Machine Learning Department
School of Computer Science
Carnegie Mellon University

Vision-Language Models (VLMs)

Matt Gormley & Aran Nayebi
Lecture 13
October 8, 2025

Reminders

- **Homework 3: Applying and Adapting LLMs**
  - **Out: Sat, Oct 4**
  - **Due: Thu, Oct 23 at 11:59pm**

(Preview)

LATENT DIFFUSION MODEL (LDM)

Latent Diffusion Model

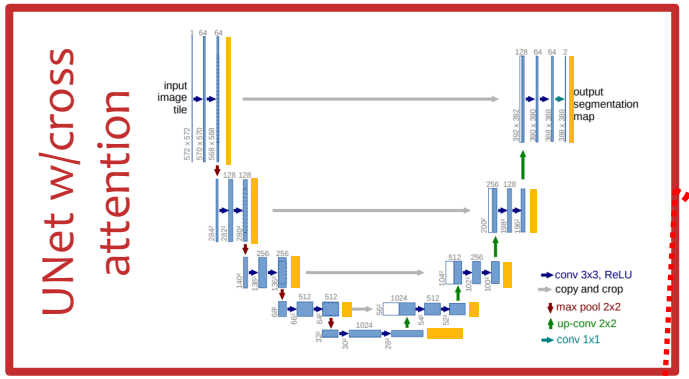
Motivation:

- diffusion models typically operate in pixel space
- yet, training typically takes hundreds of GPU days
  - 150 – 1000 V100 days [Guided Diffusion] (Dhariwal & Nichol, 2021)
  - 256 TPU-v4s for 4 days = 1000 TPU days [Imagen] (Sharia et al., 2022)
- inference is also slow
  - 50k samples in 5 days on A100 GPU [Guided Diffusion] (Dhariwal & Nichol, 2021)
  - 15 seconds per image

Key Idea:

- train an autoencoder (i.e. encoder-decoder model) that learns an efficient latent space that is perceptually equivalent to the data space
- keeping the autoencoder fixed, train a diffusion model on the latent representations of real images z_0 = encoder(x)
  - forward model: latent representation z_0 → noise z_T
  - reverse model: noise z_T → latent representation z_0
- to generate an image:
  - sample noise z_T
  - apply reverse diffusion model to obtain a latent representation z_0
  - decode the latent representation to an image x
- condition on prompt via cross attention in latent space

Latent Diffusion Model (LDM)



prompt space

latent space

pixel space

$\widehat{\mathbf{y}}$

τ_θ

LLM

$\mathbf{y}$

orange cat

$p_\theta(\mathbf{z}_T)$

$p_\theta(\mathbf{z}_{T-1} \mid \mathbf{z}_T, \widehat{\mathbf{y}})$

$p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t+1}, \widehat{\mathbf{y}})$

$p_\theta(\mathbf{z}_0 \mid \mathbf{z}_1, \widehat{\mathbf{y}})$

z_T z_{T-1} ... z_{t+1} z_t ... z_1 z

$\mathcal{D}$

Decoder

$\widetilde{\mathbf{x}}$

$\mathcal{E}$

Encoder

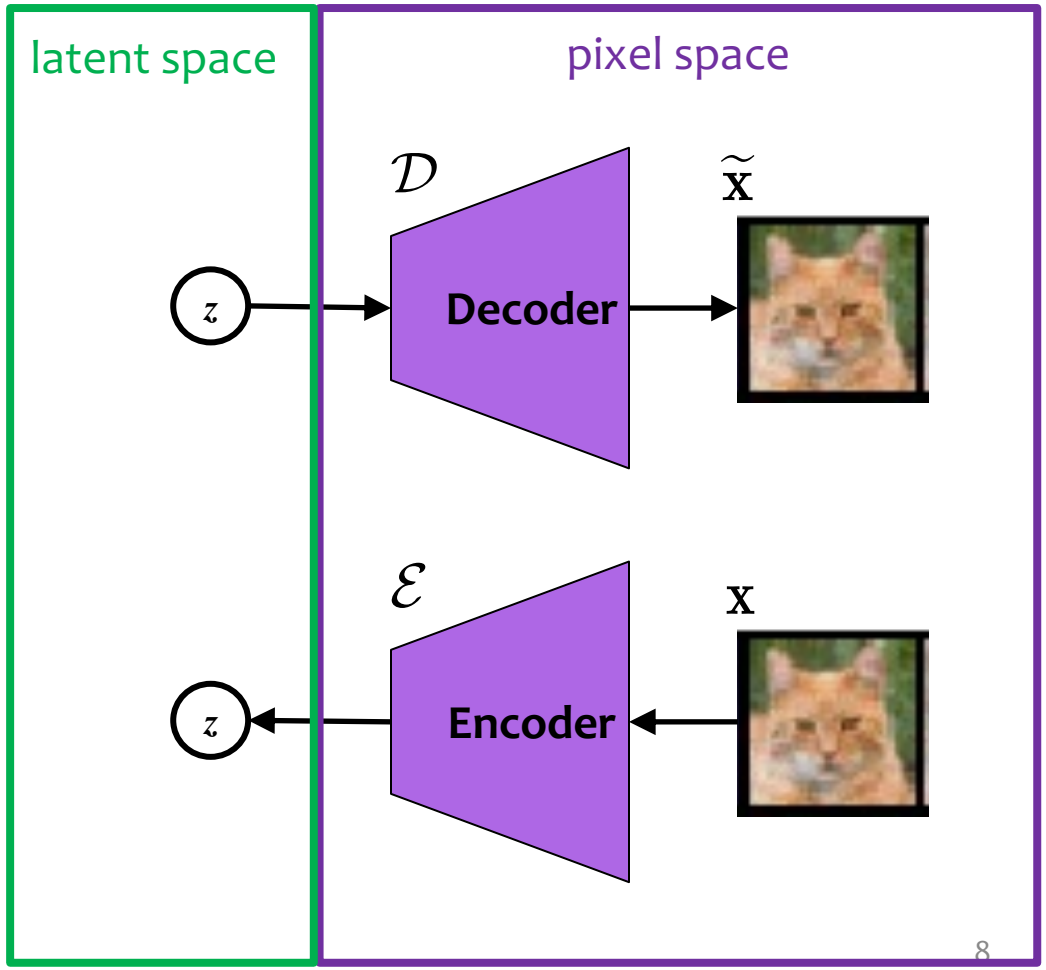
$\mathbf{x}$

$q_\phi(\mathbf{z}_T \mid \mathbf{z}_{T-1})$

$q_\phi(\mathbf{z}_{t+1} \mid \mathbf{z}_t)$

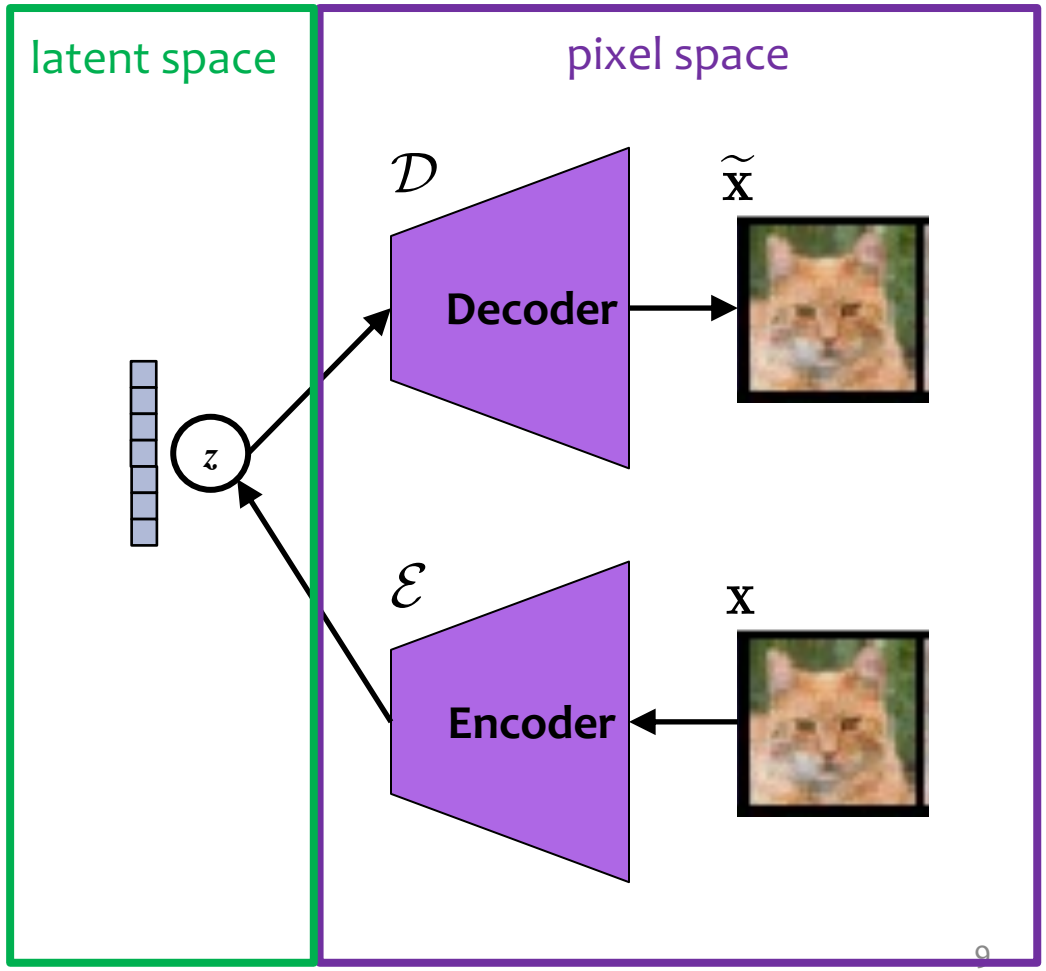
$q_\phi(\mathbf{z}_1 \mid \mathbf{z}_0)$

LDM: Autoencoder

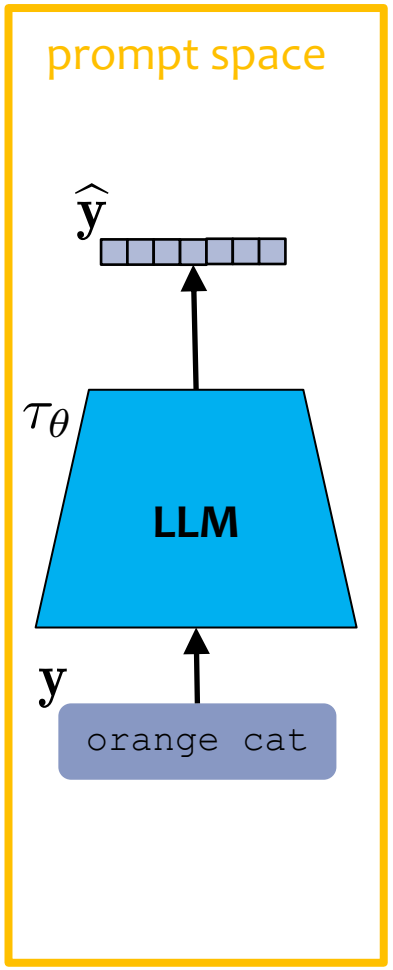


LDM: Autoencoder

- The autoencoder is chosen so that it can project high dimensional images (e.g. 1024x1024) down to low dimensional **latent space** and **faithfully** project back up to **pixel space**
- The original LDM paper considers two options:
  1. a VAE-like model (regularizes the noise towards a Gaussian)
  2. a VQGAN (performs vector quantization in the decoder; i.e., it uses a discrete codebook)
- This model is trained ahead of time just on raw images (no text prompts) and then frozen
- The frozen encoder-decoder can be reused for all subsequent LDM training



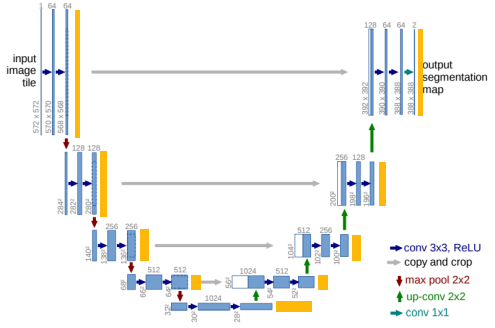
LDM: the Prompt Model



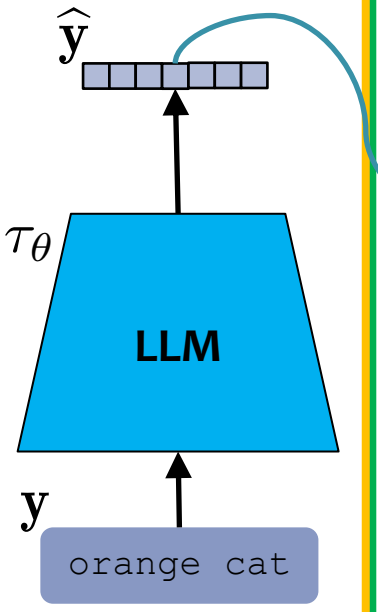
- The prompt model is just a Transformer LM
- We learn its parameters alongside the diffusion model
- The goal is to build up good representations of the text prompts such that they inform the latent diffusion process

