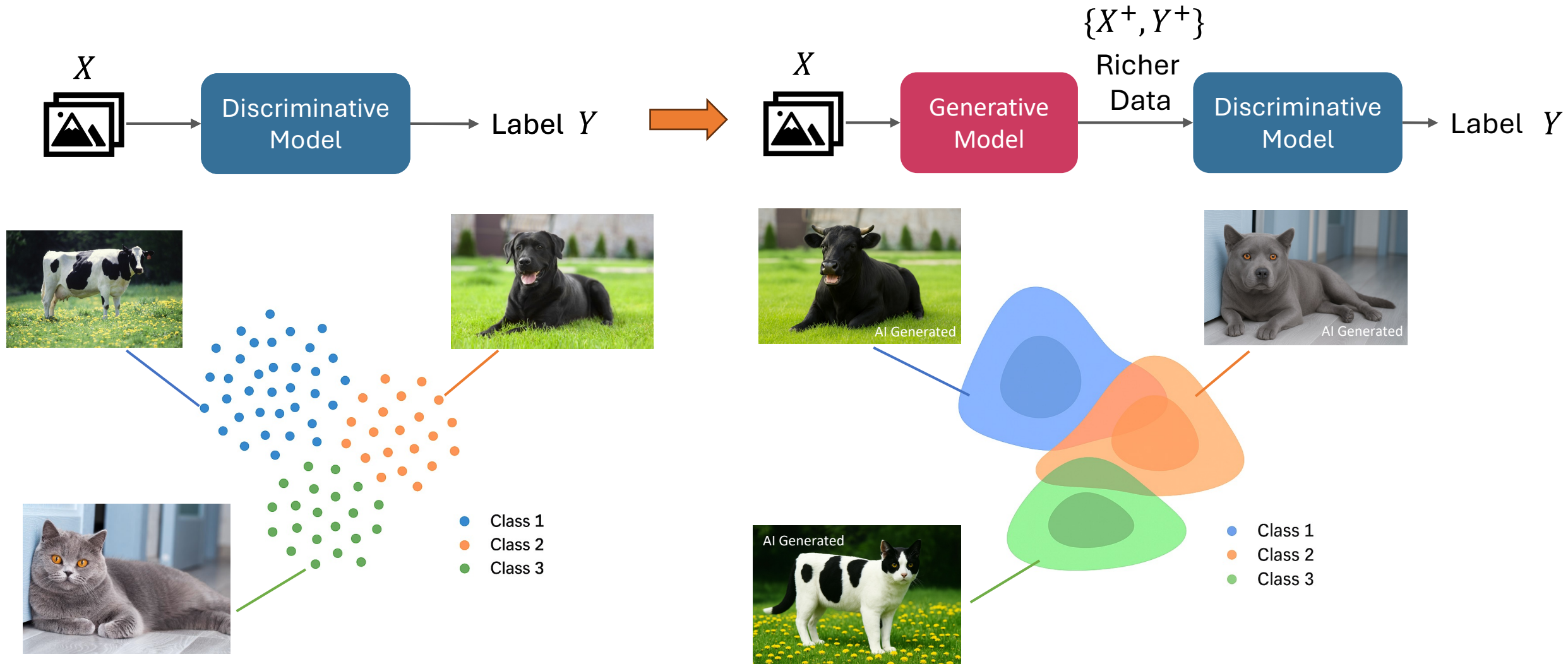
- VAEs
 - Can infer a useful latent representation for a dataset
 - Better generators than autoencoders due to latent space regularization
 - Lower quality reconstructions and generated samples (usually blurry)
- Autoregressive models
 - Model data as a sequence → predict the next element/token given history
 - Slow generation for long sequences; limited global coherence for images
- GANs
 - Can not find a latent representation for a given sample (no encoder)
 - Usually better generators than VAEs
 - Unstable training
- Diffusion models
 - Stable optimization
 - High diversity and fidelity
 - Flexible conditioning (e.g., ControlNet)
 - Scales well with compute

Generative Models as Data Generator

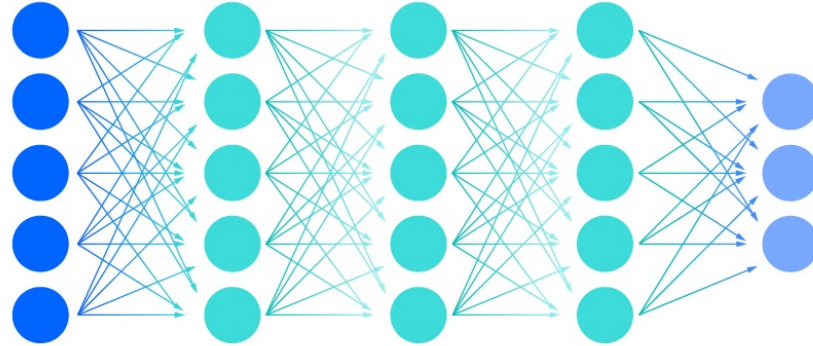
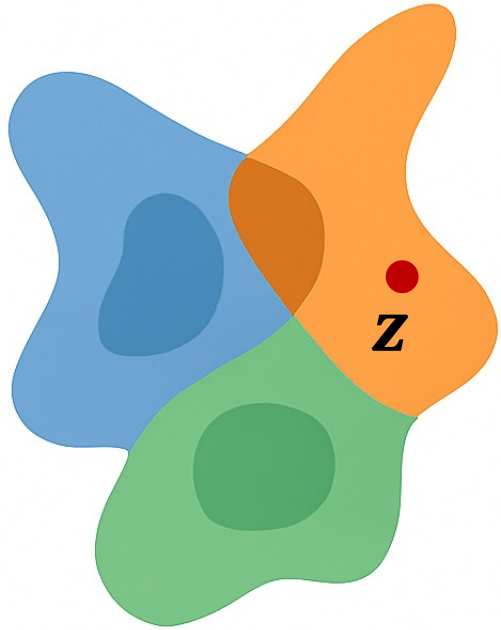
- Manual annotation is laborious and expensive
- The size can be small in some classes (long-tailed) and thus diversity is a problem