LDM: with DDPM

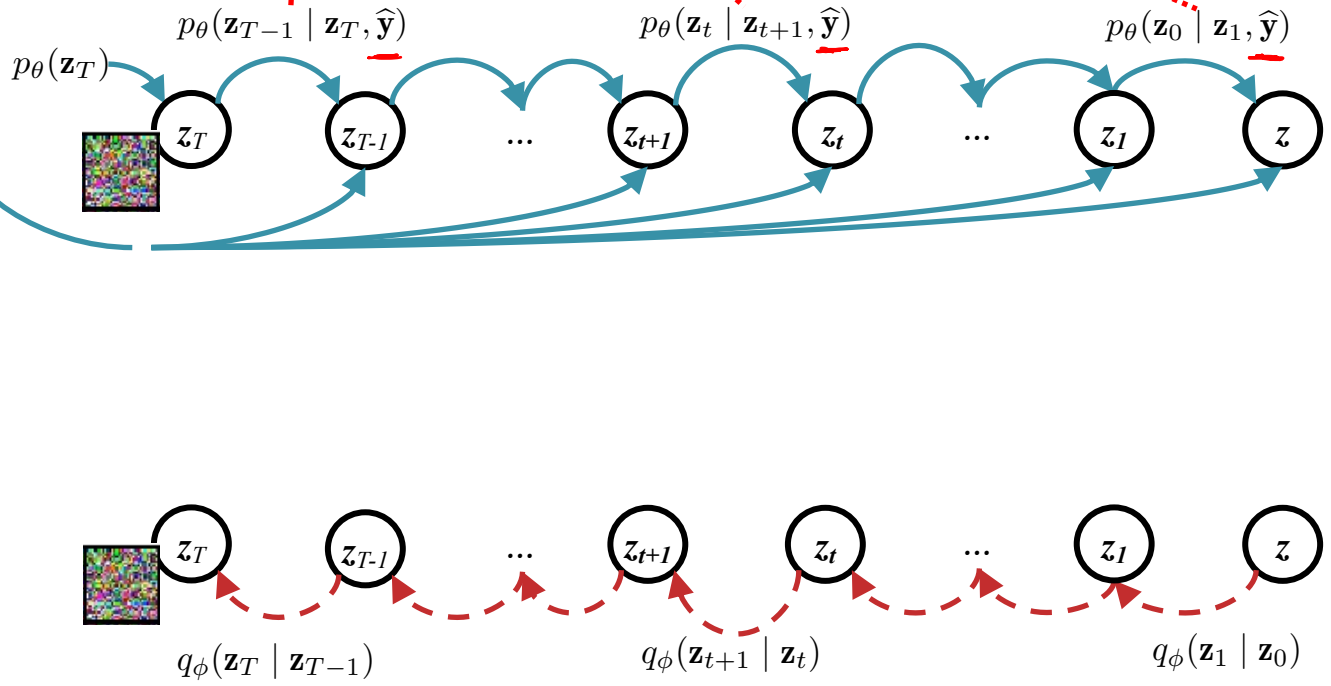
UNet w/cross attention



prompt space

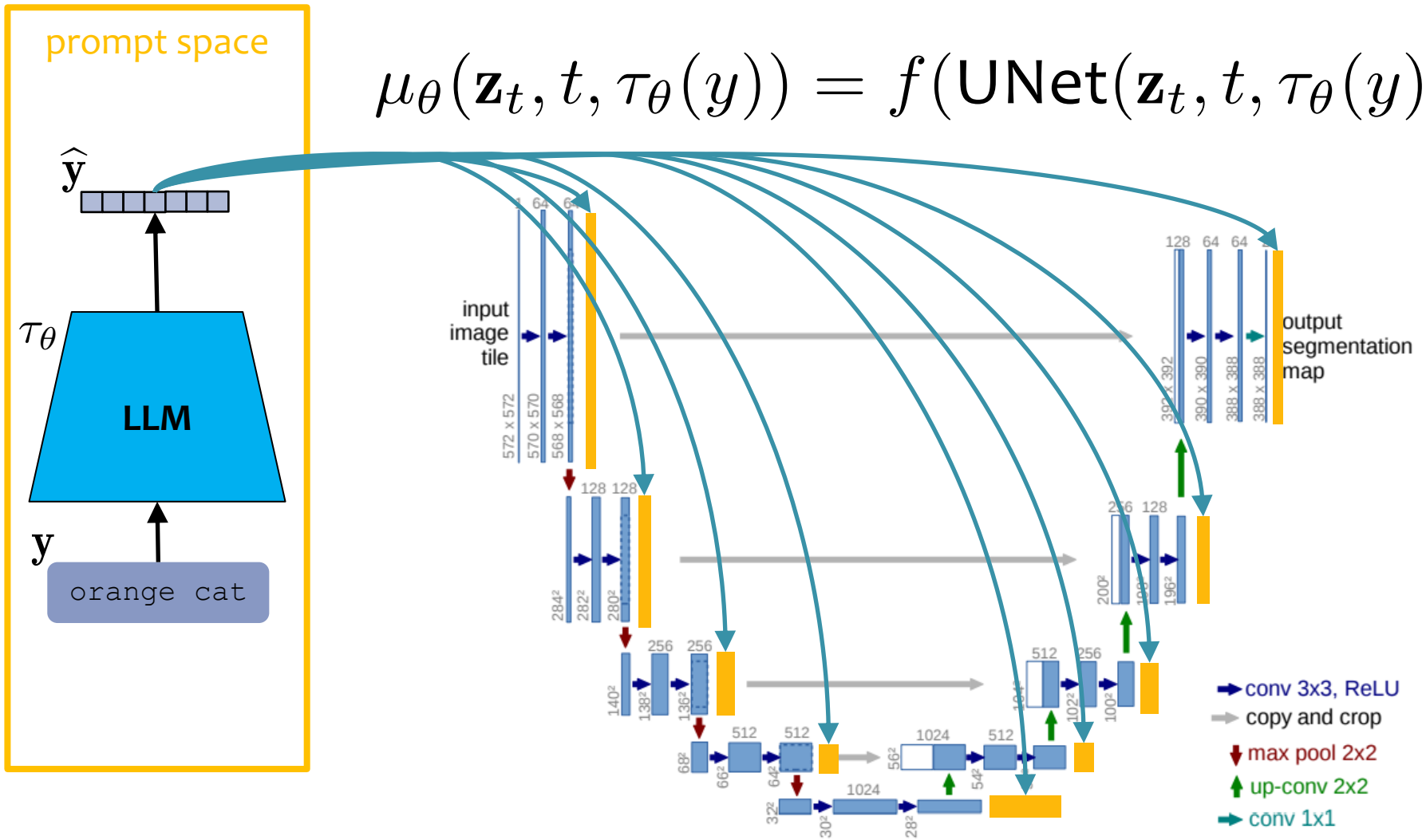


latent space



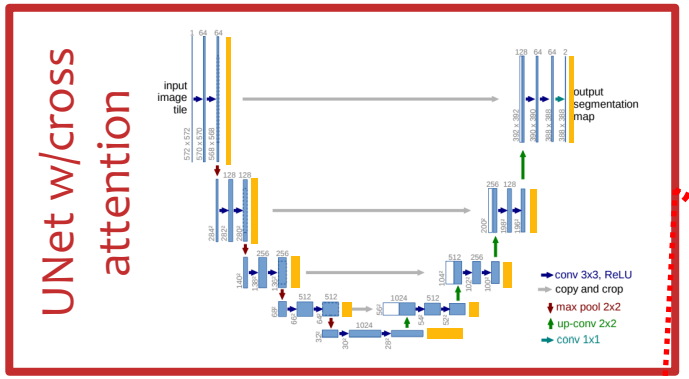
LDM: Noise Model

$$\mu_\theta(\mathbf{z}_t, t, \tau_\theta(y)) = f(\text{UNet}(\mathbf{z}_t, t, \tau_\theta(y)))$$



- The noise model includes **cross attention** (yellow boxes) to the representation of the prompt text
- During training we optimize both the parameters of the UNet noise model and the parameters of the LLM simultaneously

Latent Diffusion Model (LDM)



prompt space

latent space

pixel space

$\widehat{\mathbf{y}}$

τ_θ

LLM

$\mathbf{y}$

orange cat

$p_\theta(\mathbf{z}_T)$

$p_\theta(\mathbf{z}_{T-1} \mid \mathbf{z}_T, \widehat{\mathbf{y}})$

$p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t+1}, \widehat{\mathbf{y}})$

$p_\theta(\mathbf{z}_0 \mid \mathbf{z}_1, \widehat{\mathbf{y}})$

z_T z_{T-1} ... z_{t+1} z_t ... z_1 z

$\mathcal{D}$

Decoder

$\widetilde{\mathbf{x}}$

$\mathcal{E}$

Encoder

$\mathbf{x}$

$q_\phi(\mathbf{z}_T \mid \mathbf{z}_{T-1})$

$q_\phi(\mathbf{z}_{t+1} \mid \mathbf{z}_t)$

$q_\phi(\mathbf{z}_1 \mid \mathbf{z}_0)$

LDM: Learning the Diffusion Model + LLM

Given a training sample $\mathbf{z}_0$, we want

$$p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \tau_\theta(y))$$

to be as close as possible to

$$q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{z}_0)$$

Intuitively, this makes sense: if the *learned* reverse process is supposed to subtract away the noise, then whenever we're working with a specific $\mathbf{z}_0$ it should subtract it away exactly as *exact* reverse process would have.

Objective Function:

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\right]$$

Algorithm 1 Training

initialize θ
for $e \in \{1, \ldots, E\}$ **do**
  for $x_0, y \in \mathcal{D}$ **do**
    $t \sim \text{Uniform}(1, \ldots, T)$
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_t \leftarrow \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
    $\ell_t(\theta) \leftarrow \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \tau_\theta(\mathbf{y}))\|^2$
    $\theta \leftarrow \theta - \nabla_\theta \ell_t(\theta)$

VISION LANGUAGE MODELS (VLMS)

Multimodal Models

- Previously: Text-to-image models – adapt generative models for vision in order to guide their output toward some desired target using natural language
  - Output is still an image

- Today: visual language models (VLMs) – adapt generative models for text in order to allow them to interact with images (as well as text) as input
  - Output is (typically) still text

possibly images too

Slide from Henry Chai

VLM: Tasks

- Common benchmarks for VLMs include
  - **Visual reasoning**: given an image (or a pair of images) determine if some natural language statement about the image(s) is true or false
  - **Visual grounding**: locate an object in some image given a natural language description
  - **Visual question answering**: given an image (or images), respond to arbitrary, potentially open-ended questions about the content.
  - **Caption generation**: create natural language descriptions of content of some image

Slide from Henry Chai

VLM: Tasks

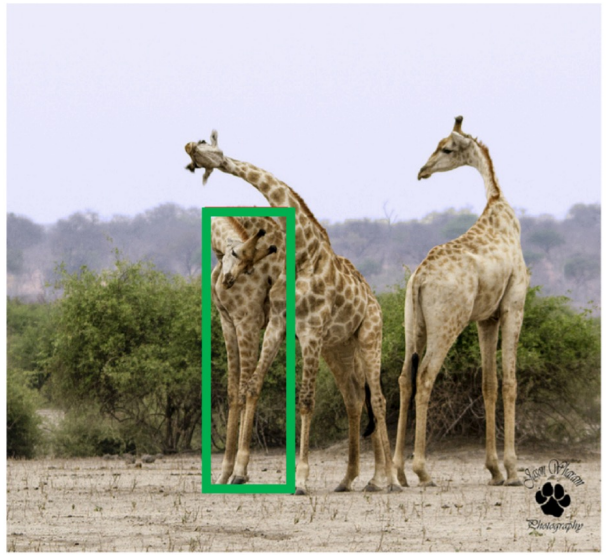
- Common benchmarks for VLMs include
  - **Visual reasoning**: given an image (or a pair of images) determine if some natural language statement about the image(s) is true or false

Slide from Henry Chai

Source: https://aclanthology.org/P19-1644.pdf

VLM: Tasks

- Common benchmarks for VLMs include
  - **Visual reasoning**: given an image (or a pair of images) determine if some natural language statement about the image(s) is true or false
  - **Visual grounding**: locate an object in some image given a natural language description



RefCOCO:
1. giraffe on left
2. first giraffe on left

RefCOCO+:
1. giraffe with lowered head
2. giraffe head down

RefCOCOg:
1. an adult giraffe scratching its back with its horn
2. giraffe hugging another giraffe

Slide from Henry Chai

Source: https://arxiv.org/pdf/1608.00272

VLM: Tasks

- Common benchmarks for VLMs include

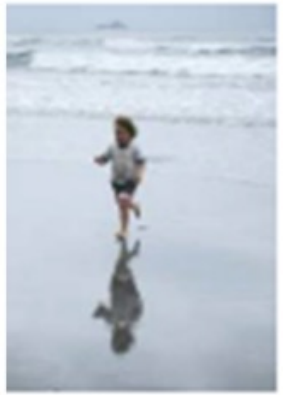
  - **Visual question answering**: given an image (or images), respond to arbitrary, potentially open-ended questions about the content.

Slide from Henry Chai

Source: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8100153

VLM: Tasks

- Common benchmarks for VLMs include



Ground Truth Caption: A little boy runs away from the approaching waves of the ocean.

Generated Caption: A young boy is running on the beach.

Ground Truth Caption: A brunette girl wearing sunglasses and a yellow shirt.

Generated Caption: A woman in a black shirt and sunglasses smiles.

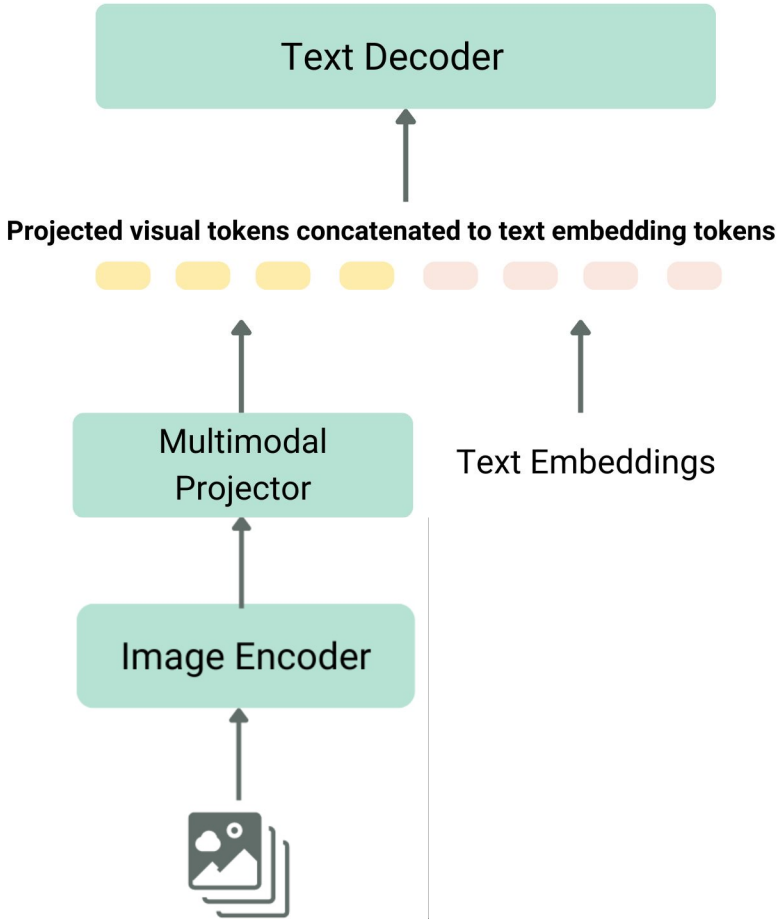
- **Caption generation**: create natural language descriptions of content of some image

Slide from Henry Chai

Source: https://dl.acm.org/doi/pdf/10.1145/3295748

VLM: Architecture

- High-level idea: convert both the image and the text inputs into embedding vectors, then pass those vectors into a decoder-only transformer and do next (text) token prediction



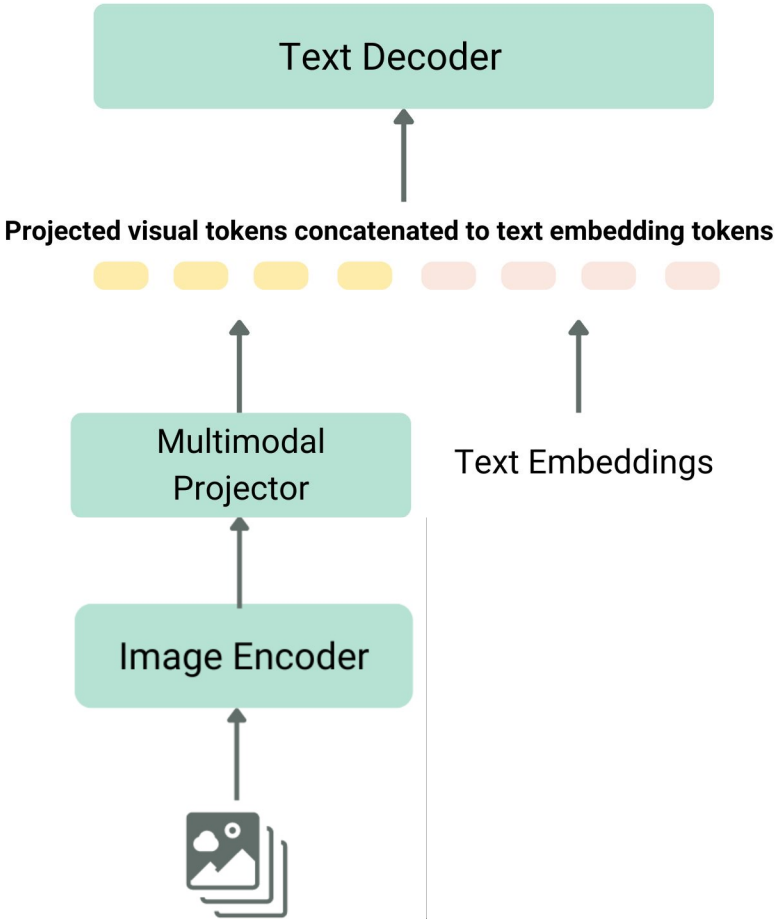
- Two common encoders:
  - VQ-VAE encoder followed by an embedding layer that converts the discrete tokens into dense numerical vectors
  - CLIP encoder, that directly learns an embedding vector using a contrastive pre-training objective

Slide from Henry Chai

Source: https://huggingface.co/blog/vlms

VLM: Architecture

- High-level idea: convert both the image and the text inputs into embedding vectors, then pass those vectors into a decoder-only transformer and do next (text) token prediction



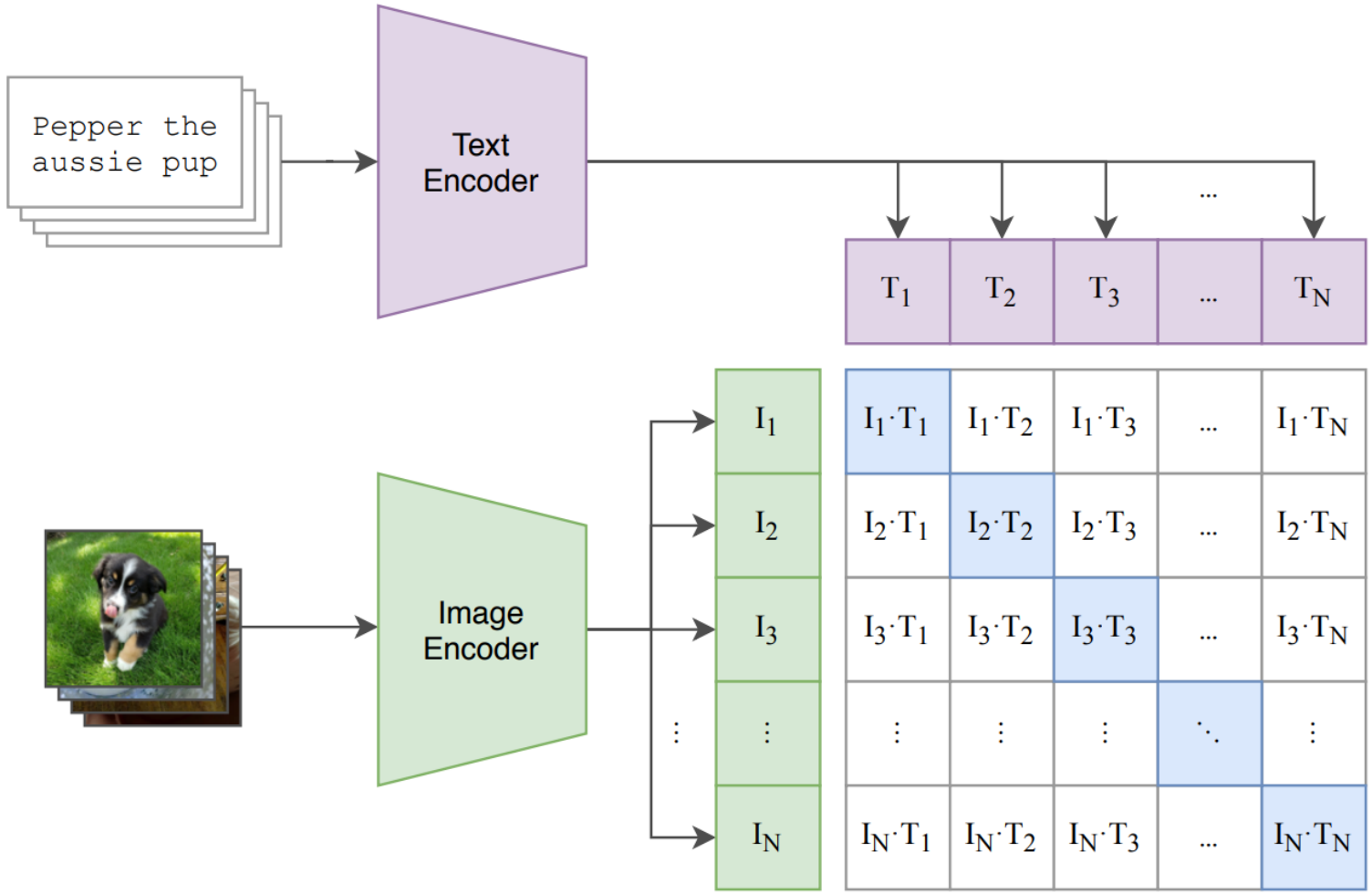
- Two common encoders:
  - VQ-VAE encoder followed by an embedding layer that converts the discrete tokens into dense numerical vectors
  - CLIP encoder, that directly learns an embedding vector using a contrastive pre-training objective

Slide from Henry Chai

Source: https://huggingface.co/blog/vlms

IMAGE ENCODERS

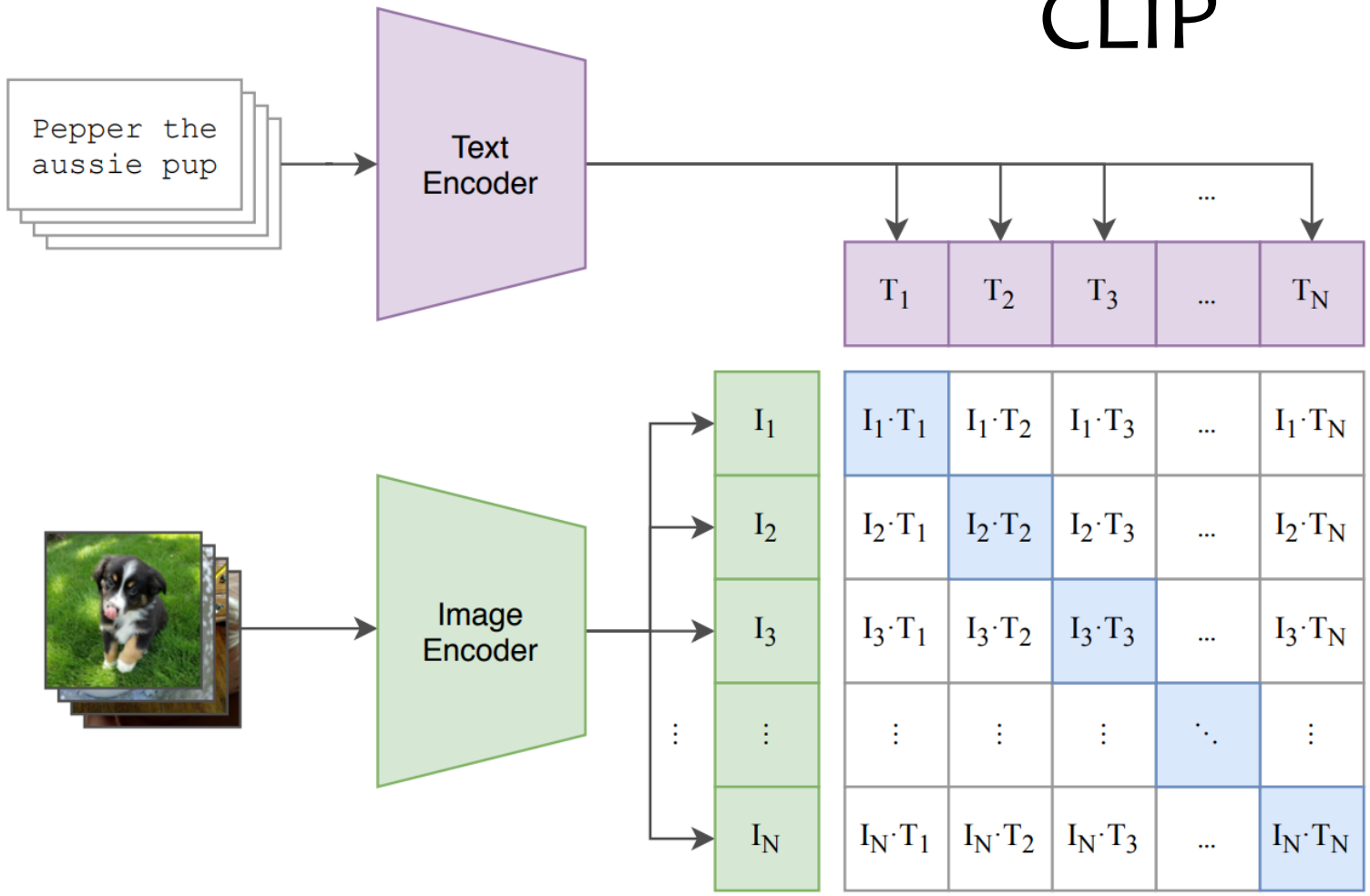
CLIP



- The text encoder is, e.g., an encoder-only transformer
- The image encoder is, e.g., a ResNet-like CNN or ViT
- Both are linearly projected into same-dimensional vectors i.e., the multi-modal embedding space
- Assume we have a mini-batch $B = \{(I_1, T_1), (I_2, T_2), \ldots, \{(I_1, T_1)\}$ of size N

Figure from Radford et al. (2021) https://arxiv.org/pdf/2103.00020

CLIP



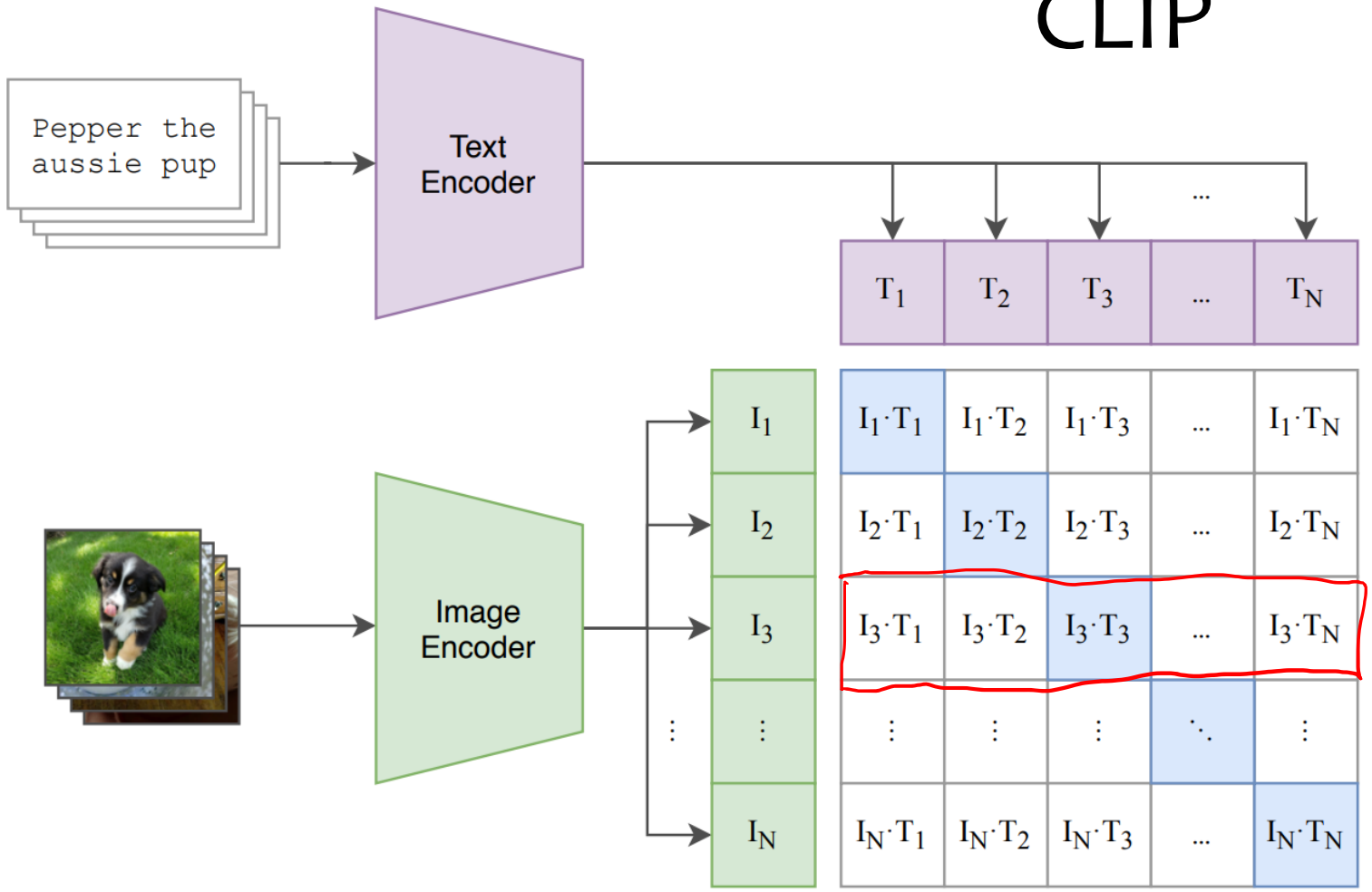
Incorrect (but intuitive) objective function:

$$\max \left[\sum_{i=1}^{N} I_i^\top T_i - \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} I_i^\top T_j \right]$$

Given a mini-batch of N (image, caption) pairs, both encoders are simultaneously pre-trained to maximize the cosine similarity of corresponding image-caption embedding vectors and minimize all other pairwise cosine similarities