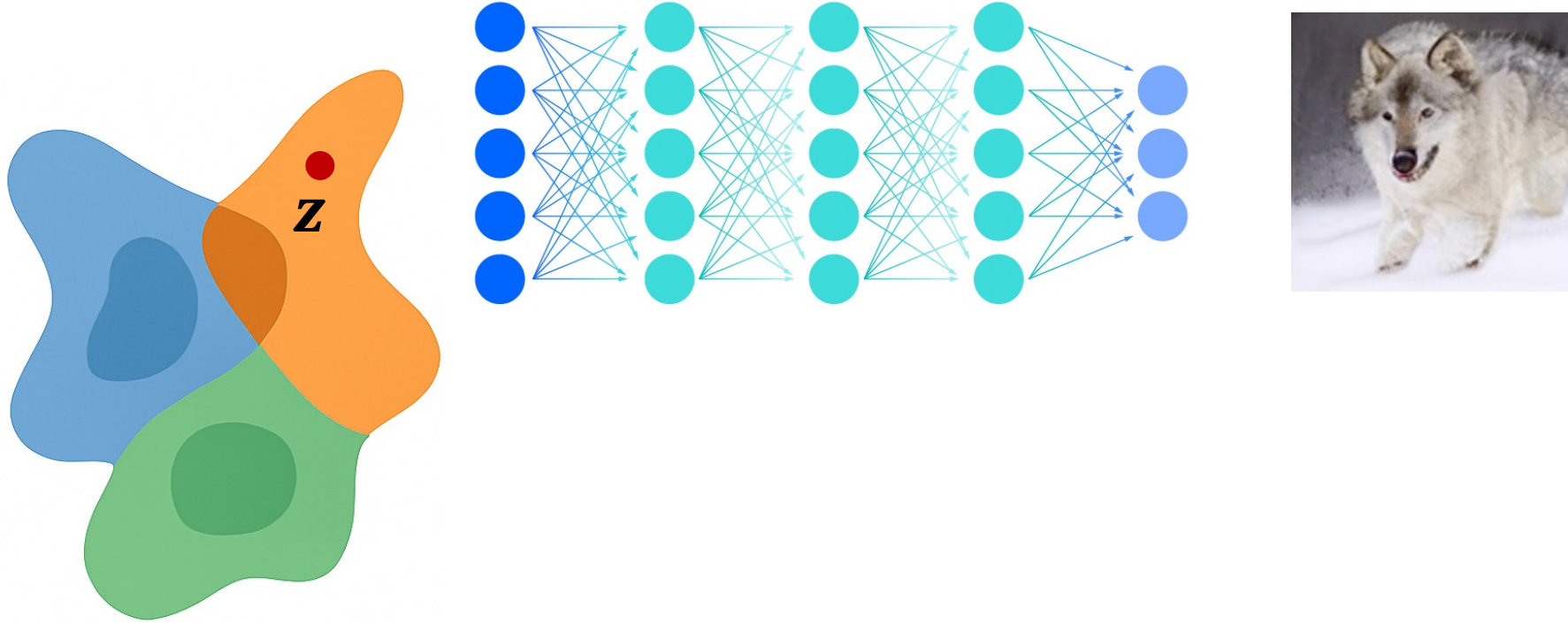
Generative Models as Data Generator



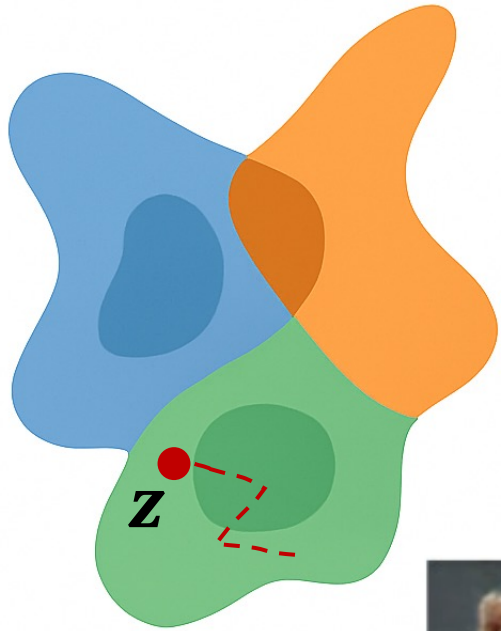
Generative Models as Data Generator



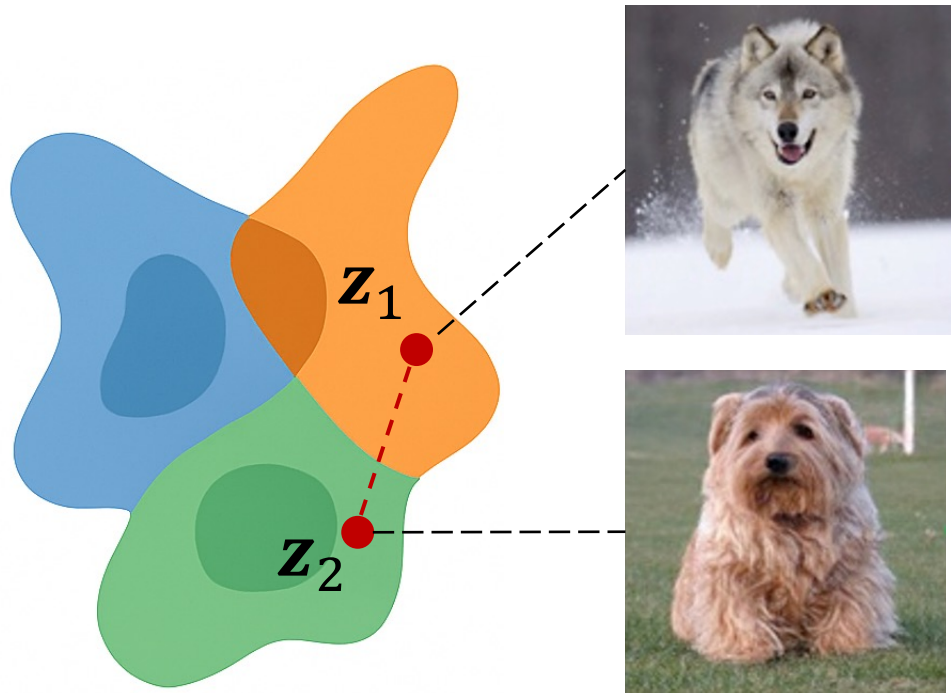
Generative Models as Data Generator



Generative Models as Data Generator



Generative Models as Data Generator



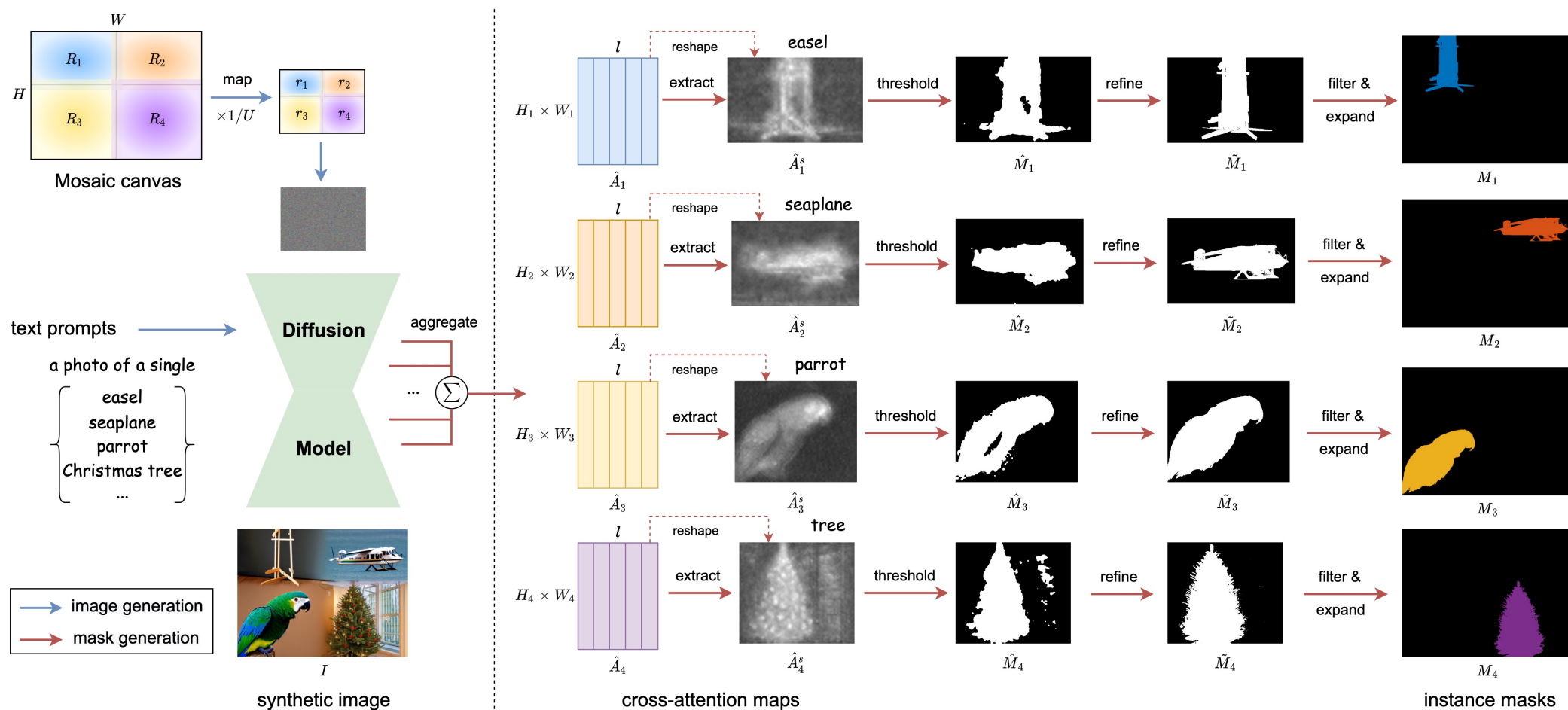
You can manipulate the latent variable to change the visual properties

- Len parameters
- Camara angle
- Motion etc.



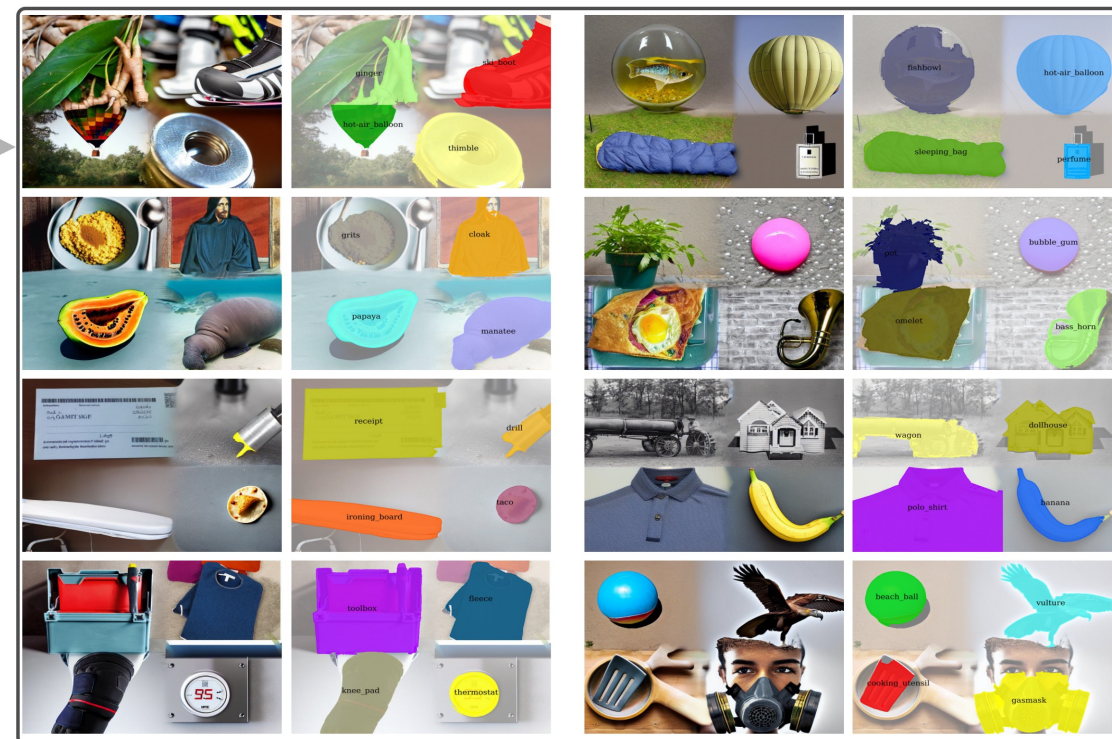
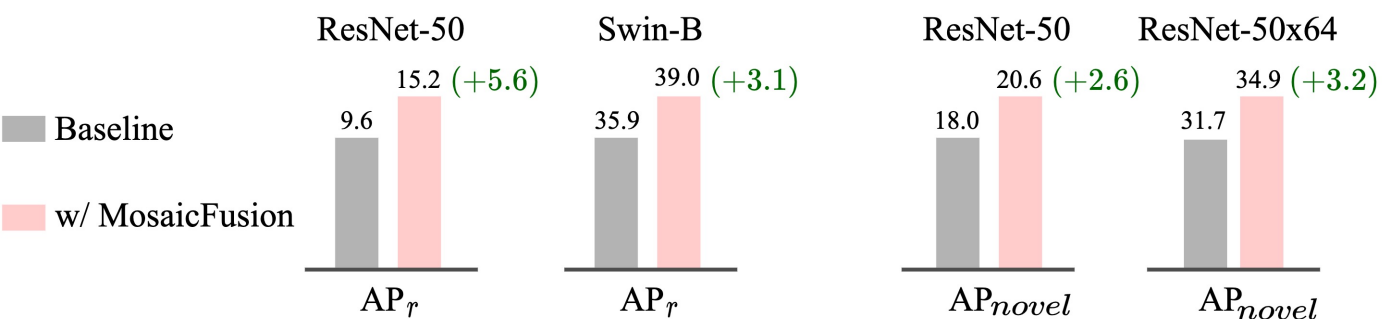
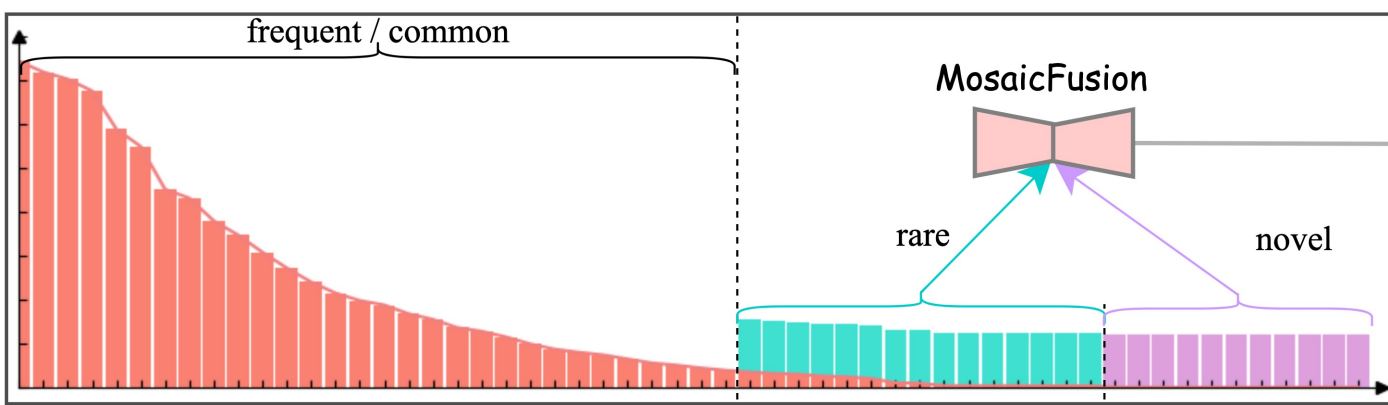
Generative Models as Data Generator

Generative models provide labels Y too apart from data X



Generative Models as Data Generator

We can expand existing instance segmentation datasets (rare and novel categories) to train a model



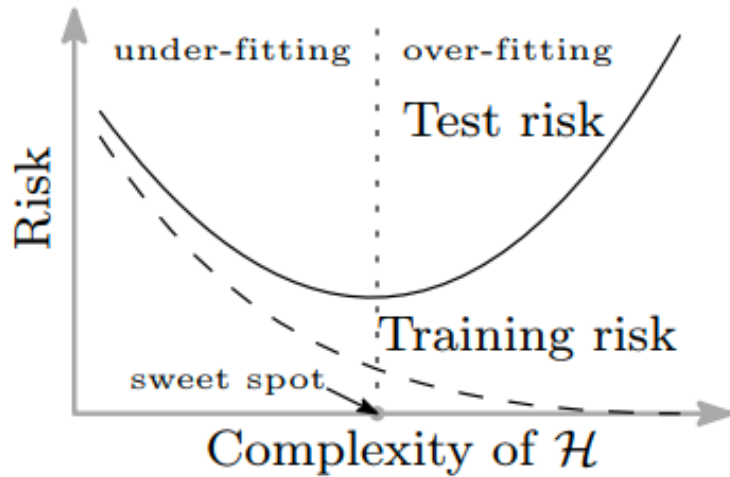
generated instances with masks

Rise of Foundation Models

What is a Foundation Model?

- Definition: Single model pretrained on large, diverse data for broad tasks
- One general model instead of many task-specific models
- Pretraining on diverse datasets gives broad capabilities
- Performance increases with data, model size, compute (“scaling laws”)

What We Used to Know

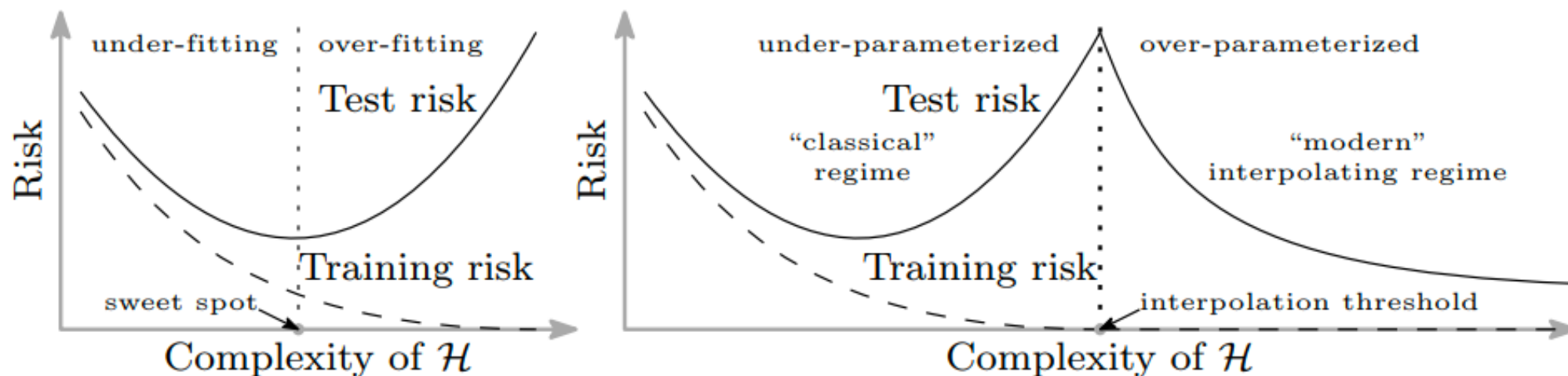


Underfitting - when a model unable to capture the underlying pattern of the data

Overfitting - when our model is too complex and too specialized on a small number of training data

Early stopping is a common regularization technique to avoid over-fitting

Deep Double Descent



Let p be the number of parameters of the learner and n the number of training data points

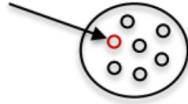
- Under-parametrized functions are defined by $p \ll n$
- Over-parametrized functions are defined by $p \gg n$
- Interpolation threshold is the minimal capacity needed to overfit the training set, $p = n$

In the modern ML regime, over-fitting is considered beneficial, and over-parametrized functions with high model complexity lead to successful generalization.

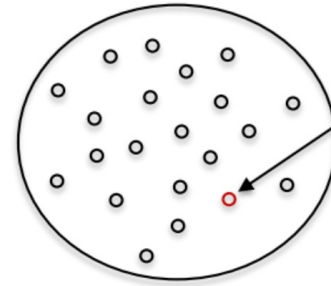
Deep Double Descent

- When $p = n$, the model possesses just enough parameters to over-fit all the training data. However, it also exhibits a significant variance, making it unable to generalize
- When $p \gg n$, the model has much greater parameters than the number of training data. In this regime, the learner $f_{\theta}(x)$ continues to over-fit but critically, the L_2 norm of its parameters $\|\theta\|_2$ is significantly minimized by SGD, effectively reducing the model capacity (regularization effect).

Function with the
smallest norm
 $\|\theta\|_2 = 8.7$



Space of functions
that just overfit
(interpolation point)



Function with the
smallest norm
 $\|\theta\|_2 = 0.3$

Space of functions that overfit
and possess high capacity
(larger space $\Rightarrow$ more functions $\Rightarrow$ lower
 $\|\theta\|_2$ value than at interpolation point)

Deep Double Descent

- The double descent learning mechanism is applicable to both non-linear and linear ML models.
 - This includes techniques such as decision trees, kernel methods, and deep learning.
- The phenomenon is independent of the nature of the datasets involved.
- One important ML principle is that more data provides better results
 - Both theory and empirical experiments align on this principle
 - However, this trend continually increases the critical threshold p^* of required network parameters for the double descent phenomenon to manifest.