Figure from Radford et al. (2021) https://arxiv.org/pdf/2103.00020

CLIP



Correct objective function:

$$\max \sum_{i=1}^{N} \left[\log \frac{\exp\left(\frac{I_i^\top T_i}{\tau}\right)}{\sum_{j=1}^{N} \exp\left(\frac{I_i^\top T_j}{\tau}\right)} + \log \frac{\exp\left(\frac{I_i^\top T_i}{\tau}\right)}{\sum_{j=1}^{N} \exp\left(\frac{I_j^\top T_i}{\tau}\right)} \right]$$

Given a mini-batch of N (image, caption) pairs, both encoders are simultaneously pre-trained to maximize the cosine similarity of corresponding image-caption embedding vectors and minimize all other pairwise cosine similarities

Figure from Radford et al. (2021) https://arxiv.org/pdf/2103.00020

CLIP

Can be interpreted in terms of two conditional probability distributions:

1. **Image-to-Text distribution:**

$$p(T_j \mid I_i) = \frac{\exp\left(\frac{I_i^\top T_j}{\tau}\right)}{\sum_{k=1}^{N} \exp\left(\frac{I_i^\top T_k}{\tau}\right)}$$

softmax of each row

I_i ↑ $I_1, \ldots, I_N$

2. **Text-to-Image distribution:**

$$p(I_j \mid T_i) = \frac{\exp\left(\frac{I_j^\top T_i}{\tau}\right)}{\sum_{k=1}^{N} \exp\left(\frac{I_k^\top T_i}{\tau}\right)}$$

softmax of each column

T_i ↑ $T_1, \ldots, T_N$

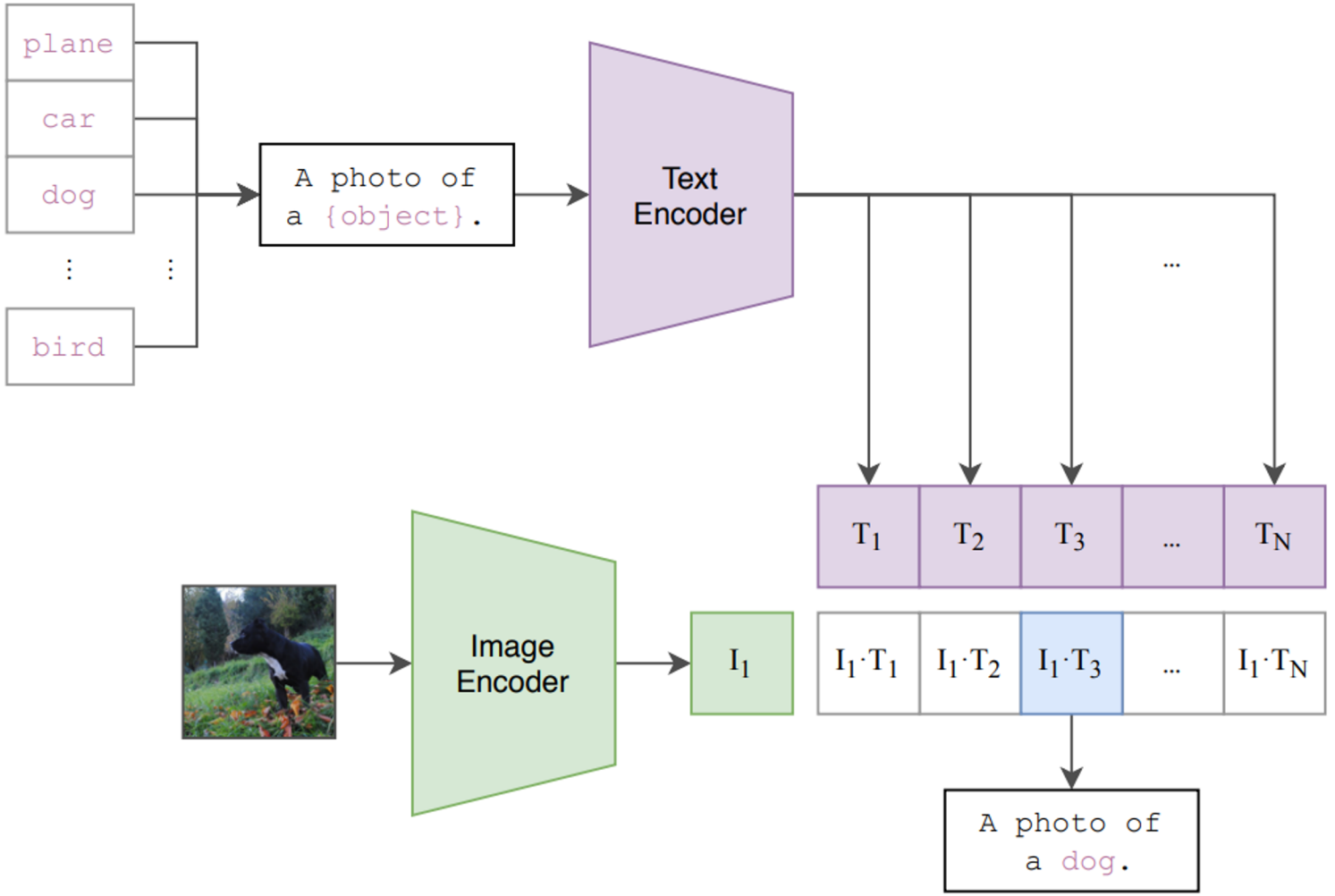
The bidirecional **contrastive loss** maximizes the log-likelihood of the correct (matched) pairs under both distributions:

$$\sum_{i=1}^{N} \log p(T_i \mid I_i) + \log p(I_i \mid T_i)$$

Correct objective function:

$$\max \sum_{i=1}^{N} \left[\log \frac{\exp\left(\frac{I_i^\top T_i}{\tau}\right)}{\sum_{j=1}^{N} \exp\left(\frac{I_i^\top T_j}{\tau}\right)} + \log \frac{\exp\left(\frac{I_i^\top T_i}{\tau}\right)}{\sum_{j=1}^{N} \exp\left(\frac{I_j^\top T_i}{\tau}\right)} \right]$$

CLIP for Zero Shot Classification



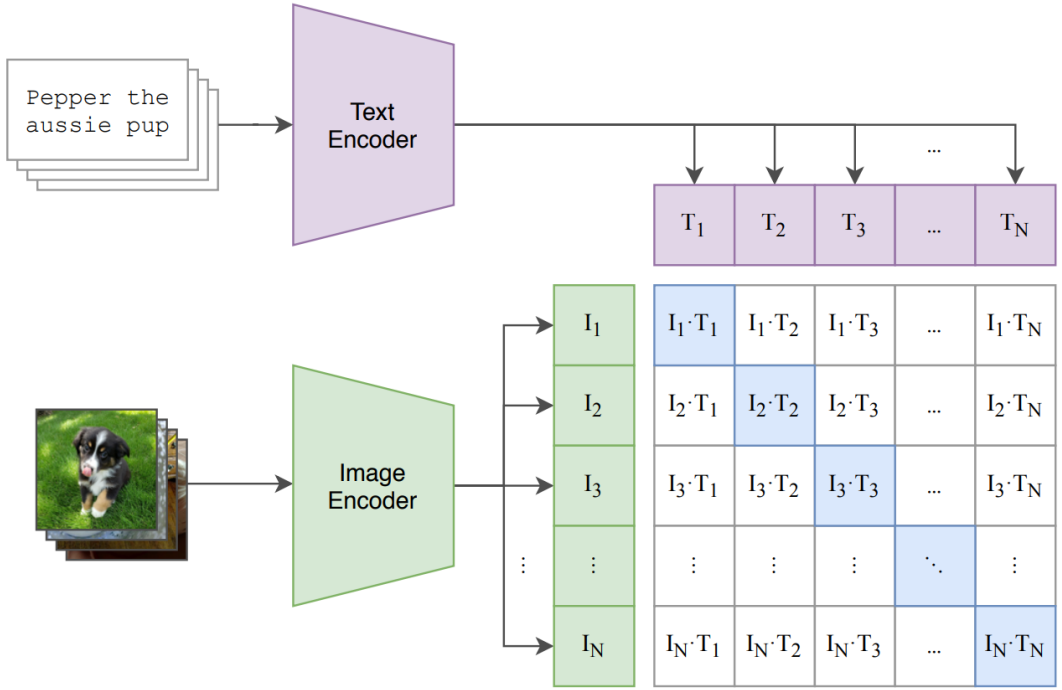
Among the N labels, choose the one that has the highest probability:

$$\hat{y} = \underset{i}{\operatorname{argmax}} P(T_i \mid I_1)$$

$$= \underset{i}{\operatorname{argmax}} \frac{\exp\left(\frac{I_1^\top T_i}{\tau}\right)}{\sum_{j=1}^{N} \exp\left(\frac{I_1^\top T_j}{\tau}\right)}$$

Figure from Radford et al. (2021) https://arxiv.org/pdf/2103.00020

SigLIP



- The CLIP objective function works fine if your batch sizes are small enough to fit on a single machine
- But it creates communication overhead if you need to spread the above matrix across many machines
- SigLIP partly solves this issue by replacing the softmax with a sigmoid

CLIP objective function:

$$\sum_{i=1}^{N}\left[\log\frac{\exp\left(\frac{I_i^\top T_i}{\tau}\right)}{\sum_{j=1}^{N}\exp\left(\frac{I_i^\top T_j}{\tau}\right)}+\log\frac{\exp\left(\frac{I_i^\top T_i}{\tau}\right)}{\sum_{j=1}^{N}\exp\left(\frac{I_j^\top T_i}{\tau}\right)}\right]$$

SigLIP objective function:

$$\max\sum_{i=1}^{N}\sum_{j=1}^{N}\log\frac{1}{1+\exp\left(y_{ij}\left(-wI_i^\top T_j+b\right)\right)}$$

where $y_{ij}=\begin{cases}1 & \text{if } i=j \text{ (matching pair)}\\ -1 & \text{otherwise.}\end{cases}$

and $w=\exp(w')$ and b are learnable parameters

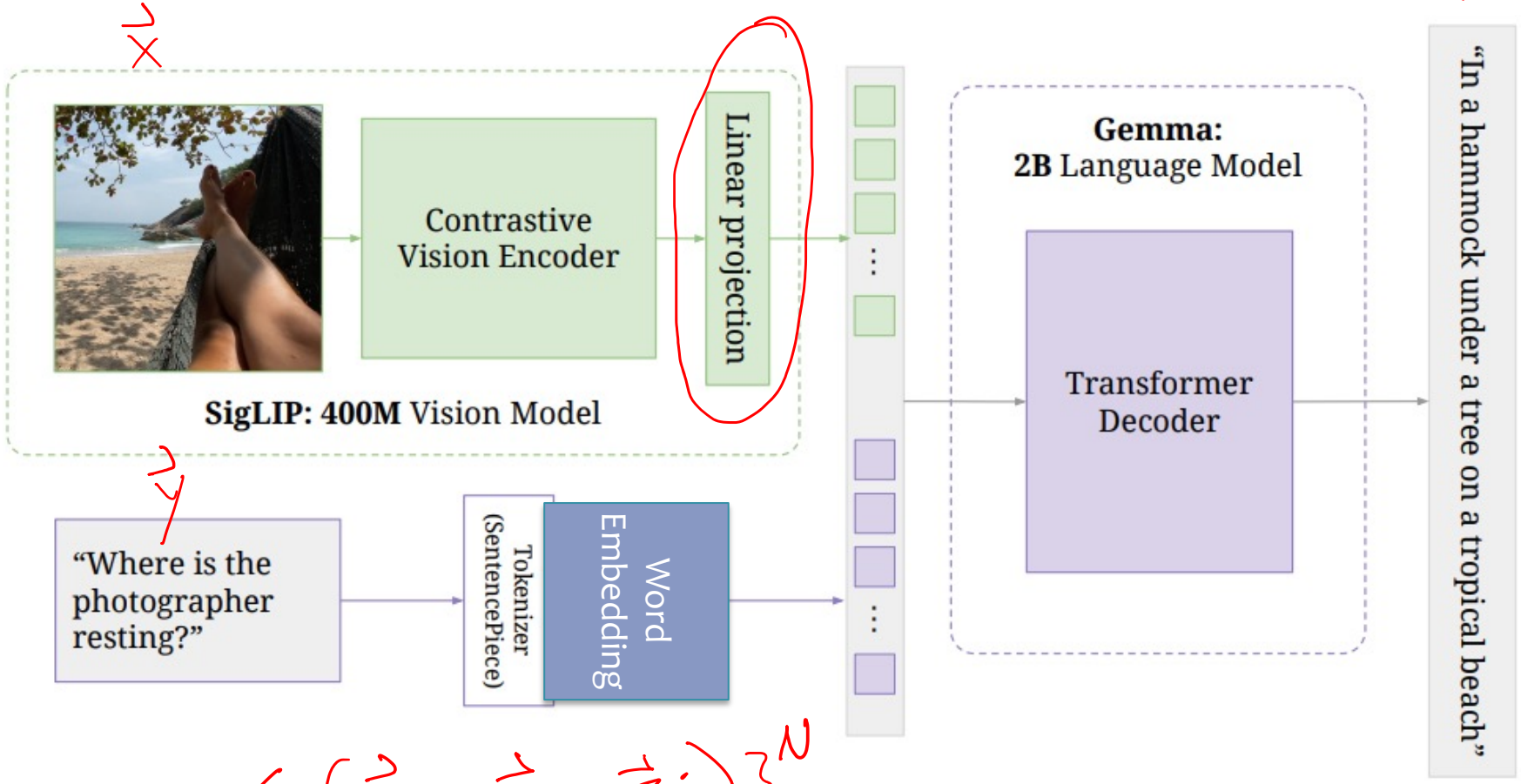
Figure from Radford et al. (2021) https://arxiv.org/pdf/2103.00020

(VLMs that read text/images but only write text)

VLMS WITH TEXT-ONLY DECODERS

PaliGemma

- SigLIP is a variant of CLIP
- Gemma is a 2B LLM (open source counterpart to Gemini)
- Linear projection is creating image embeddings, but they live in the same high dimensional space as the word embeddings
- The LLM has to figure out how to use the image embeddings in some useful way so as to maximize the likelihood of the correct text response



$$D = \{(\vec{x}_i, \vec{y}_i, \vec{z}_i)\}_{i=1}^{N}$$

Llava

Figure from https://arxiv.org/pdf/2407.07726

Qwen-VL

Freezing subsets of the model parameters (corresponding to pretrained submodels) is common during training

Qwen-VL as an example:

1. Qwen-VL first freezes the LLM in order to learn effective image embeddings that align with the word space
2. Then, all parameters are unfrozen for a while
3. Finally, it freezes only the ViT image encoder

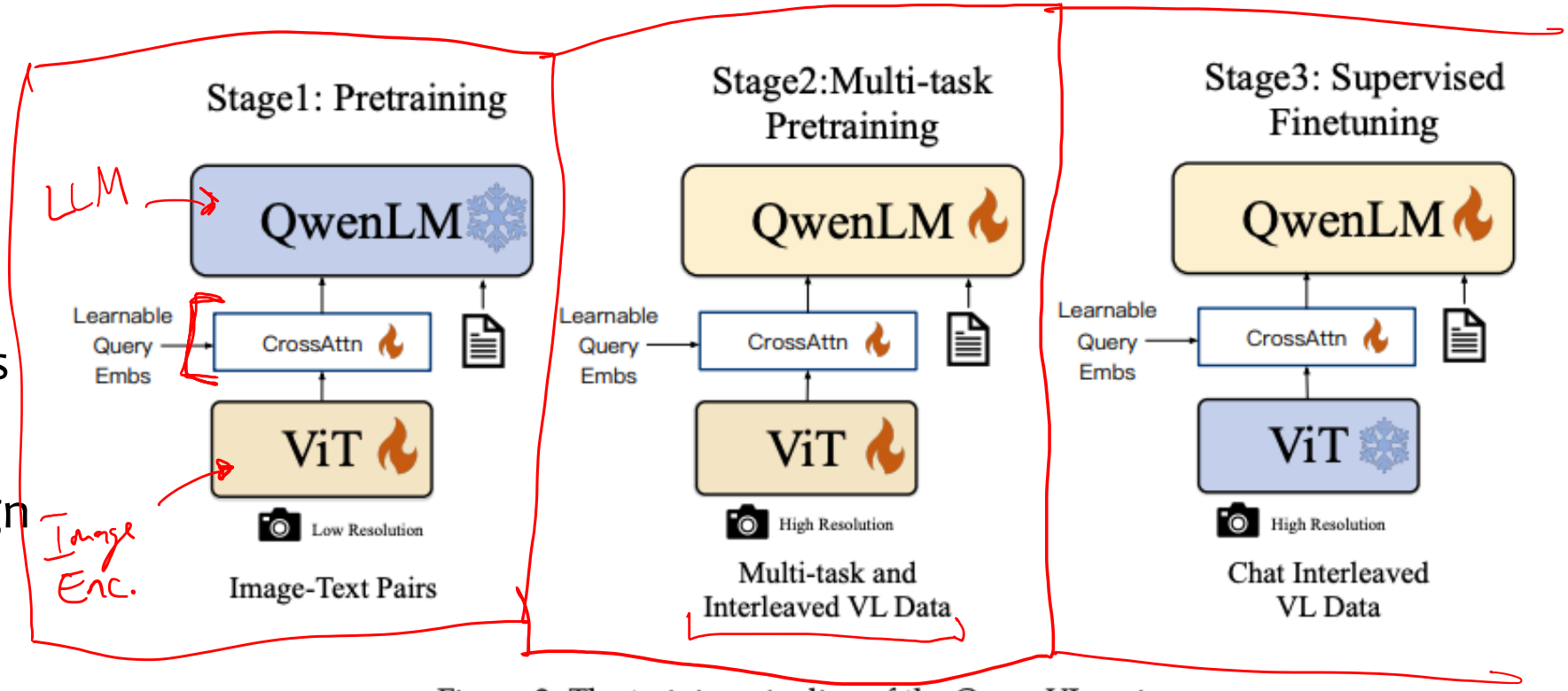


Figure 3: The training pipeline of the Qwen-VL series.

Figure from https://arxiv.org/pdf/2308.12966

Qwen-VL

Just as LLMs are typically evaluated on a variety of text benchmarks assessing the understanding / generation abilities of the model…

…VLMs are evaluated on a variety of text/image benchmarks assessing their ability to interpret and respond to queries about multimodal data.

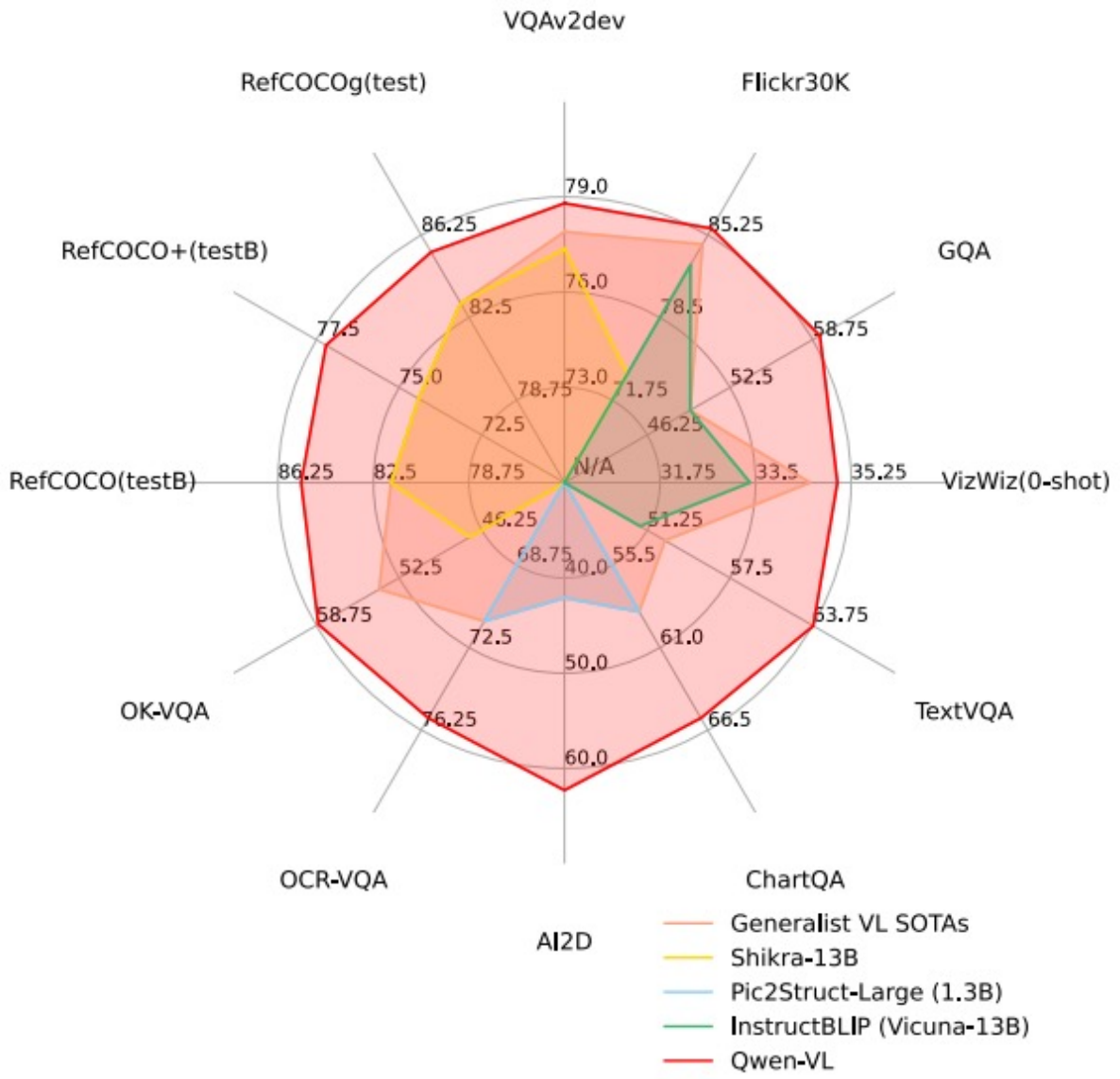


Figure from https://arxiv.org/pdf/2308.12966

Llama 3.2 Vision

Just as LLMs are typically evaluated on a variety of text benchmarks assessing the understanding / generation abilities of the model…

…VLMs are evaluated on a variety of text/image benchmarks assessing their ability to interpret and respond to queries about multimodal data.

Vision instruction-tuned benchmarks

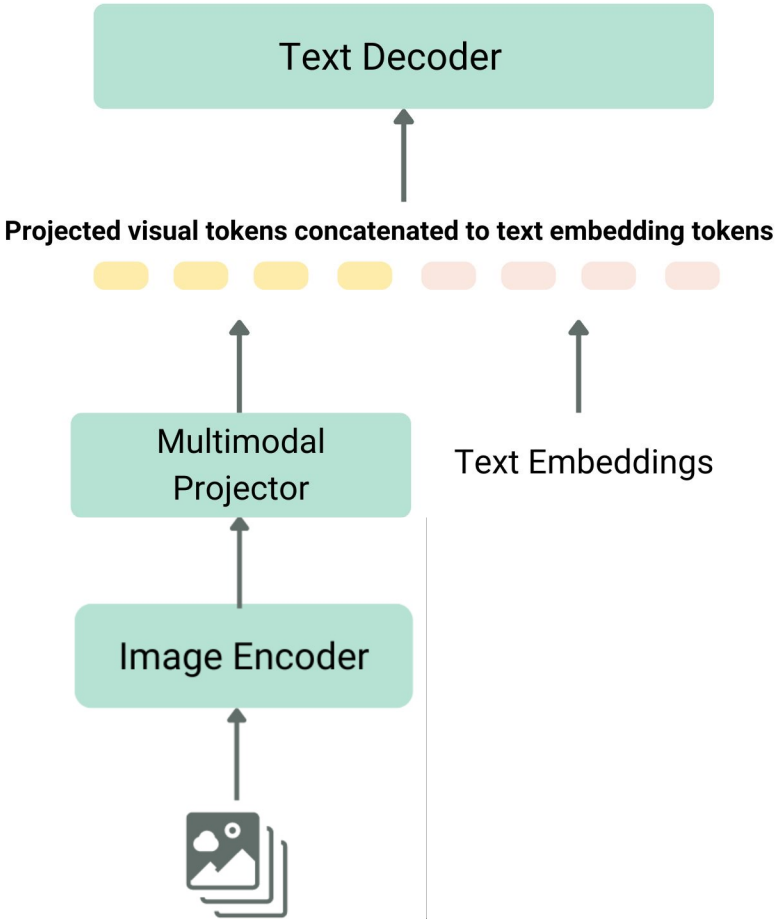
Modality	Category / Benchmark	Llama 3.2 11B	Llama 3.2 90B	Claude 3 – Haiku	GPT-4o-mini
Image	College-level Problems and Mathematical Reasoning / MMMU (val, 0-shot CoT, micro avg accuracy)	50.7	**60.3**	50.2	59.4
	MMMU-Pro, Standard (10 opts, test)	33.0	**45.2**	27.3	42.3
	MMMU-Pro, Vision (test)	23.7	33.8	20.1	**36.5**
	MathVista (testmini)	51.5	**57.3**	46.4	56.7
	Charts and Diagram Understanding / ChartQA (test, 0-shot CoT relaxed accuracy)*	83.4	85.5	81.7	—
	AI2 Diagram (test)*	91.1	92.3	86.7	—
	DocVQA (test, ANLS)*	88.4	90.1	88.8	—
	General Visual Question Answering / VQAv2 (test)	75.2	78.1	—	—
Text	General / MMLU (0-shot, CoT)	73.0	**86.0**	75.2 (5-shot)	82.0
	Math / MATH (0-shot, CoT)	51.9	68.0	38.9	**70.2**
	Reasoning / GPQA (0-shot, CoT)	32.8	**46.7**	33.3	40.2
	Multilingual / MGSM (0-shot, CoT)	68.9	86.9	75.1	**87.0**

Figure from https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/

VISION LANGUAGE MODELS (VLMS)

VLM: Architecture

- High-level idea: convert both the image and the text inputs into embedding vectors, then pass those vectors into a decoder-only transformer and do next (text) token prediction



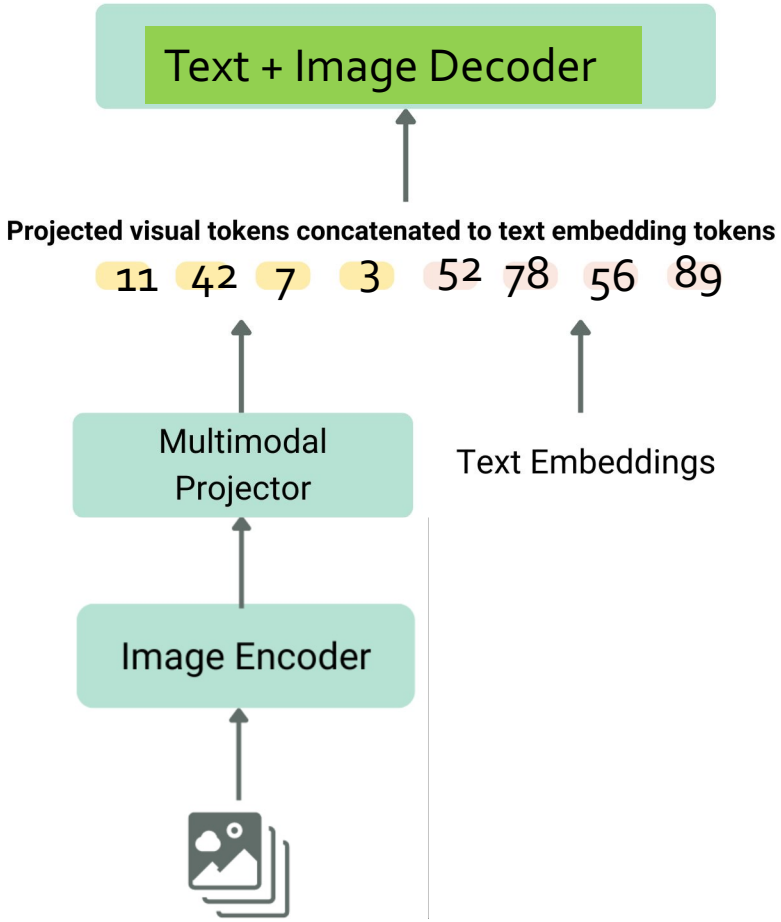
- Two common encoders:
  - VQ-VAE encoder followed by an embedding layer that converts the discrete tokens into dense numerical vectors
  - CLIP encoder, that directly learns an embedding vector using a contrastive pre-training objective

Slide from Henry Chai

Source: https://huggingface.co/blog/vlms

VLM: Architecture

- High-level idea: convert both the image and the text inputs into **integers**, then pass those **integers** into a decoder-only transformer and do next (**text or image**) token prediction