Neural Scaling Law

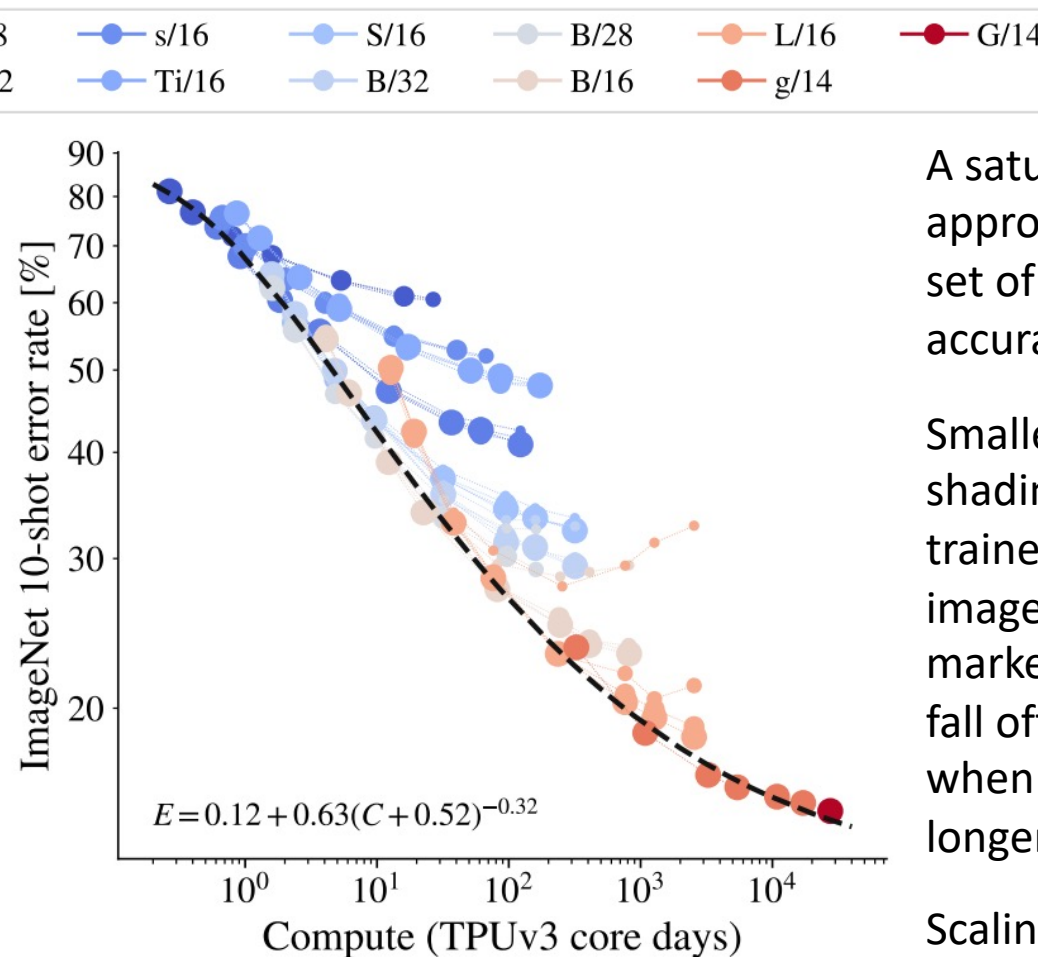
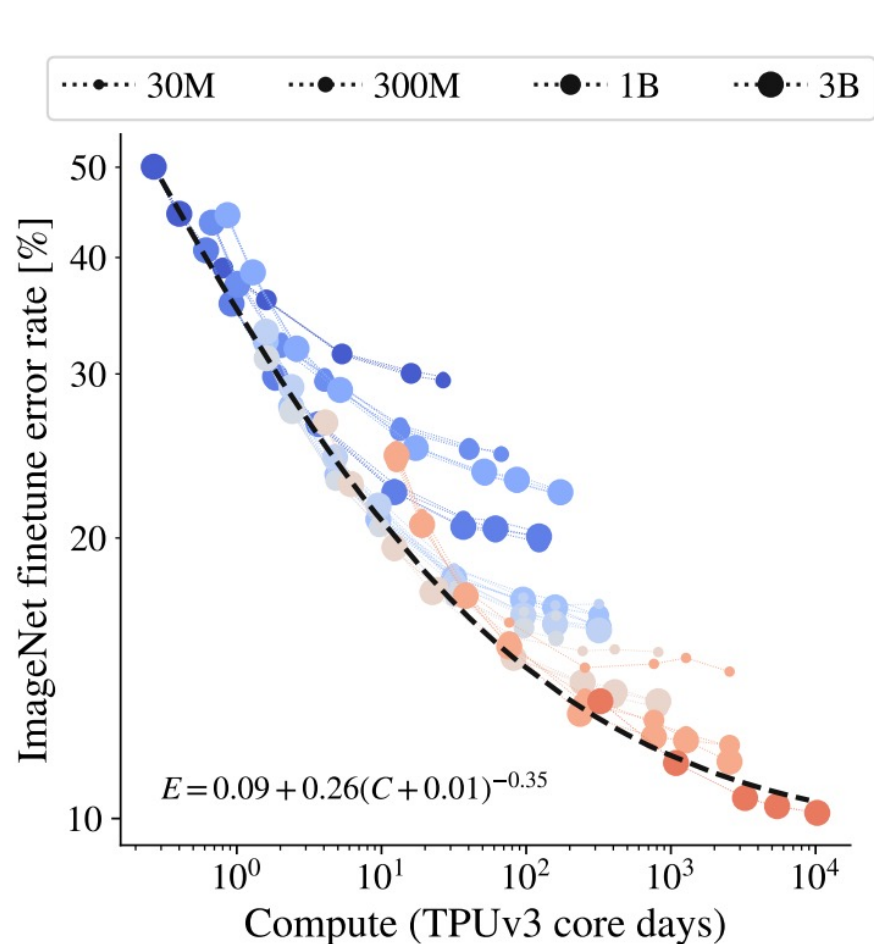
- Neural scaling law - **empirical** relationships between
 - **Model size**: Number of parameters.
 - **Dataset size**: Amount of training data.
 - **Compute**: Training steps or FLOPs (floating-point operations per second).
- Performance (e.g., loss or accuracy) improves **predictably** with increases of the above factors.
- Often follows a power law $L = A \times N^{-\alpha} + B$

L : Loss

N : Scaling variable (e.g., model size or dataset size)

α, A, B : Empirically determined constants, α is the scaling exponent that determines the impact of variable

Neural Scaling Law



Representation quality, measured as ImageNet finetune and linear 10-shot error rate, as a function of total training compute

A saturating power-law approximates the best set of solutions fairly accurately

Smaller models (blue shading), or models trained on fewer images (smaller markers), saturate and fall off the frontier when trained for longer

Scaling up compute, model and data together improves representation quality

Neural Scaling Law

- Observations

- Resources should be proportionally allocated across model size, dataset size, and compute.
- Performance gains taper off if additional data is not diverse or useful.
- Large models trained on diverse datasets exhibit strong transfer learning capabilities.
- Larger models may develop new capabilities not present in smaller ones (e.g., language understanding or reasoning)

Transformers

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

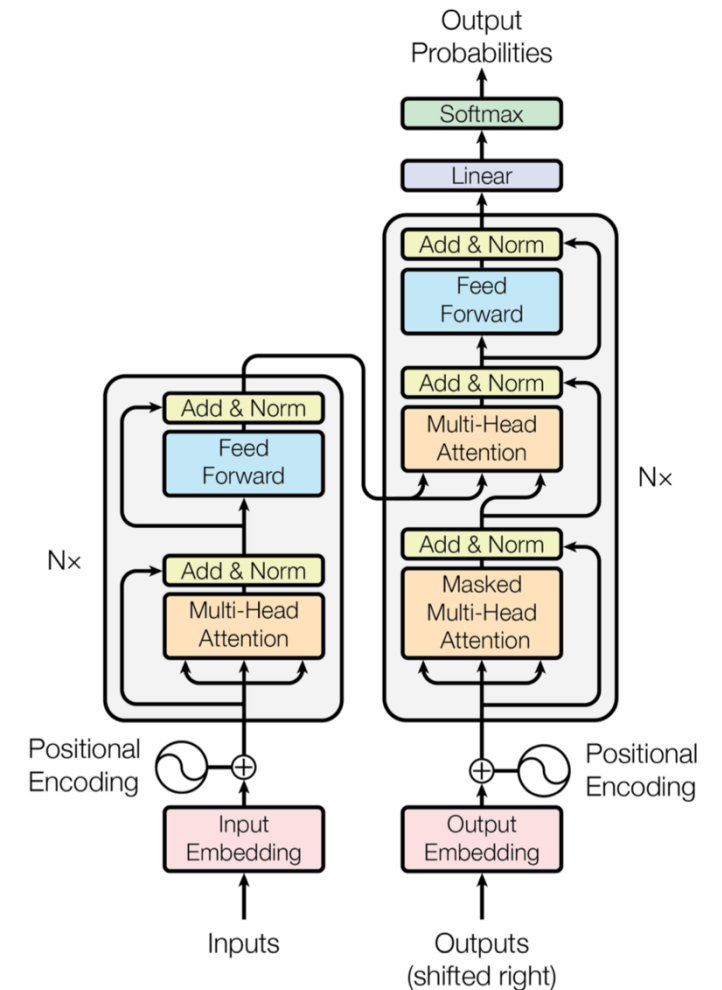
Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaizer@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

“Don’t force everything into a single recurrent state. A better approach is to keep all past representations and let the model attend to them.”

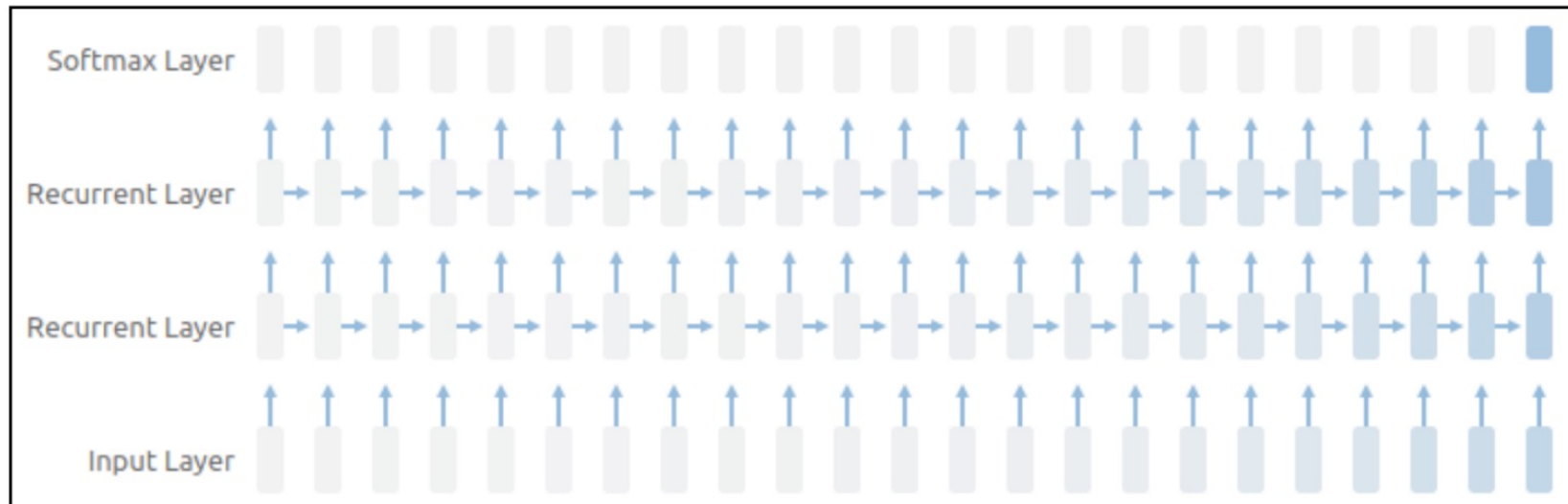


Transformers

Issues with RNN:

During gradient back-propagation learning of RNN, the gradient can end up being **multiplied a large number of times** (as many as the number of time steps) by the weight matrix associated with the connections between the neurons of the recurrent hidden layer.

If the weights in this matrix are **small**, the recursive derivative can lead to a situation called **vanishing gradients** where the gradient signal gets so small that **learning either becomes very slow or stops working altogether**



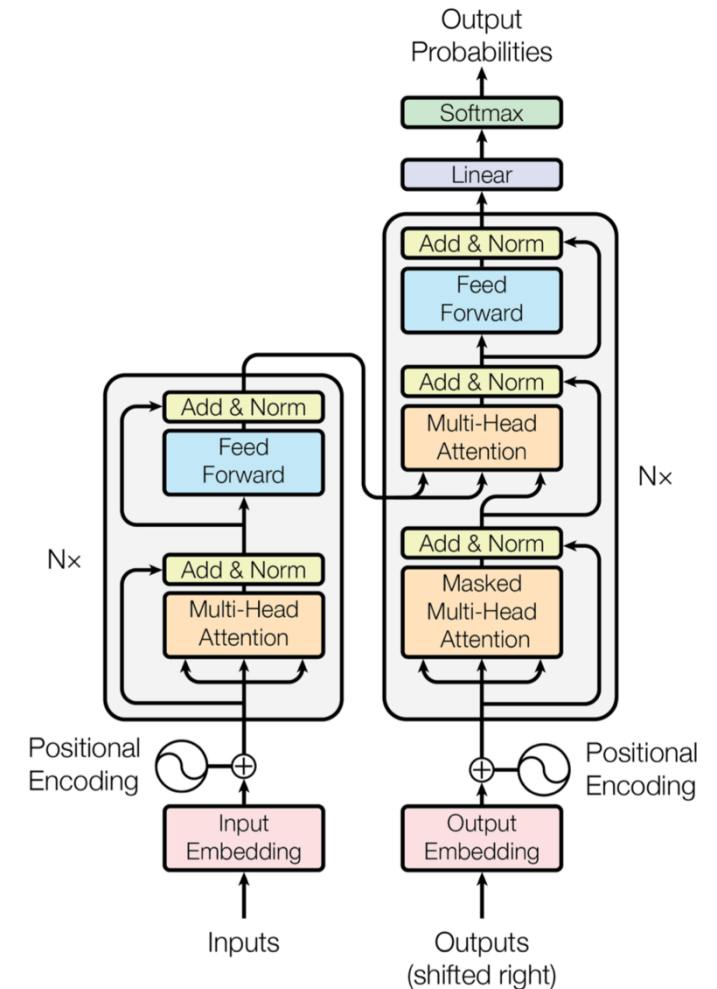
Contribution from the earlier steps becomes insignificant in the gradient

Transformers

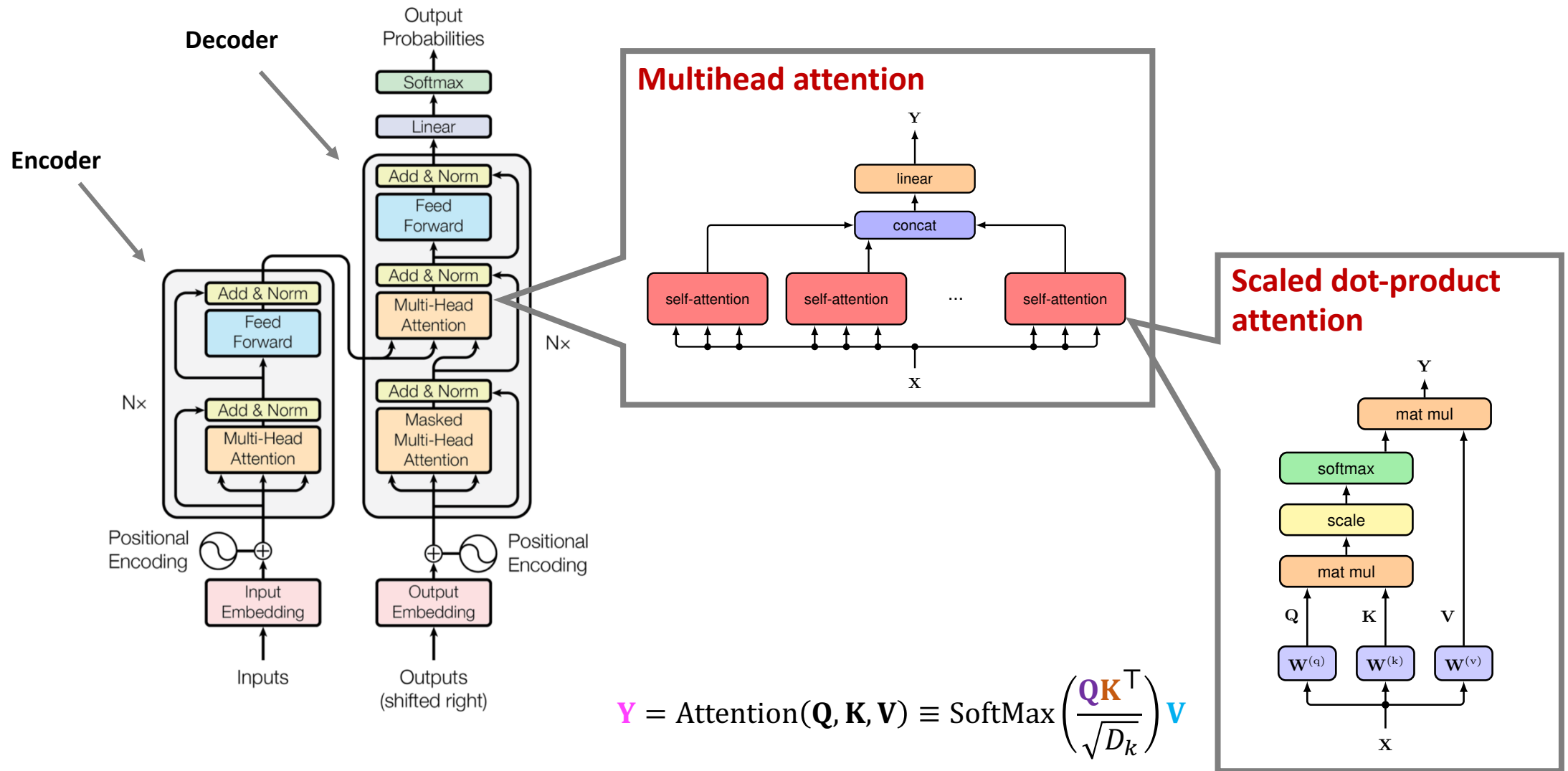
(Original) Transformer is an architecture for transforming one sequence into another one with the help of two parts (Encoder and Decoder).

Transformer advantages:

- Attention-based models allow modeling of dependencies without regard to their distance in the input or output sequences
- Uniform architecture for all modalities
- Emergent abilities with scale – scaling law



Transformers



Self-attention

Key, Value, Query (analogy with information retrieval)

Choose which movie to watch in an online movie streaming service.

Each movie is associated with a list of attributes, such as the genre (comedy, action, etc.), the names of the leading actors, the length of the movie, and so on. The user could then search through a catalogue to find a movie that matches their preferences.

The corresponding movie file itself is called a **value**.

We could automate this by encoding the attributes of each movie in a vector called the **key**.

The user could then provide their own personal vector of values for the desired attributes, which we call the **query**.

value



key

Comedy
Romantic
Cartoon

Action
Sci-fi

Horror
Mystery

query

I want to watch an *action* movie tonight

Self-attention

Key, Value, Query

The movie service could then compare the **query** vector with all the **key** vectors to find the best match and send the corresponding movie to the user in the form of the **value** file.

We can think of the user 'attending' to the particular movie whose **key** most closely matches their **query**.

This would be considered a form of hard attention in which a single **value** vector is returned.

For the transformer, we generalize this to soft attention in which we use continuous variables to measure the degree of match between queries and keys and we then use these variables to weight the influence of the value vectors on the outputs.

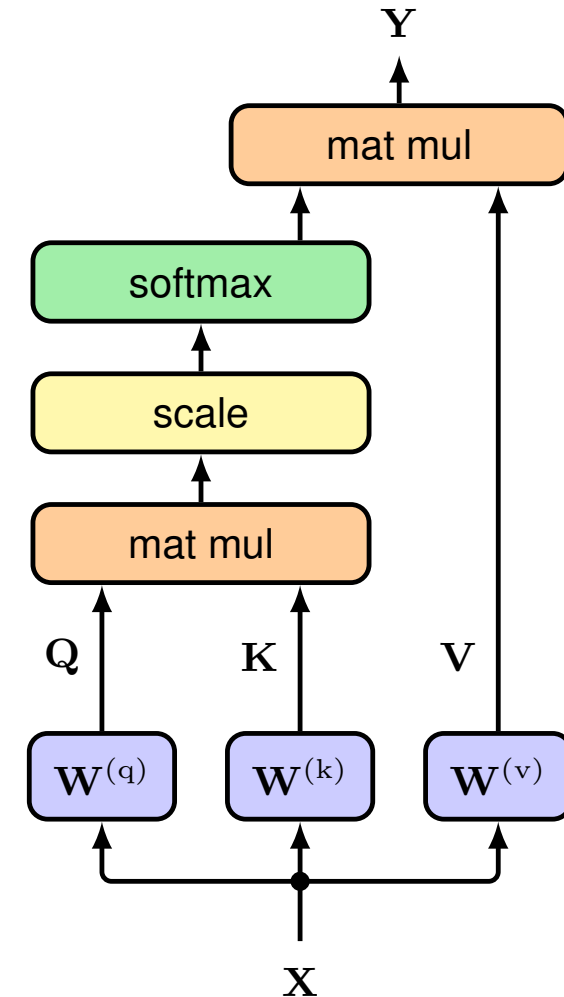


Self-attention

Self-attention = Scaled dot-product attention

The output is a weighted sum of the values, where the weight assigned to each value is determined by the dot-product of the query with all the keys

$$\mathbf{Y} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \equiv \text{SoftMax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}}\right)\mathbf{V}$$



Information flow in a scaled dot-product self-attention. This structure constitutes a single attention 'head'.

Multi-head self-attention

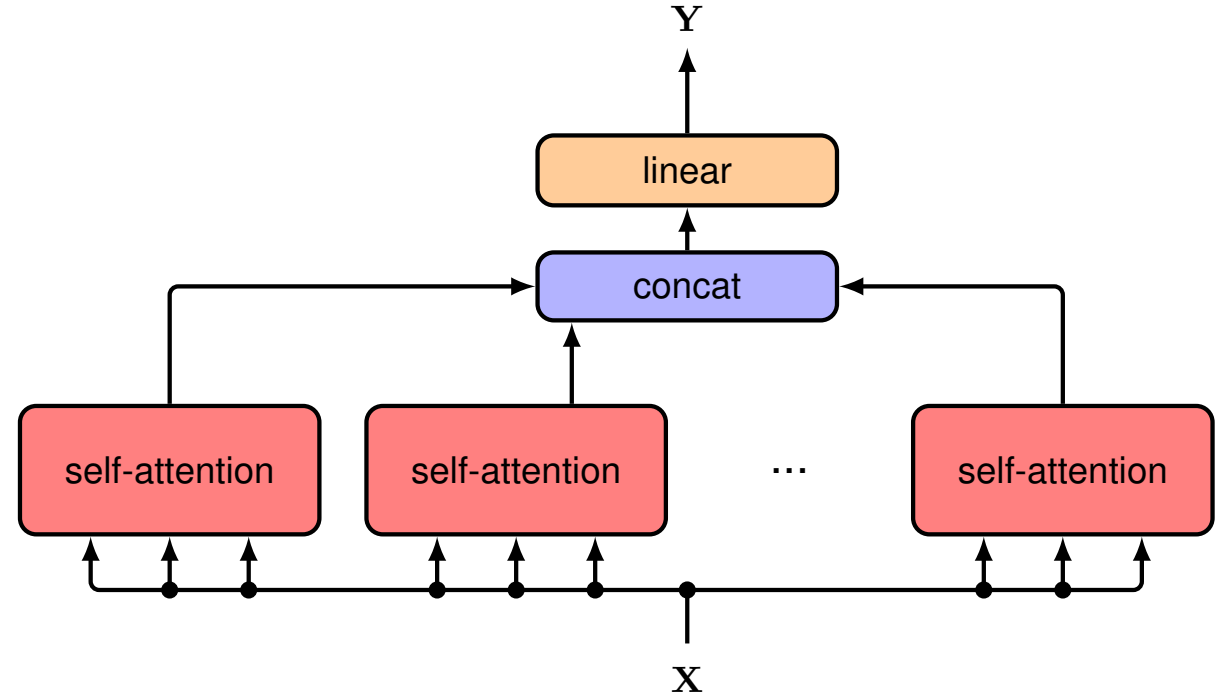
Multi-Head Self-Attention

There might be multiple patterns of attention that are relevant at the same time.

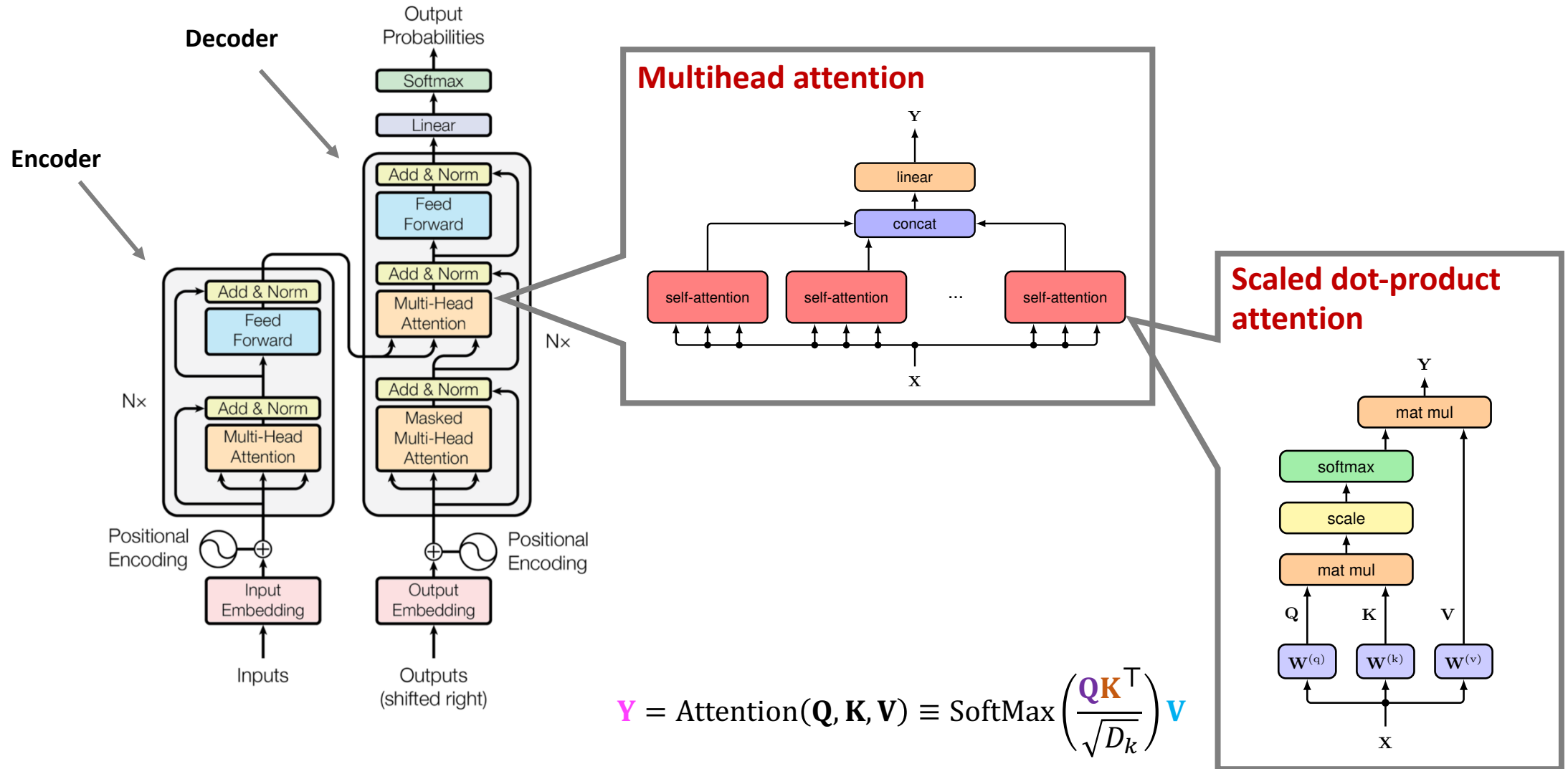
Using a single attention head can lead to **averaging over** these effects.