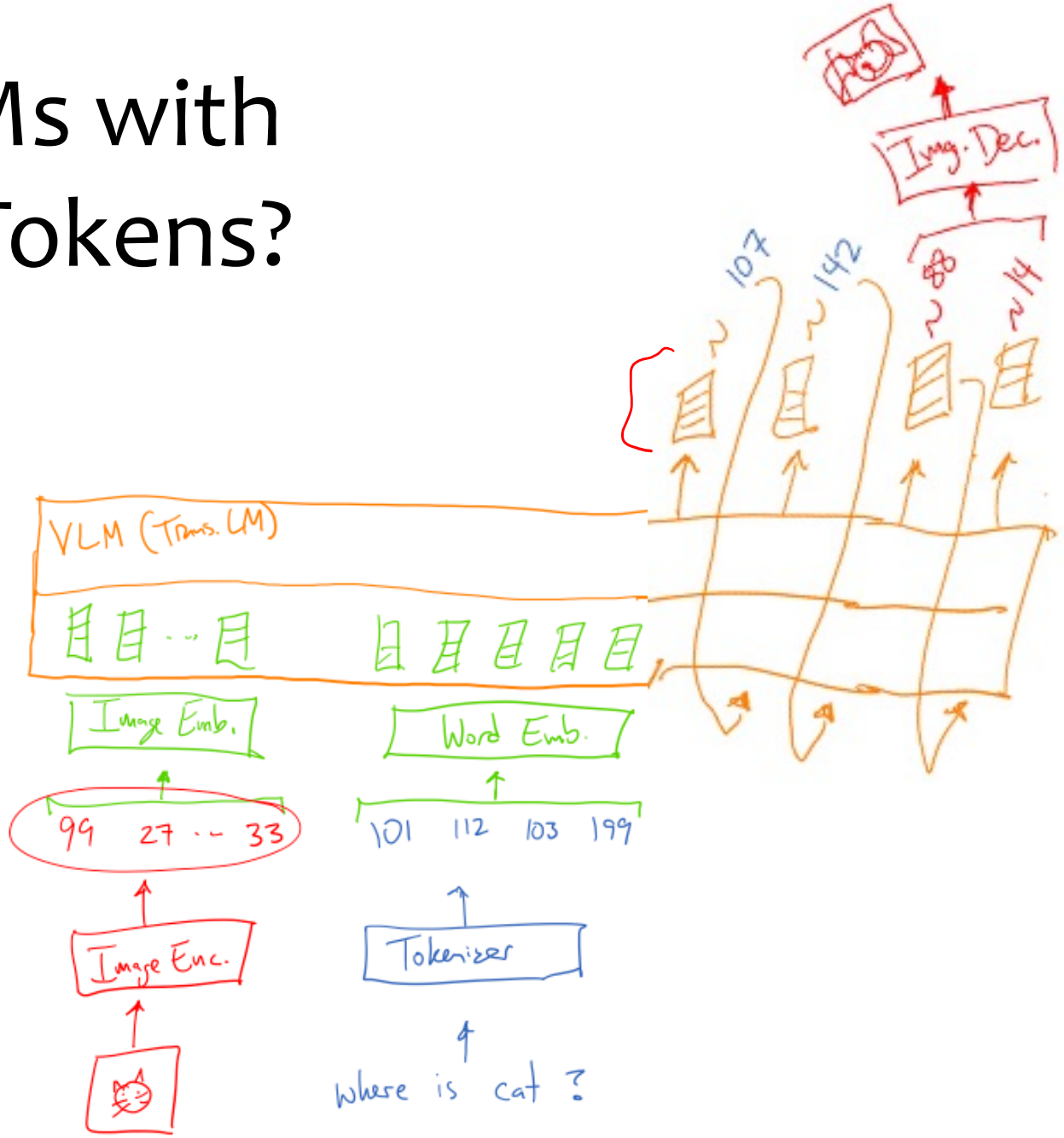


- Two common encoders:
  - VQ-VAE encoder followed by an embedding layer that converts the discrete tokens into dense numerical vectors
  - CLIP encoder, that directly learns an embedding vector using a contrastive pre-training objective

Slide from Henry Chai

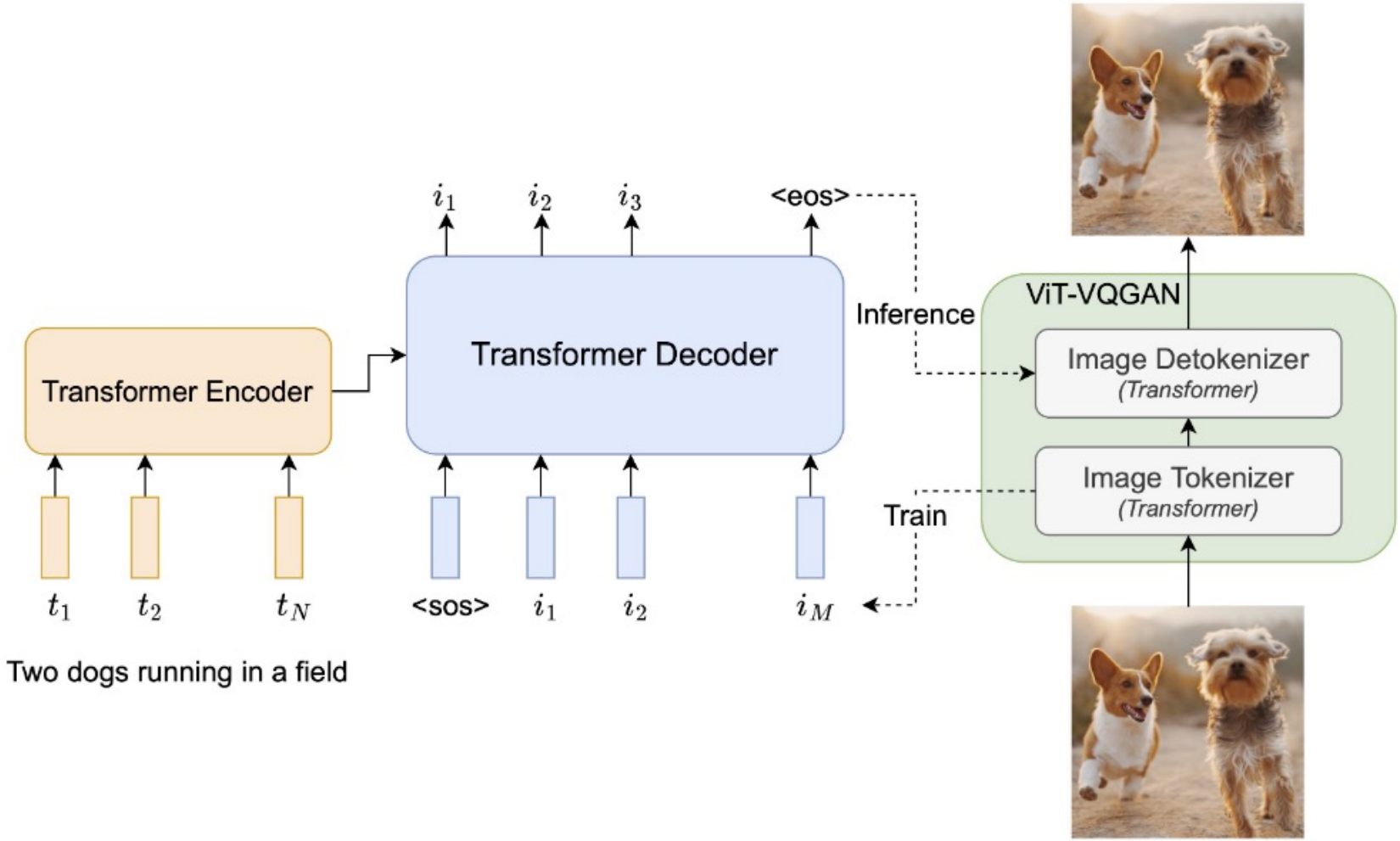
Source: https://huggingface.co/blog/vlms

Why VLMs with Integer Tokens?