Instead we can use multiple attention heads in parallel.

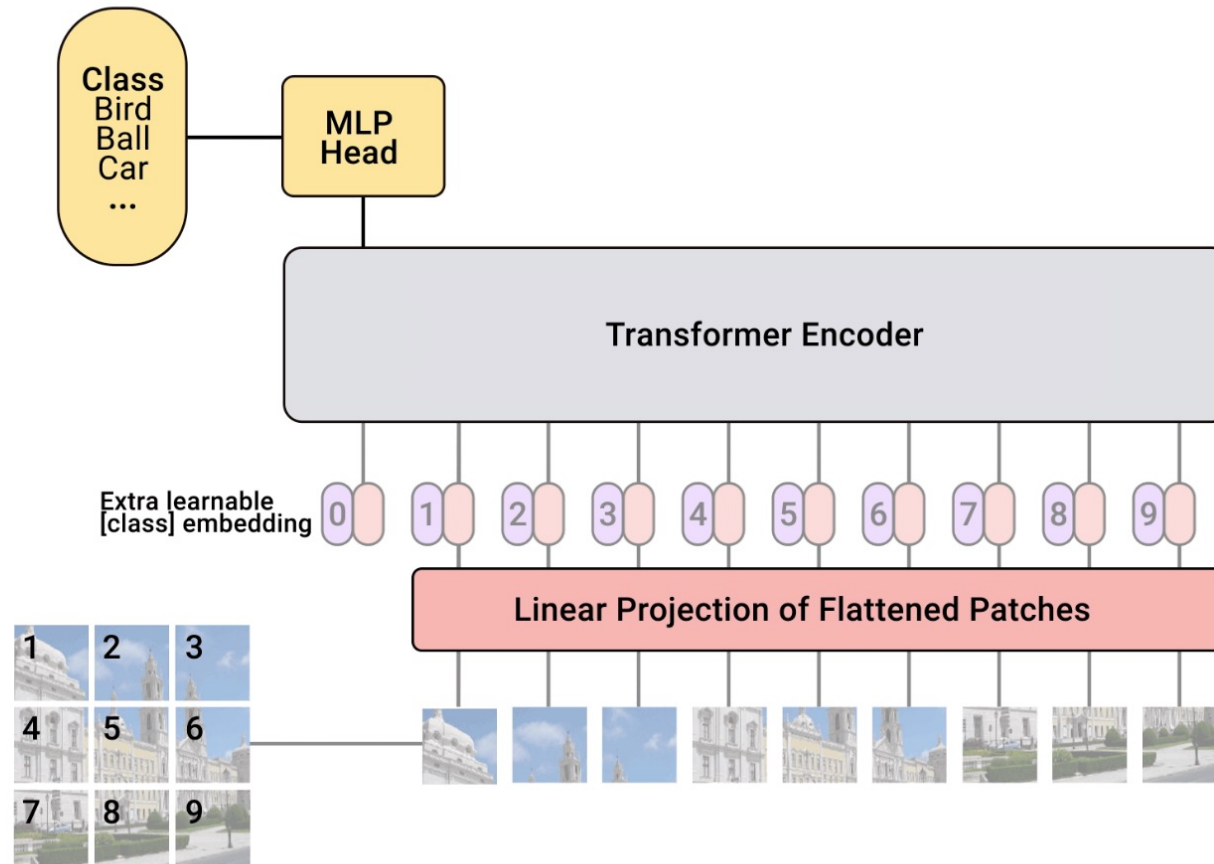
These consist of **identically structured copies of the single head**, with **independent learnable parameters** that govern the calculation of the query, key, and value matrices.



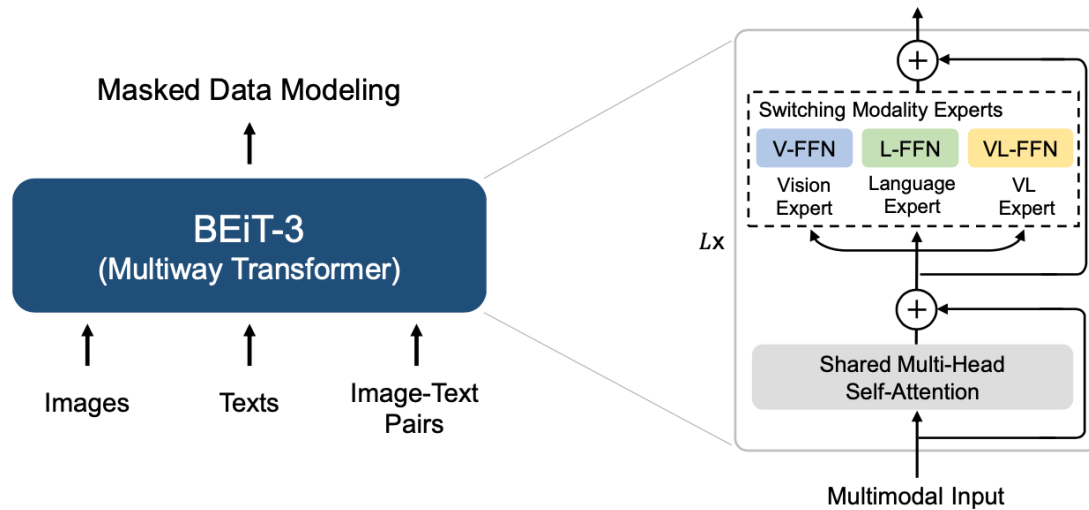
Transformers



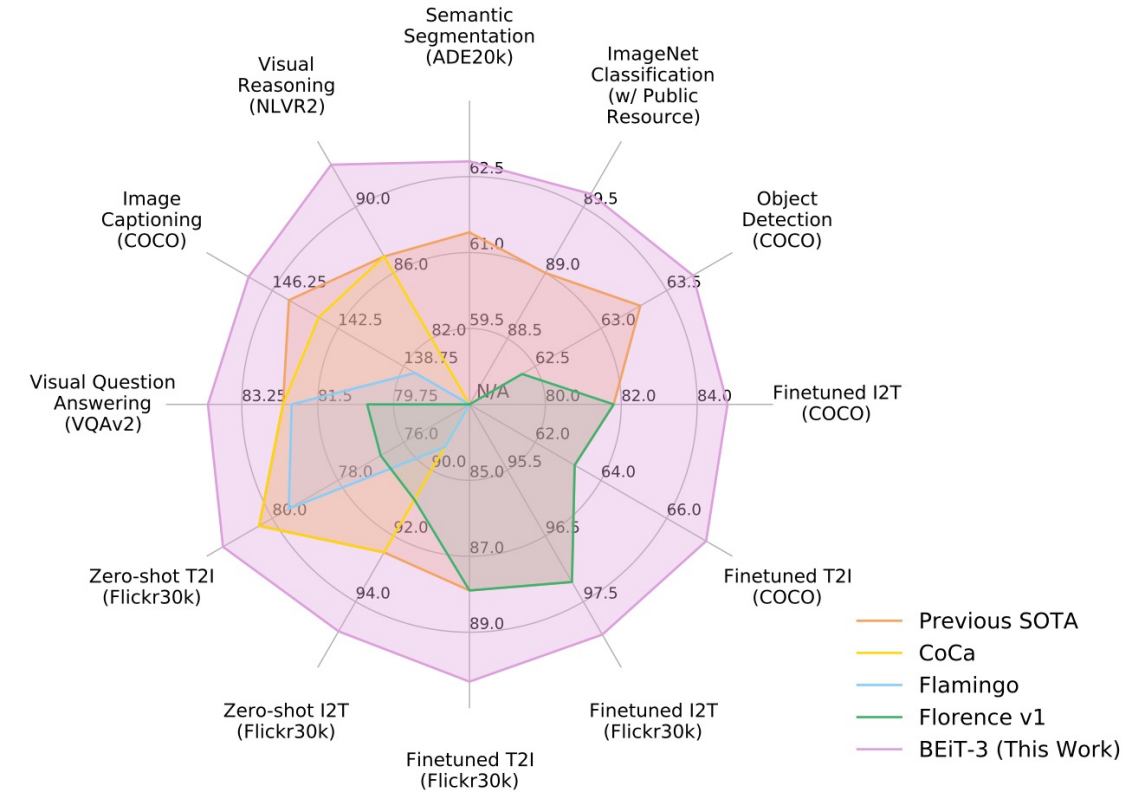
Vision Transformer (ViT)



General-purpose Multimodal Foundation Model



Masked data modelling on monomodal (i.e., images, and texts) and multimodal (i.e., image-text pairs) data with a shared Multiway Transformer as the backbone network



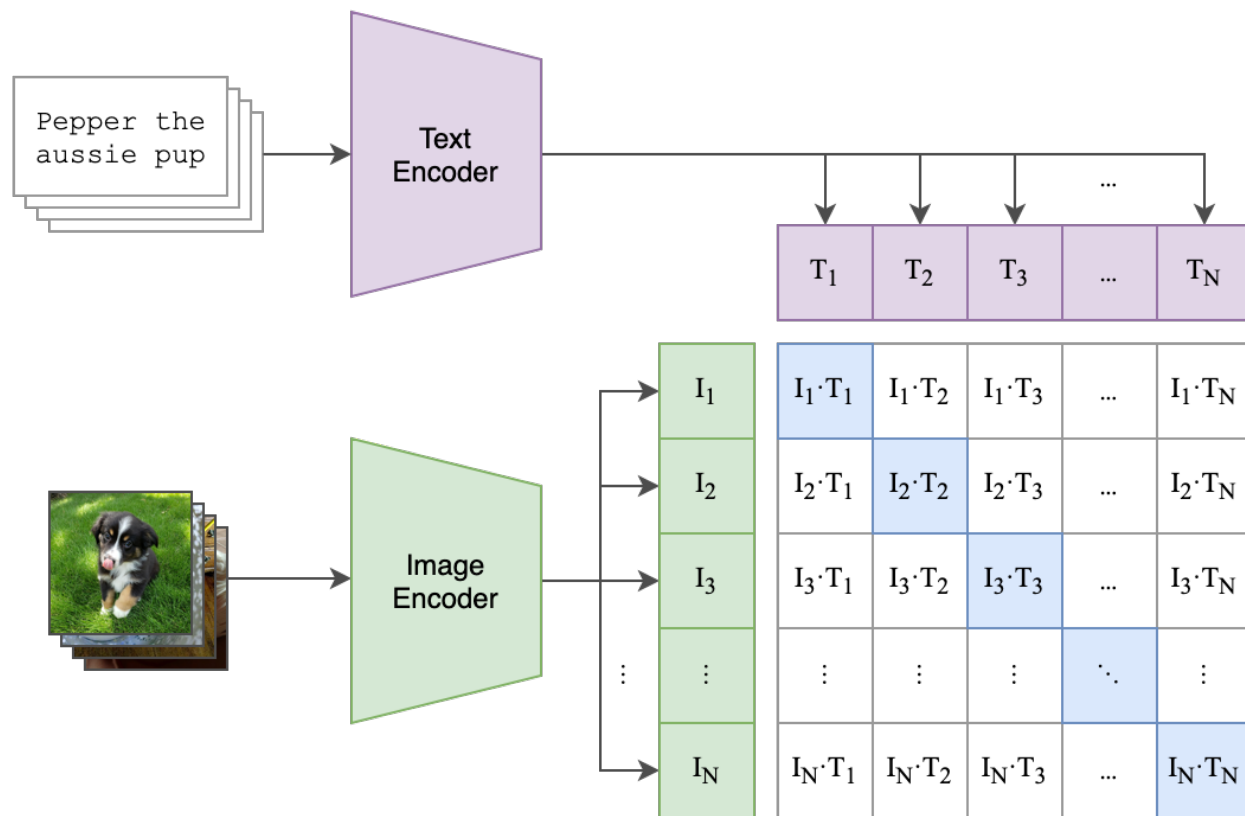
State-of-the-art performance on a broad range of tasks

Representation Learning

- High-quality representations are the crucial for modern AI.
- Generative models rely on strong encoders:
 - text encoders (for prompt understanding)
 - vision encoders (for alignment/supervision)
 - multimodal embeddings (image–text matching)
- Good representations lead to:
 - better generalization
 - stronger zero-shot transfer
 - more controllable generation (without modality alignment, generative models cannot reliably follow instructions.)

CLIP

(1) Contrastive pre-training



To generate an image from text, the model must understand the text prompt. To describe an image with text, the model must understand the image.

To align modalities (image $\leftrightarrow$ text), the model must learn a **shared representation space**.