VQ-VAES

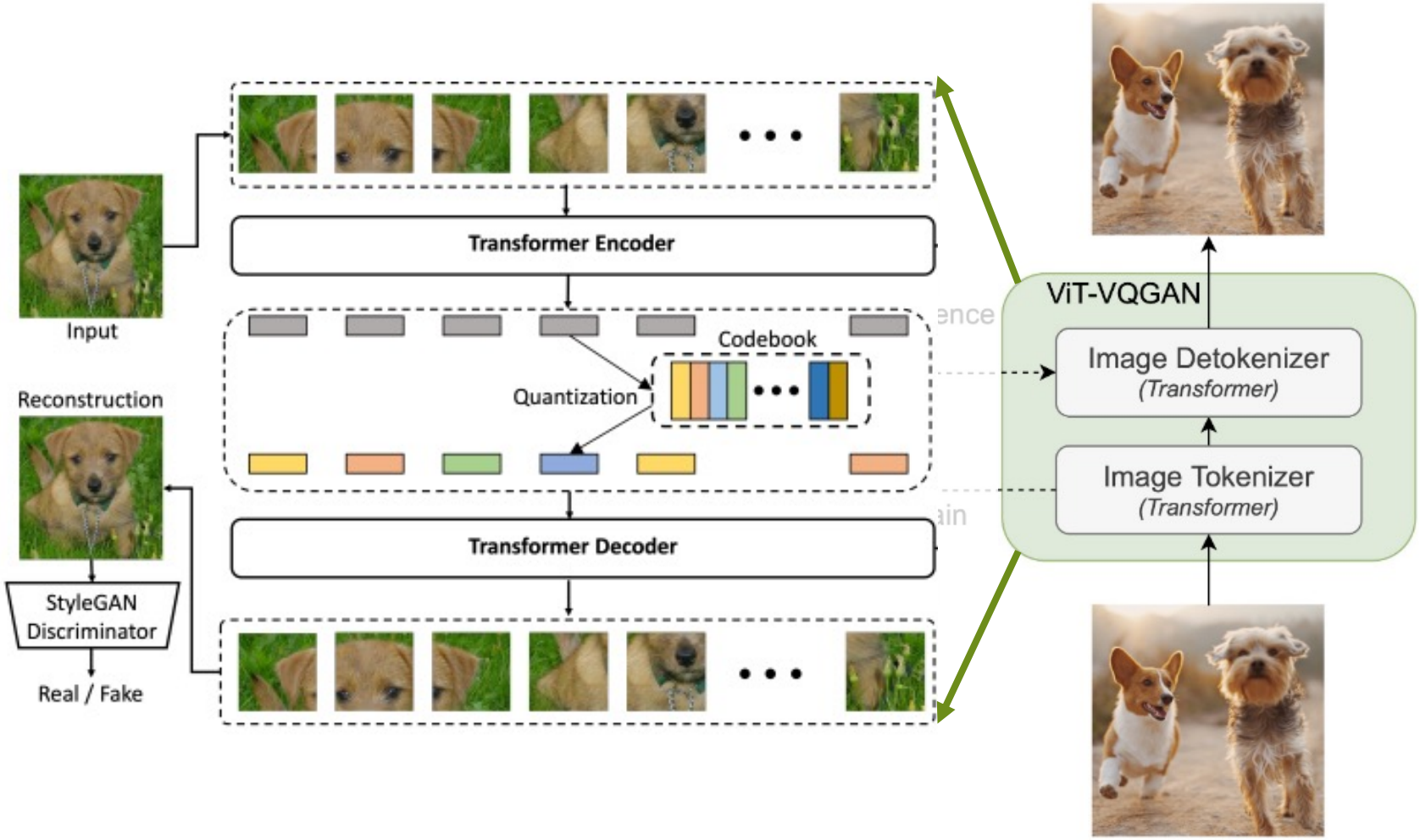
Recall: Parti



Slide from Henry Chai

Source: https://arxiv.org/pdf/2206.10789

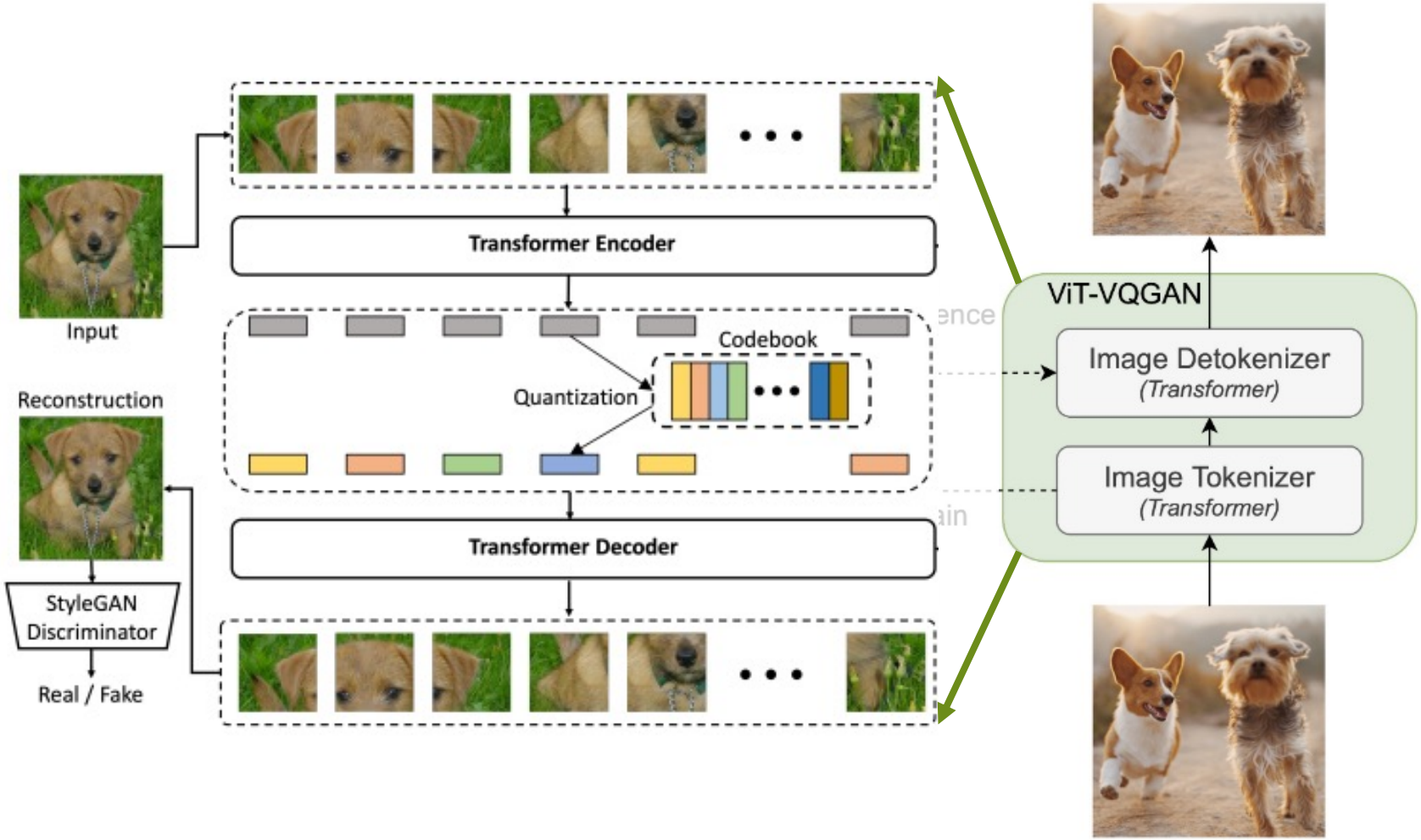
Recall: Image Tokenization



Slide from Henry Chai

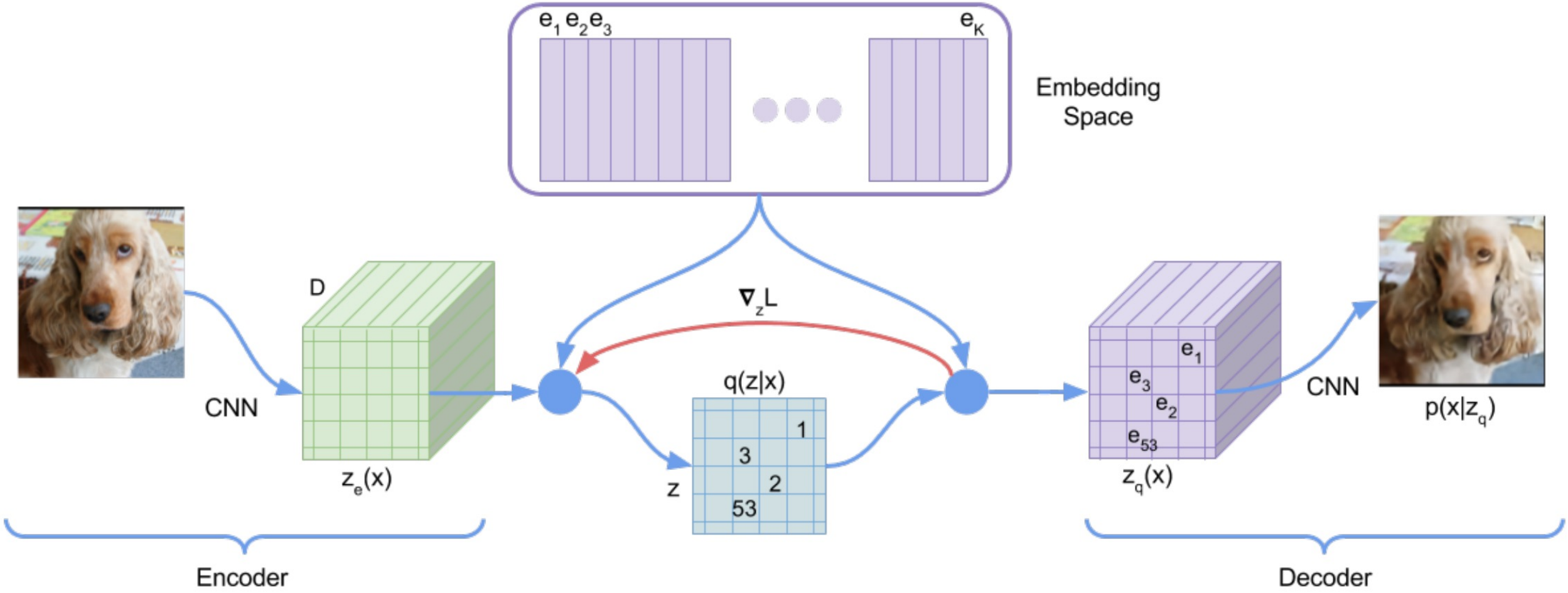
Source: https://arxiv.org/pdf/2110.04627

How can we (pre-)train these models given the non-differentiable quantization operation?



Slide from Henry Chai

Source: https://arxiv.org/pdf/2110.04627

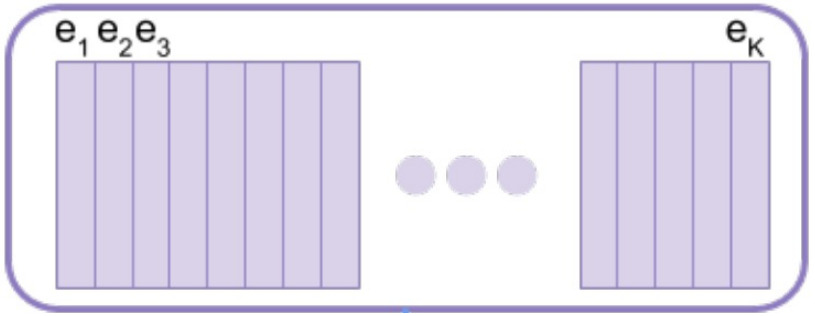


Vector-Quantized VAEs

Slide from Henry Chai

Source: https://arxiv.org/pdf/1711.00937

$K = 64$

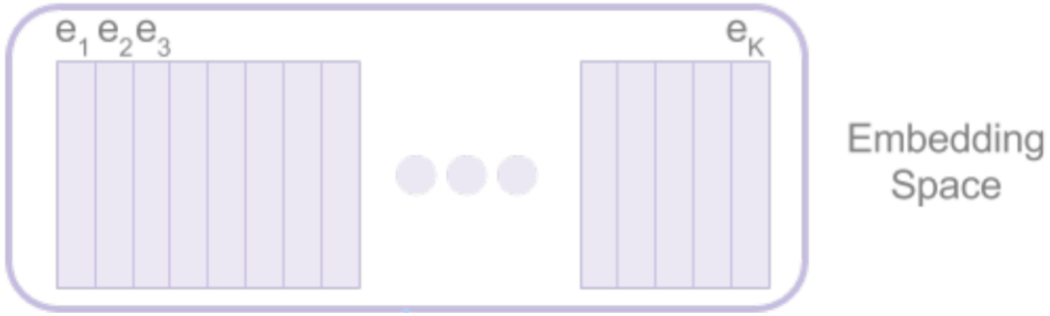


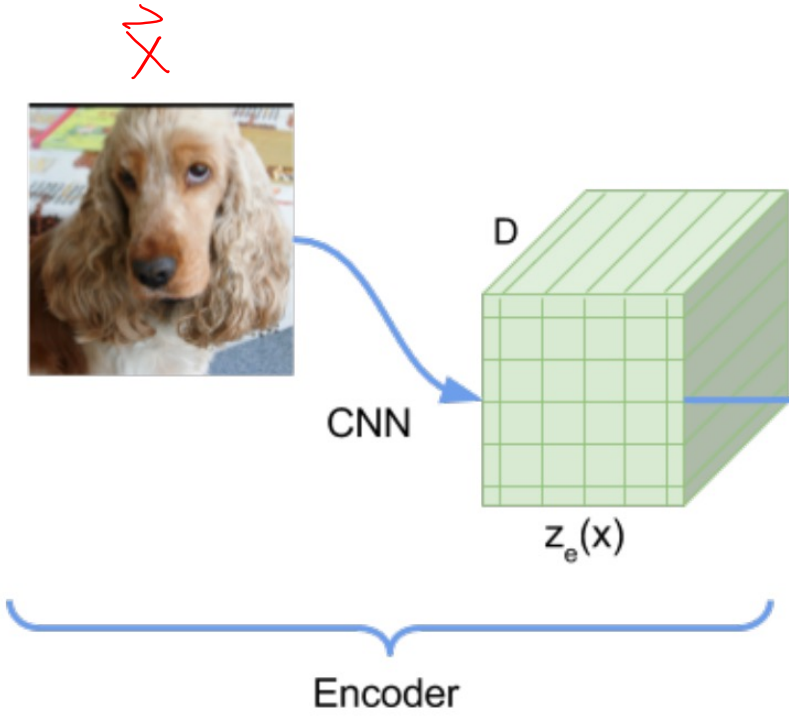
- Embedding space consists of K D-dimensional latent vectors $\{e_1, \ldots, e_K\}$ which are learned during training
- The indices $[1, \ldots, K]$ of each latent vector correspond to the "image tokens" in some fixed-length codebook

Vector-Quantized VAEs

Slide from Henry Chai

Source: https://arxiv.org/pdf/1711.00937



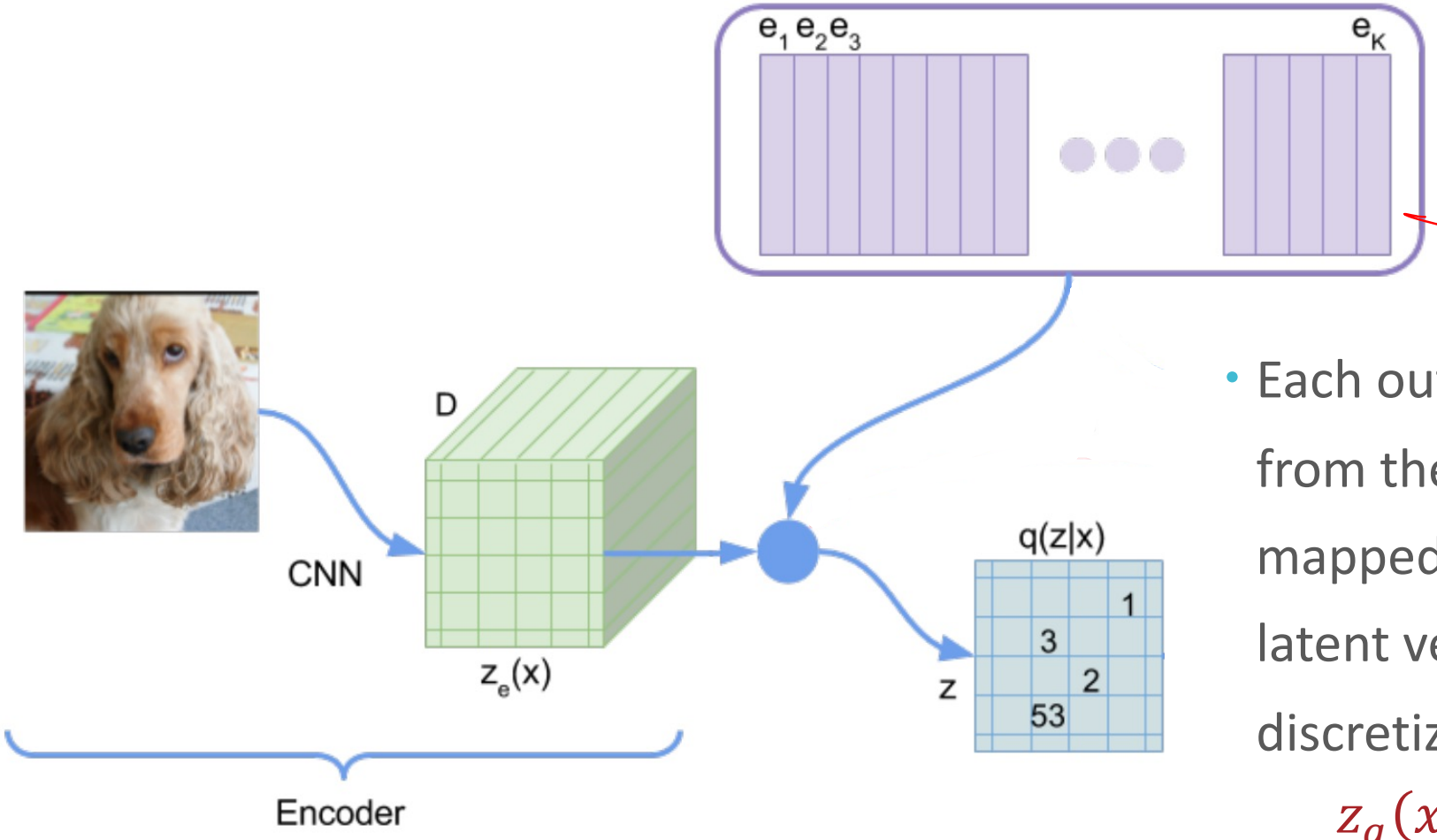


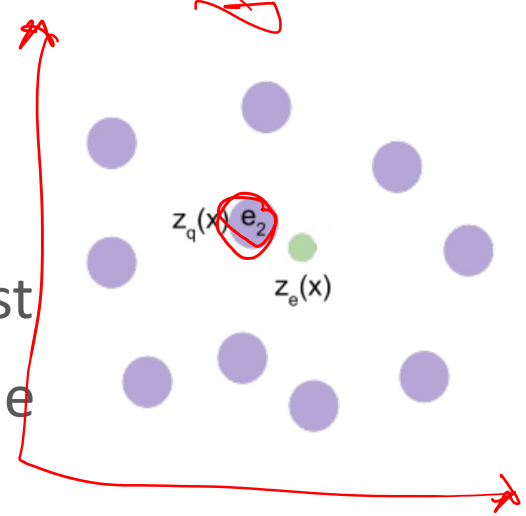
- The encoder (e.g., a ResNet-like CNN) maps images to N D-dimensional vectors

Vector-Quantized VAEs

Slide from Henry Chai

Source: https://arxiv.org/pdf/1711.00937





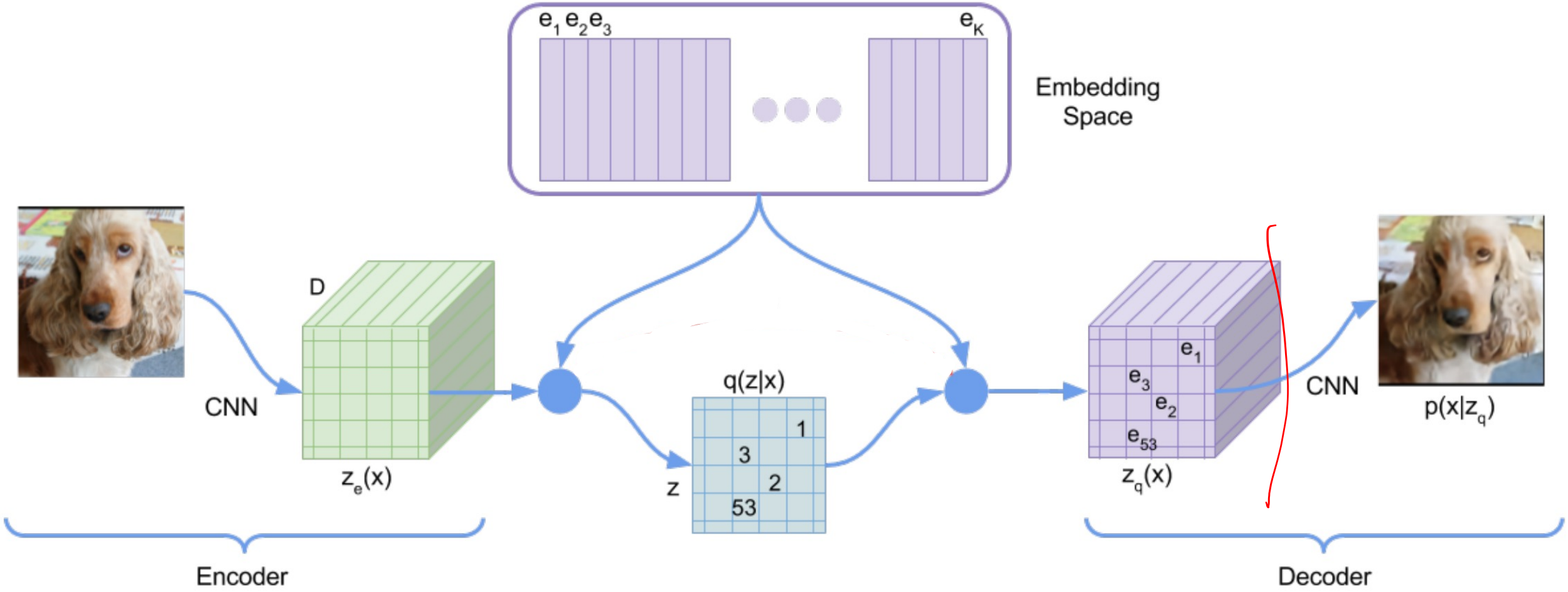
- Each output vector from the encoder is mapped to the nearest latent vector to get the discretized encoding

$$z_q(x) = \underset{e \in \{e_1,\ldots,e_K\}}{\operatorname{argmin}} \|z_e(x) - e\|_2^2$$