CLIP: Learn a joint embedding space for images and text, using large-capacity models and web-scale datasets

CLIP

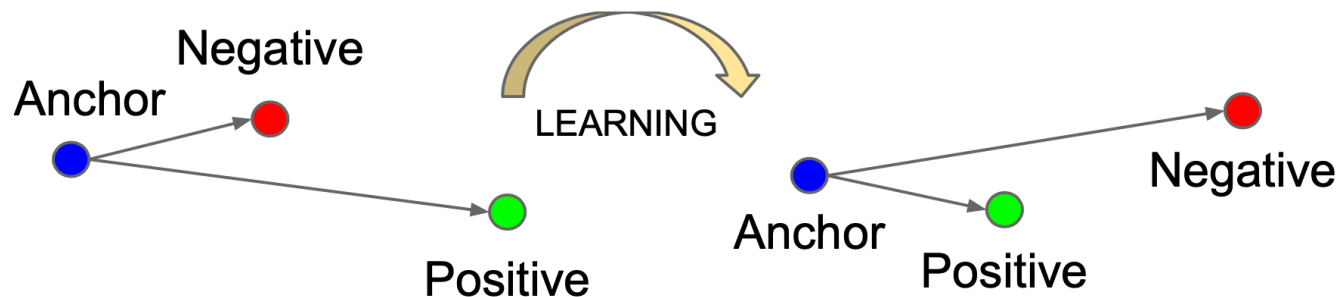
Contrastive learning

The goal is to associate each image with the correct label

Pull together matched pairs while push away unmatched pairs

Reduce pair-wise feature distance
(equivalent to increasing feature
similarity)

Increase pair-wise feature distance
(equivalent to decreasing feature
similarity)



CLIP

Backend url:

Index:

cute cat



[Clip retrieval](#) works by converting the text query to a CLIP embedding, then using that embedding to query a knn index of clip image embeddings

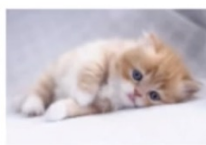
Display captions ☒
Display full captions ☐
Display similarities ☐
Safe mode ☒
Hide duplicate urls ☒
Search over

This UI may contain results with nudity and is best used by adults. The images are under their own copyright.

Are you seeing near duplicates? KNN search are good at spotting those, especially so in large datasets.



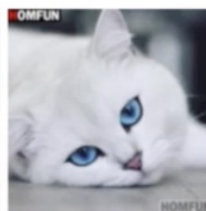
Fluffy Kitten does not know what to do.



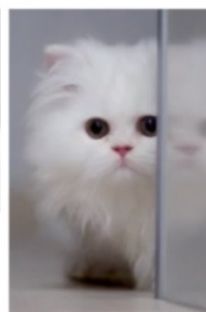
Best Cute Kitten Wallpaper No 5



cutestofthecute: (via)



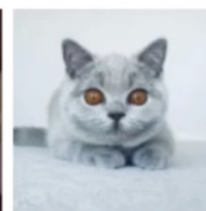
5D Diamond Painting White Cat with Blue Eyes Kit



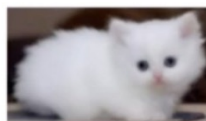
Cute White Cat Hd



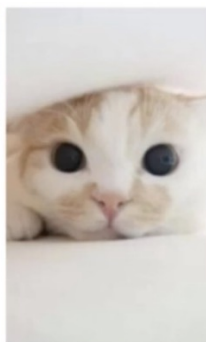
...cute little kittie... :)



Criadero especializado en British Shorthair



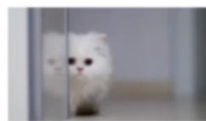
Gorgeous Himalayan Persian Kittens



Cats are one of the few



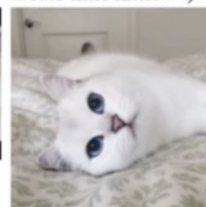
Fluffy Orange Kitten With Blue Eyes | Too Cute!



Cute cat wallpaper



This Munchkin Kitten Will Melt Your Heart With Cut...



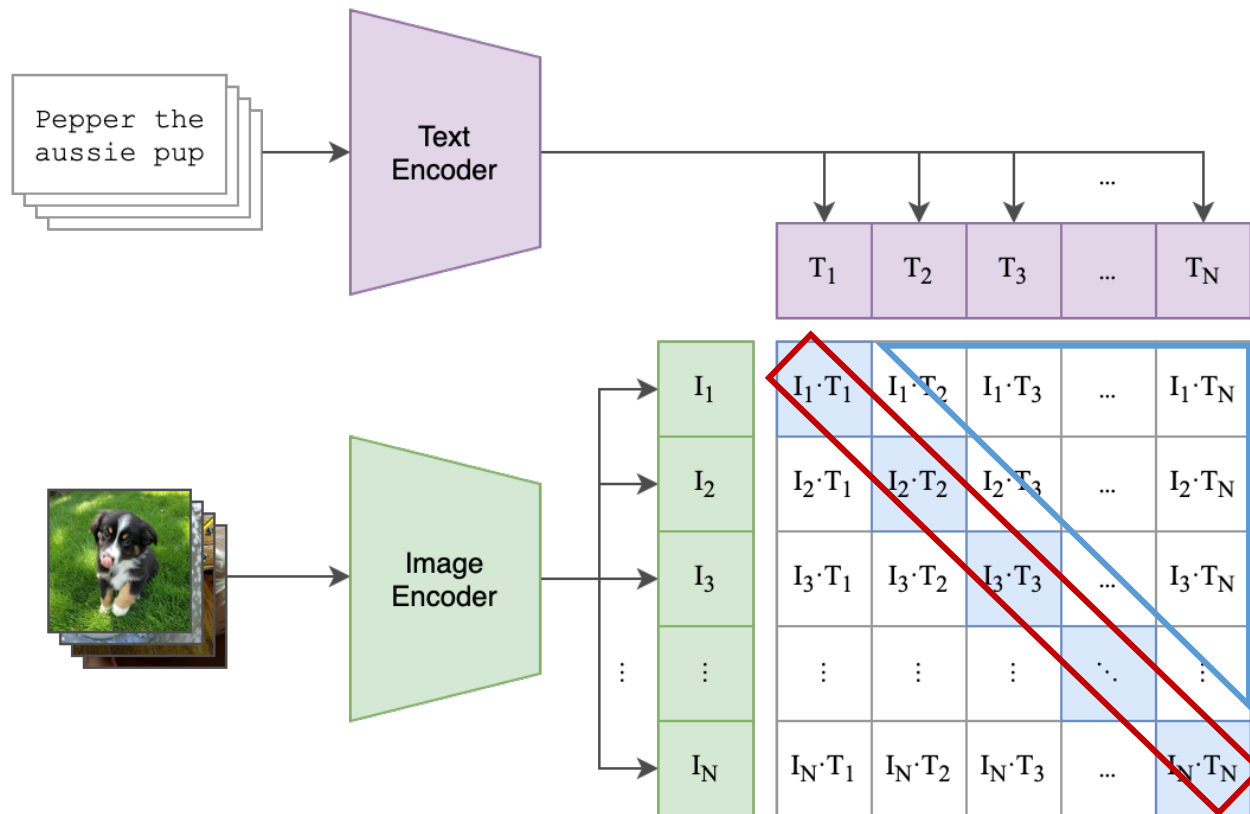
This Cat Has the Most Beautiful Eyes - We Love Cat...



Snoopy, Exotic Shorthair.

CLIP

(1) Contrastive pre-training



During training, the models are presented with batches of N image-caption pairs.

Multi-class N-pair loss

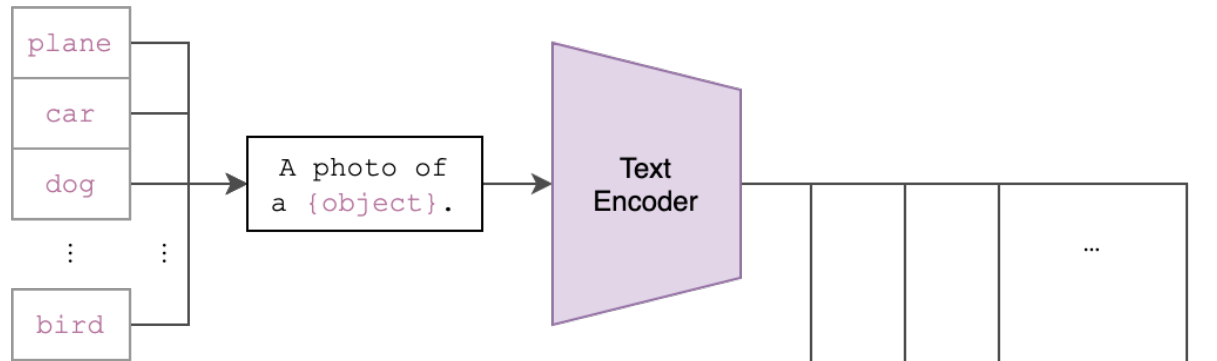
Encourages the dot product between matching image and text vectors to be high, while discouraging high dot products between non-matching pairs.

Decrease feature similarity

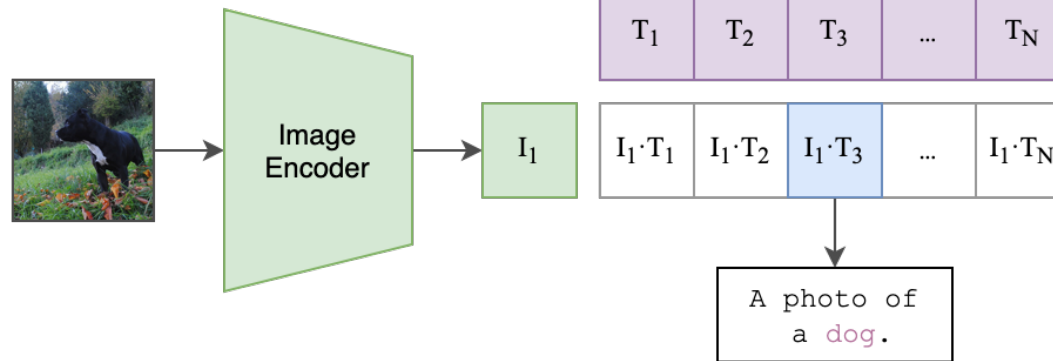
Increase feature similarity

CLIP

(2) Create dataset classifier from label text



(3) Use for zero-shot prediction



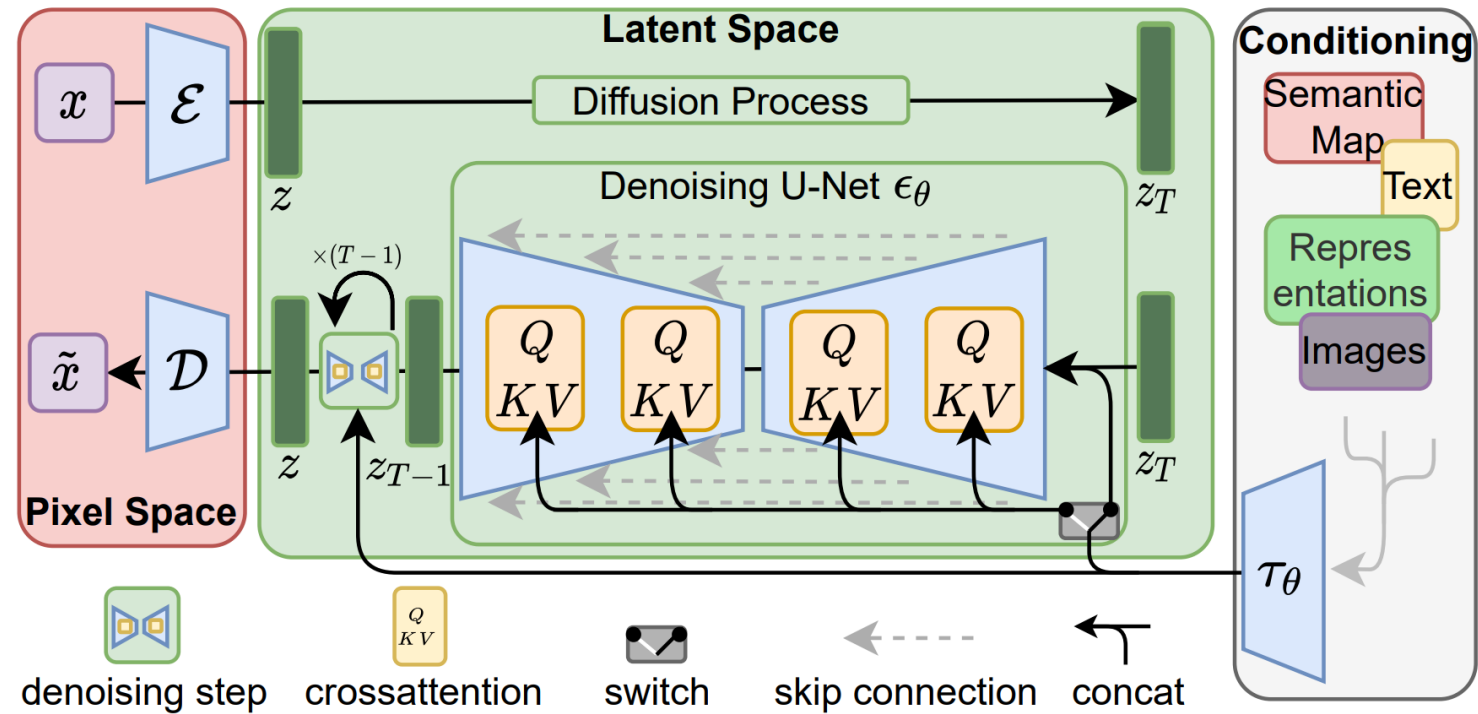
Zero-shot classification

Prompting the text encoder with class names and selecting the class whose embedding is closest to the image embedding.

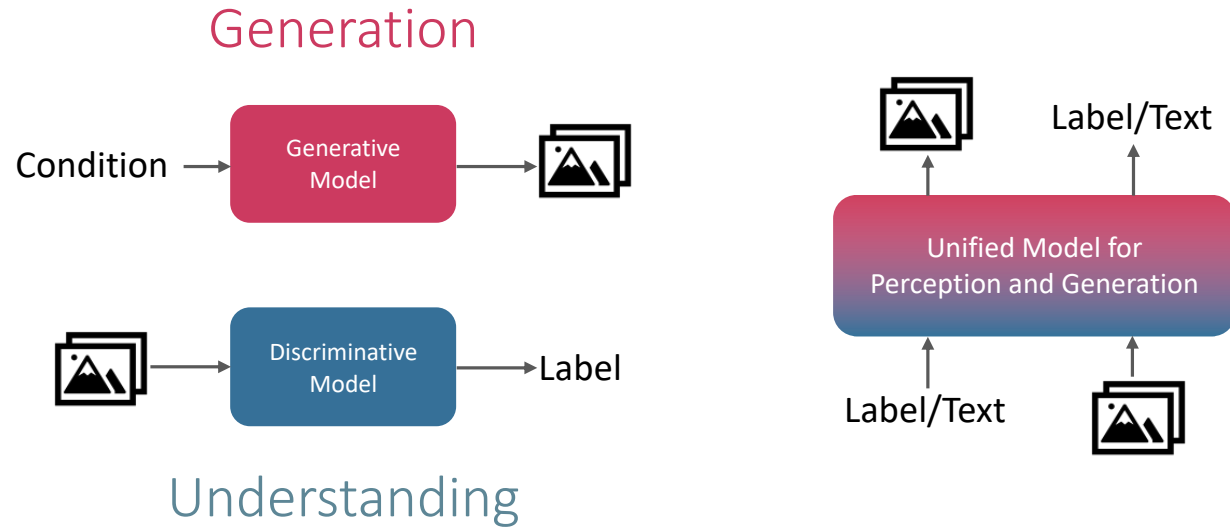
For example, to classify an image, compare the embedding of the image with the embedding of the text "A photo of a {class}.", and the {class} that results in the highest dot product is outputted.

CLIP

CLIP is not generative by itself; it provides alignment signals that make generation *controlled*.



Unified Multimodal Models



What I cannot create, I do not understand - Learn internal representations of how the world works. Predict, reason and plan actions.

Why unifying perception and generation?

- **Overall performance improvement**

- End-to-end processing of multiple modalities and tasks simultaneously
- Performance closer to human perception, cognition, and behavior

- **Scaling**

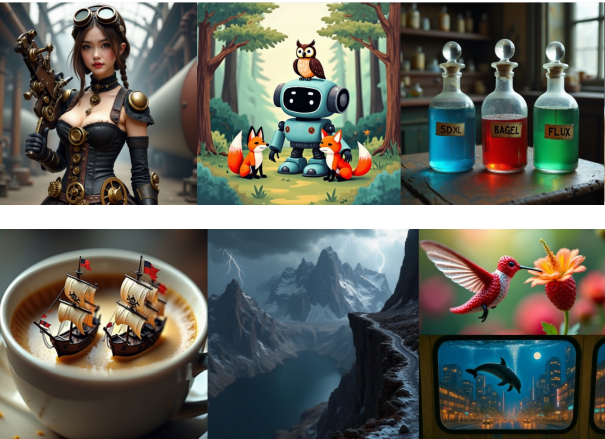
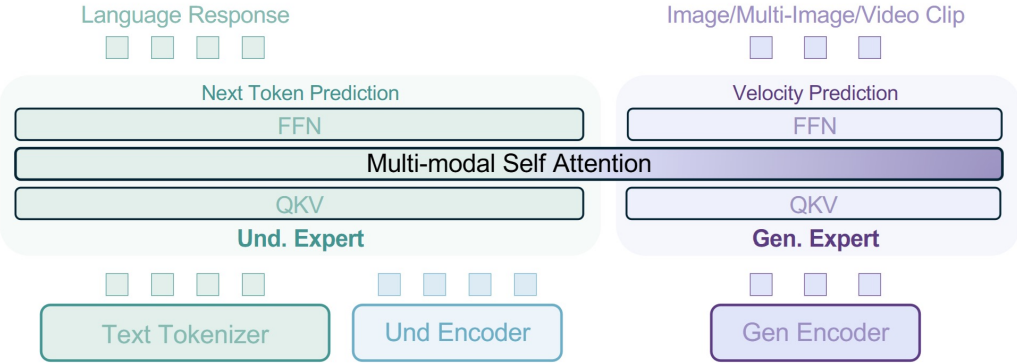
- Construct larger and more diverse datasets for training
- Can safely scale the parameters with overfitting

- **Information sharing and integration**

- Modalities are mapped into a unified state space
- No information is lost when transferring between different modalities
- Different modalities can cross-validate each other

Unified Multimodal Models

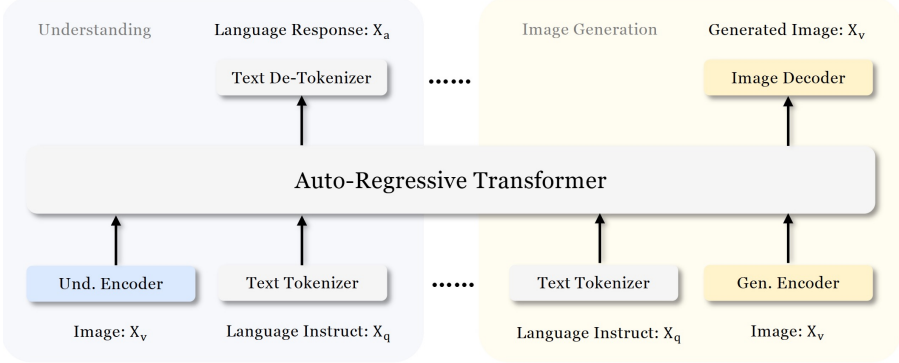
BAGEL (ByteDance, 2025)



A car made of small cars.

<think> The model should generate an image where numerous small cars are combined to form the shape and structure of a larger car. The refined prompt is: A highly detailed and creative image showcasing a large car composed entirely of numerous small cars, meticulously arranged to replicate the shape and structure of a full-sized vehicle. The small cars are tightly interlocked, creating a seamless and realistic exterior with accurate proportions and details. The setting is neutral and clean, with soft, diffused lighting to emphasize the intricacy of the composition without any distractions... </think>

Janus-Pro (DeepSeek, 2025)



Describe this scene in detail.

This image depicts a **stunning coastal landscape** featuring a series of **tall, rugged rock** formations emerging from the ocean. The rocks are layered with sedimentary rock, giving them a distinct stratified appearance. The largest formation is prominently positioned in the foreground, with smaller ones receding into the distance. The ocean waves crash against the base of these formations, creating **white foam** and a **dynamic sense of movement**.

The **coastline is sandy**, with a narrow strip of beach visible on the right side of the image. The beach is bordered by steep cliffs, which rise dramatically from the shoreline. The cliffs are covered in patches of green vegetation, adding a touch of color to the otherwise earthy tones of the rocks.

The sky above is **overcast**, with thick clouds that suggest an impending storm or recent rainfall. The light is diffused, casting a soft, muted glow over the entire scene. Despite the cloudy sky, there is a sense of tranquility and natural beauty in this coastal landscape.



A golden retriever lying peacefully on a wooden porch, with autumn leaves scattered around.



A young woman with freckles wearing a straw hat, standing in a golden wheat field.



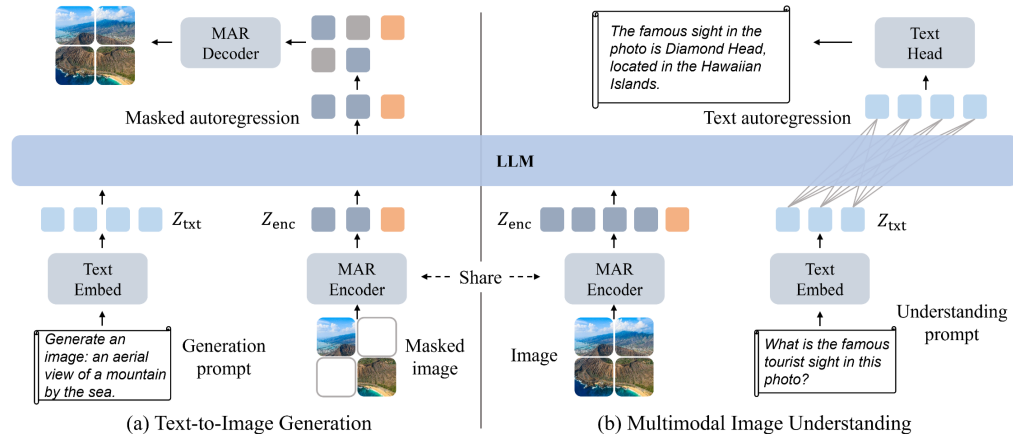
A single drop of water clinging to a green leaf, with sunlight creating a faint rainbow prism



An ancient stone bridge arching over a crystal-clear mountain stream, surrounded by lush greenery.

Unified Multimodal Models

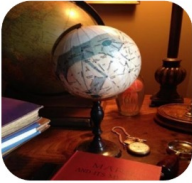
Harmon (S-Lab, NTU, 2025)



User: How many globes are there in the image?

A. 5 B. 3
C. 4 D. 2

Model: D

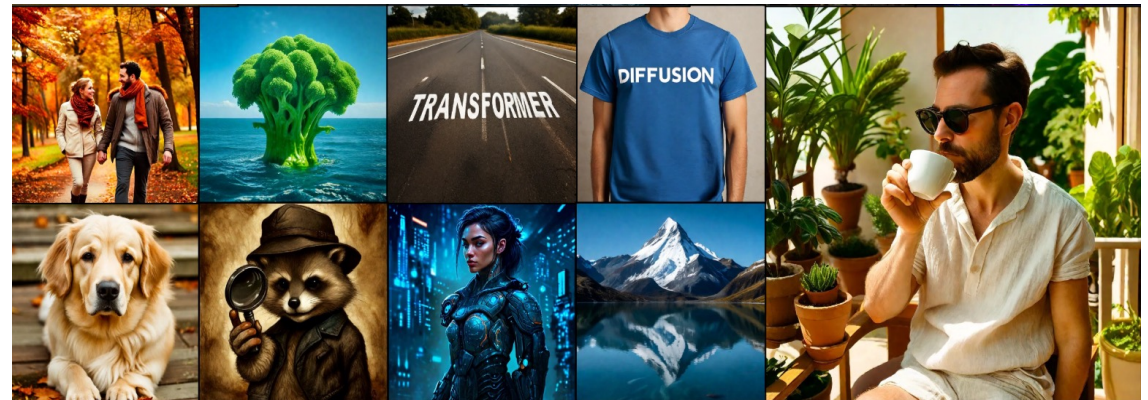
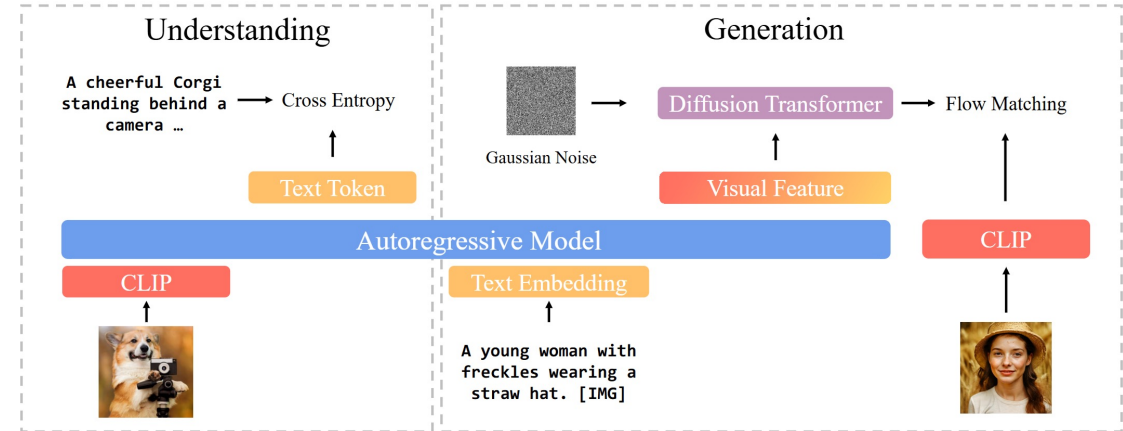


User: Is there an orange bus in the image? Please answer yes or no.

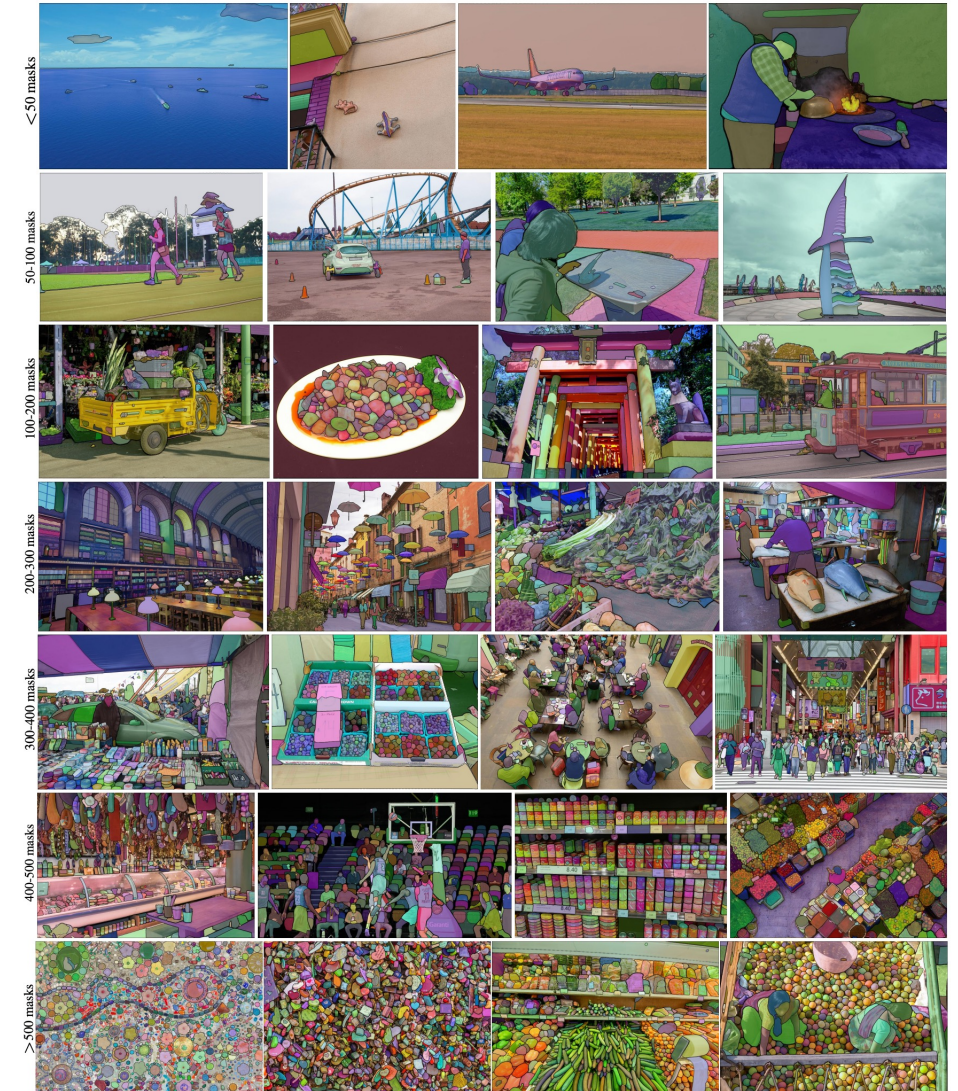
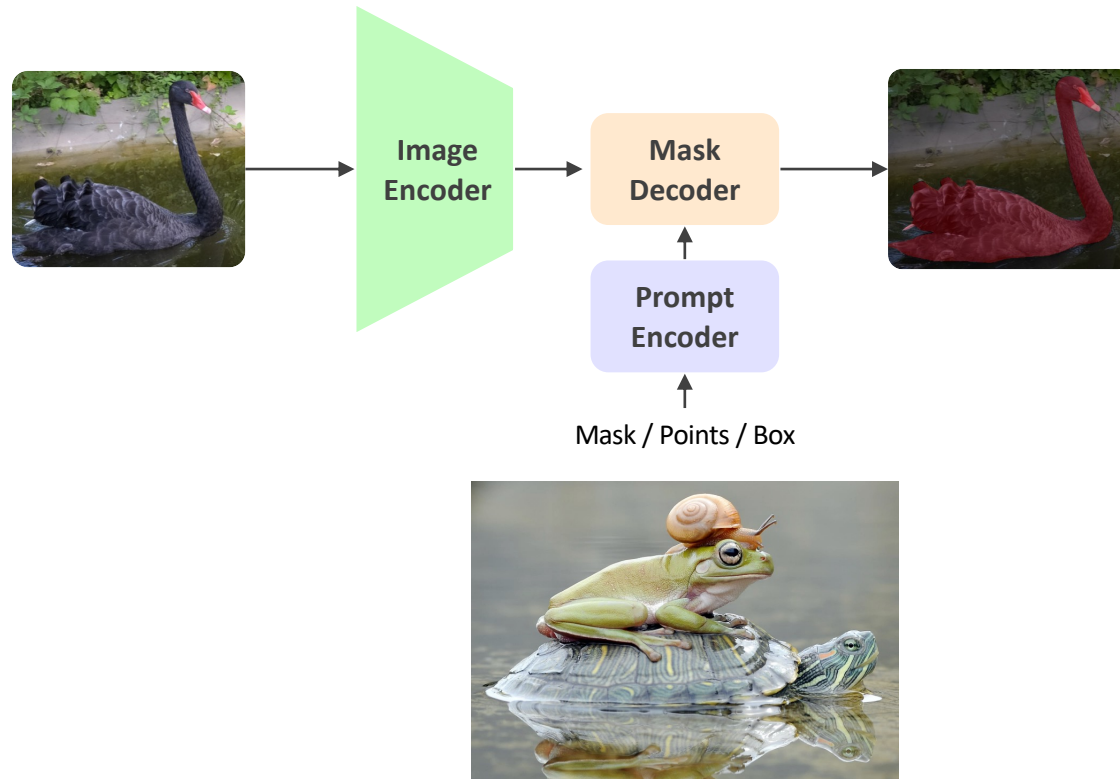
Model: No



BLIP3-o (Salesforce, 2025)

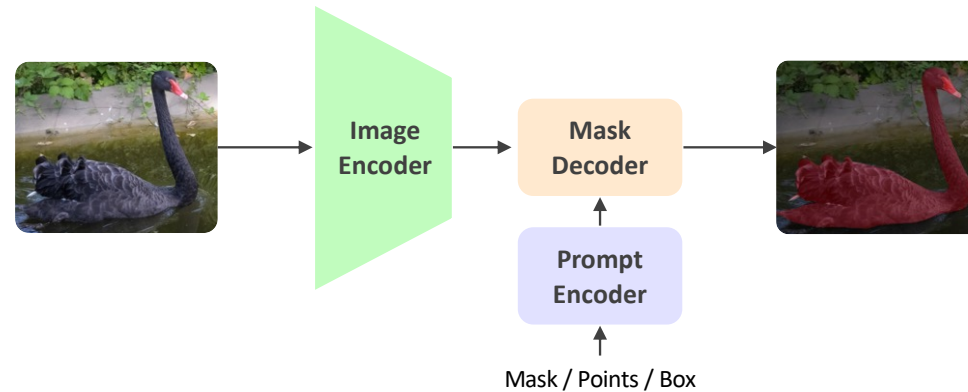


Segment Anything Model (SAM)



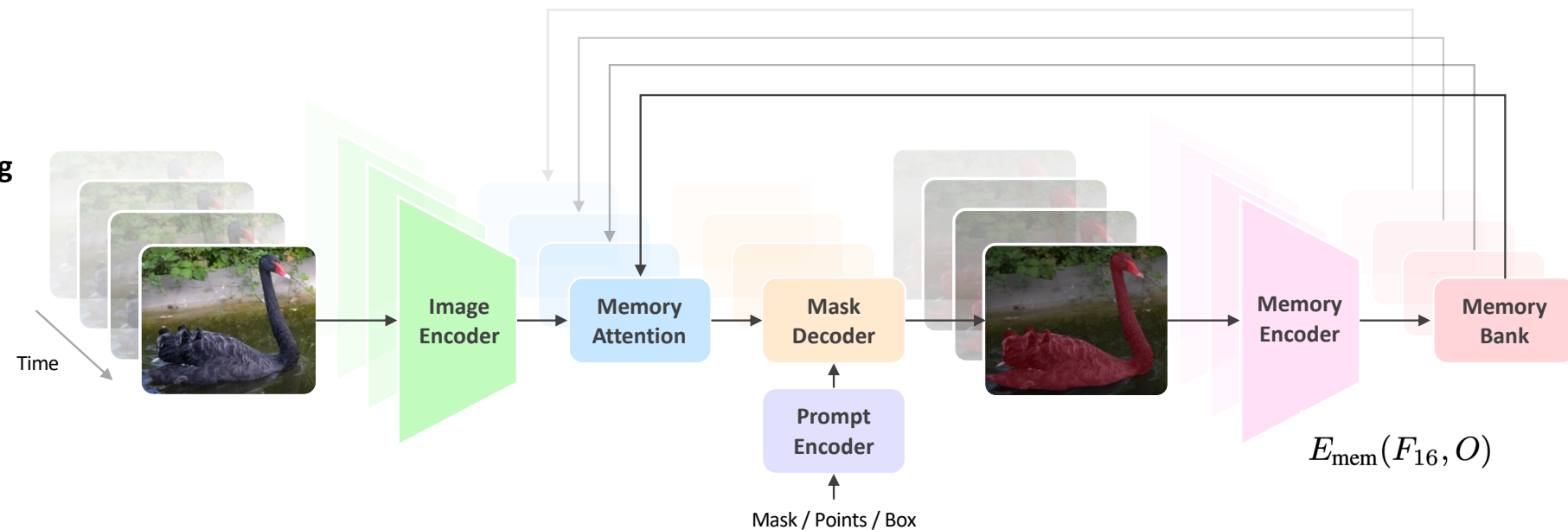
Segment Anything Model 2 (SAM2)

Alexander Kirillov et al. **Segment Anything**.
In ICCV, 2023



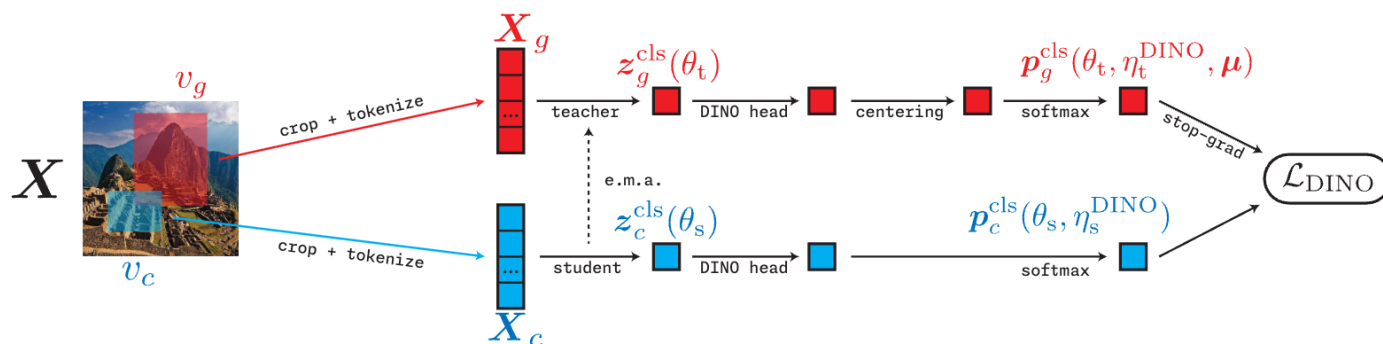
Nikhila Ravi et al. **SAM 2: Segment Anything
in Images and Videos**. In ICLR, 2025

Trained on (SA-V) 50.9K videos with 642.6K
masklets



DINO – Self-Distillation with No Labels

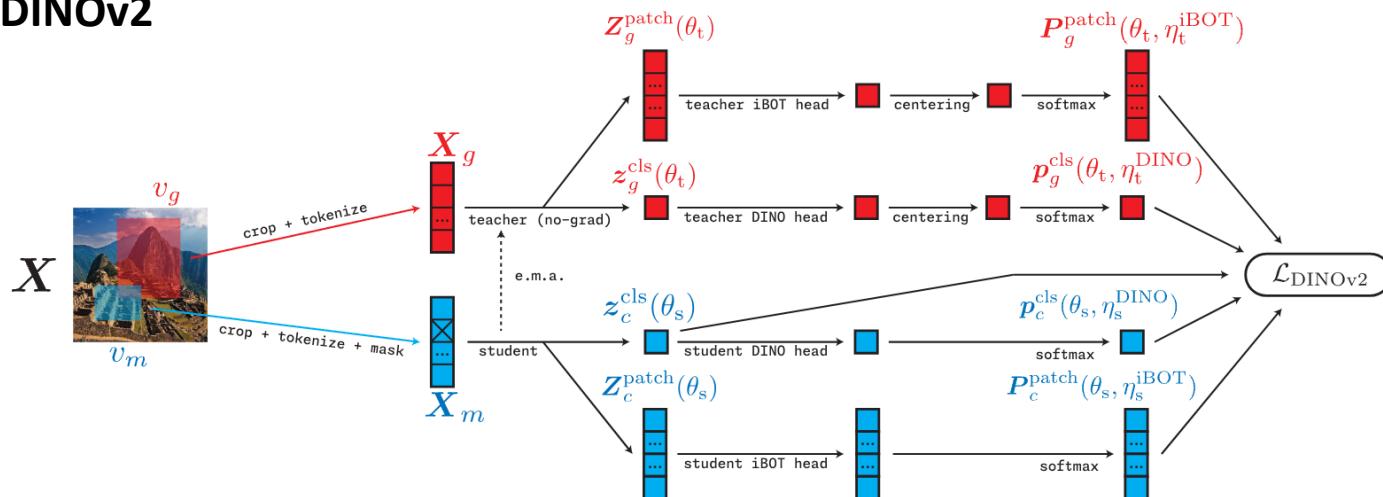
DINOv1



In the DINO pipeline, an input image is turned into patches.

Then a global view and a local view are randomly sampled.

DINOv2



The global view is pushed through the teacher encoder, while the other view is through the student encoder.

Different type of losses:

- Image-level (see DINOv1)
- Patch-level (see DINOv2 and [iBOT](#))

DINOv3: scale pretraining in terms of data and model size

[DINO](#): Emerging Properties in Self-Supervised Vision Transformers, ICCV 2021

[DINOv2](#): Learning Robust Visual Features without Supervision

[DINOv3](#)

Image credit: <https://robinwu218.github.io/SimDINO/>

DINO – Self-Distillation with No Labels

TASK	BENCHMARK	DINO ViT-B/8 0.09B	DINOv2 ViT-G/14 1.1B	DINOv3 ViT-7B/16 7B	SIGLIP 2 ViT-G-OPT/16 1.8B	PE ViT-G/14 1.9B
Segmentation	ADE-20k	31.8	49.5	55.9	42.7	38.9
Depth estimation	NYU ↓	0.537	0.372	0.309	0.494	0.436
Video tracking	DAVIS	68.7	76.6	83.3	62.9	49.8
Instance retrieval	Met	17.1	44.6	55.4	13.9	10.6
Image classification	ImageNet ReaL	85.9	89.9	90.4	90.5	90.4
Image classification	ObjectNet	39.9	66.4	79.0	78.6	80.2
Fine-grained Image classification	iNaturalist 2021	68.3	86.1	89.8	82.7	87.0

Q&A