Vector-Quantized VAEs

Slide from Henry Chai

Source: https://arxiv.org/pdf/1711.00937

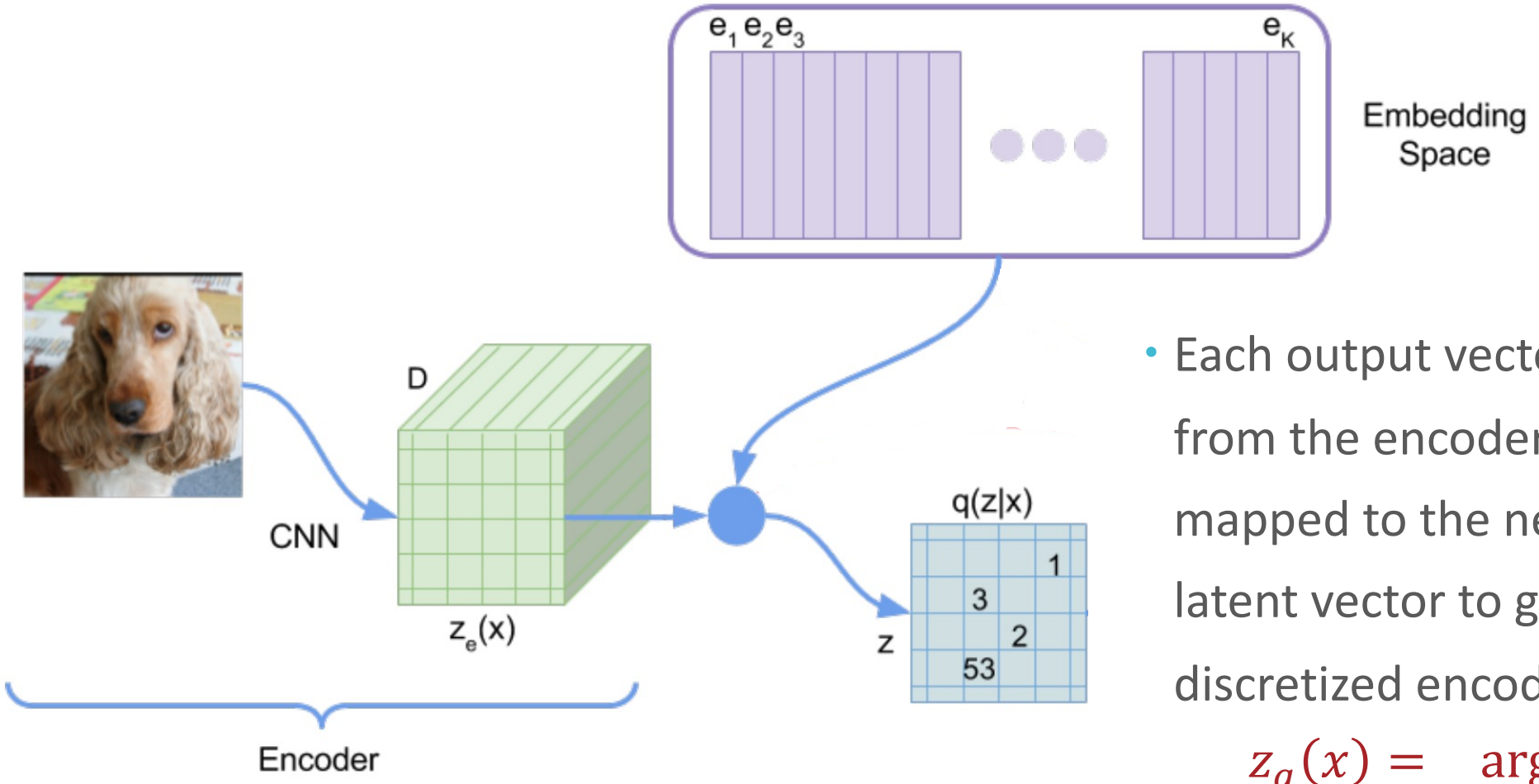


- The decoder takes the discretized representation and recreates the original image

Vector-Quantized VAEs

Slide from Henry Chai

Source: https://arxiv.org/pdf/1711.00937



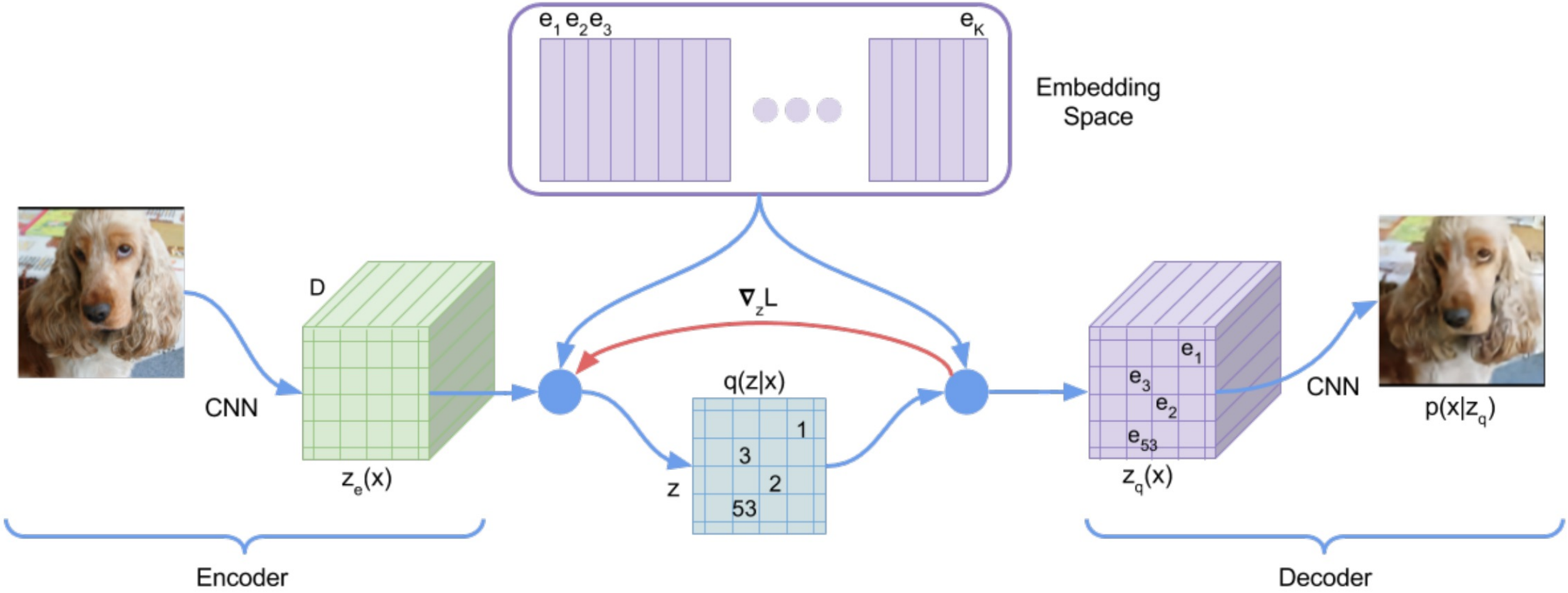
- Each output vector from the encoder is mapped to the nearest latent vector to get the discretized encoding

$$z_q(x) = \underset{e \in \{e_1,\ldots,e_K\}}{\operatorname{argmin}} \|z_e(x) - e\|_2^2$$

Wait, how would we take the gradient through the argmin?

Slide from Henry Chai

Source: https://arxiv.org/pdf/1711.00937

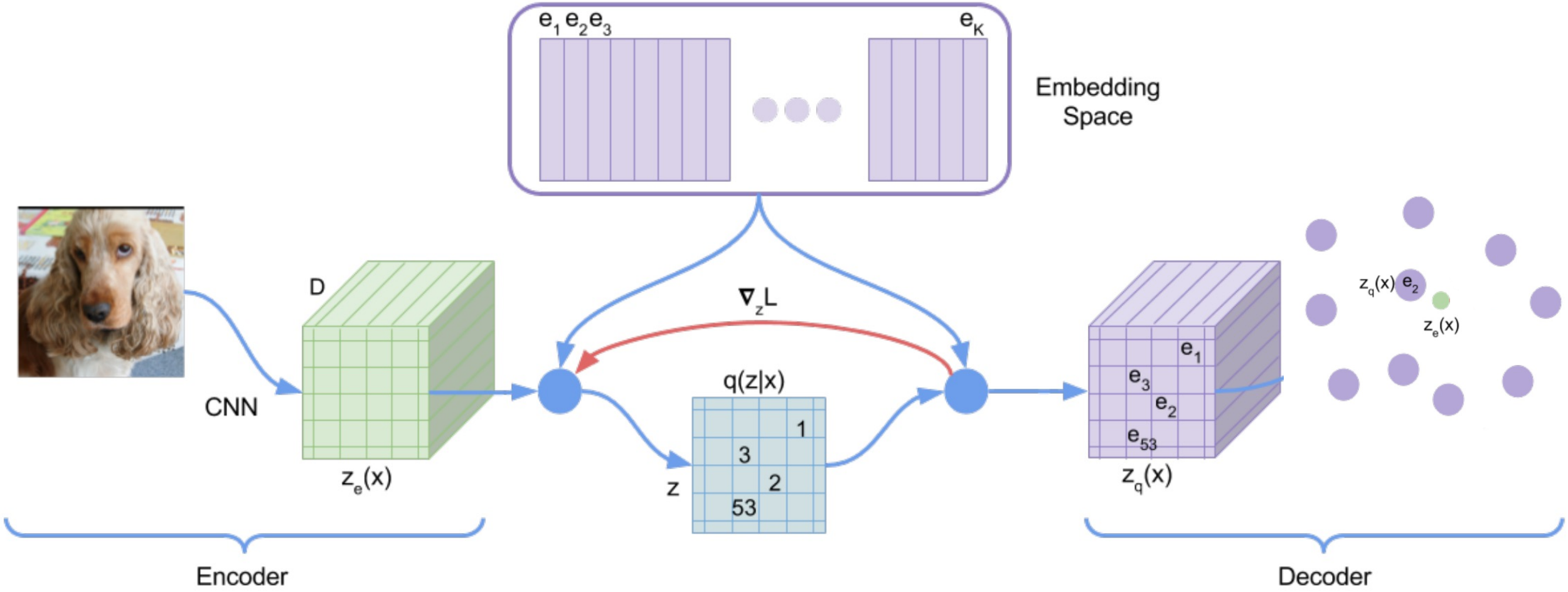


- Treat the gradient w.r.t. $z_q(x)$ as an estimate of the gradient w.r.t. $z_e(x)$

Straight-through Estimator

Slide from Henry Chai

Source: https://arxiv.org/pdf/1711.00937



- Intuition: the closer $z_q(x)$ and $z_e(x)$, the better the estimate (under certain assumptions)

Straight-through Estimator

Slide from Henry Chai

Source: https://arxiv.org/pdf/1711.00937

VQ-VAE Objective Function

- Intuition: we want the latent vectors to correspond to relevant points in the embedding space i.e., ones that are near the outputs of the encoder
- However, we also want the encoder to respect the latent vectors and not overfit to the training dataset
- Idea: augment the standard VAE objective with some regularizing terms that drive the two closer to each other

$$-\log p_\theta\left(x \middle| z_q(x)\right) + \left\|\mathrm{sg}[z_e(x)] - z_q(x)\right\|_2^2 + \beta\left\|z_e(x) - \mathrm{sg}[z_q(x)]\right\|_2^2$$

where sg is the stop-gradient operator which fixes the argument to be non-updated constant

Slide from Henry Chai

VQ-VAE Objective Function

- Intuition: we want the latent vectors to correspond to relevant points in the embedding space i.e., ones that are near the outputs of the encoder
- However, we also want the encoder to respect the latent vectors and not overfit to the training dataset
- Idea: augment the standard VAE objective with some regularizing terms that drive the two closer to each other

$$-\log p_\theta\left(x \middle| z_q(x)\right) + \left\|\mathrm{sg}[z_e(x)] - z_q(x)\right\|_2^2 + \beta\left\|z_e(x) - \mathrm{sg}[z_q(x)]\right\|_2^2$$

- The first term is the typical reconstruction error objective

Slide from Henry Chai

VQ-VAE Objective Function

- Intuition: we want the latent vectors to correspond to relevant points in the embedding space i.e., ones that are near the outputs of the encoder
- However, we also want the encoder to respect the latent vectors and not overfit to the training dataset
- Idea: augment the standard VAE objective with some regularizing terms that drive the two closer to each other

$$-\log p_\theta\left(x \middle| z_q(x)\right) + \left\|\mathrm{sg}[z_e(x)] - z_q(x)\right\|_2^2 + \beta\left\|z_e(x) - \mathrm{sg}[z_q(x)]\right\|_2^2$$

- The second term drives the latent vector to be closer to the encoder output vector that was mapped to it

Slide from Henry Chai

VQ-VAE Objective Function

- Intuition: we want the latent vectors to correspond to relevant points in the embedding space i.e., ones that are near the outputs of the encoder
- However, we also want the encoder to respect the latent vectors and not overfit to the training dataset
- Idea: augment the standard VAE objective with some regularizing terms that drive the two closer to each other

$$-\log p_\theta\left(x\middle|z_q(x)\right) + \left\|\mathrm{sg}[z_e(x)] - z_q(x)\right\|_2^2 + \beta\left\|z_e(x) - \mathrm{sg}[z_q(x)]\right\|_2^2$$

- The third term drives the encoder to output vectors closer to the latent vectors

Slide from Henry Chai

CLIP vs. VQ-VAEs

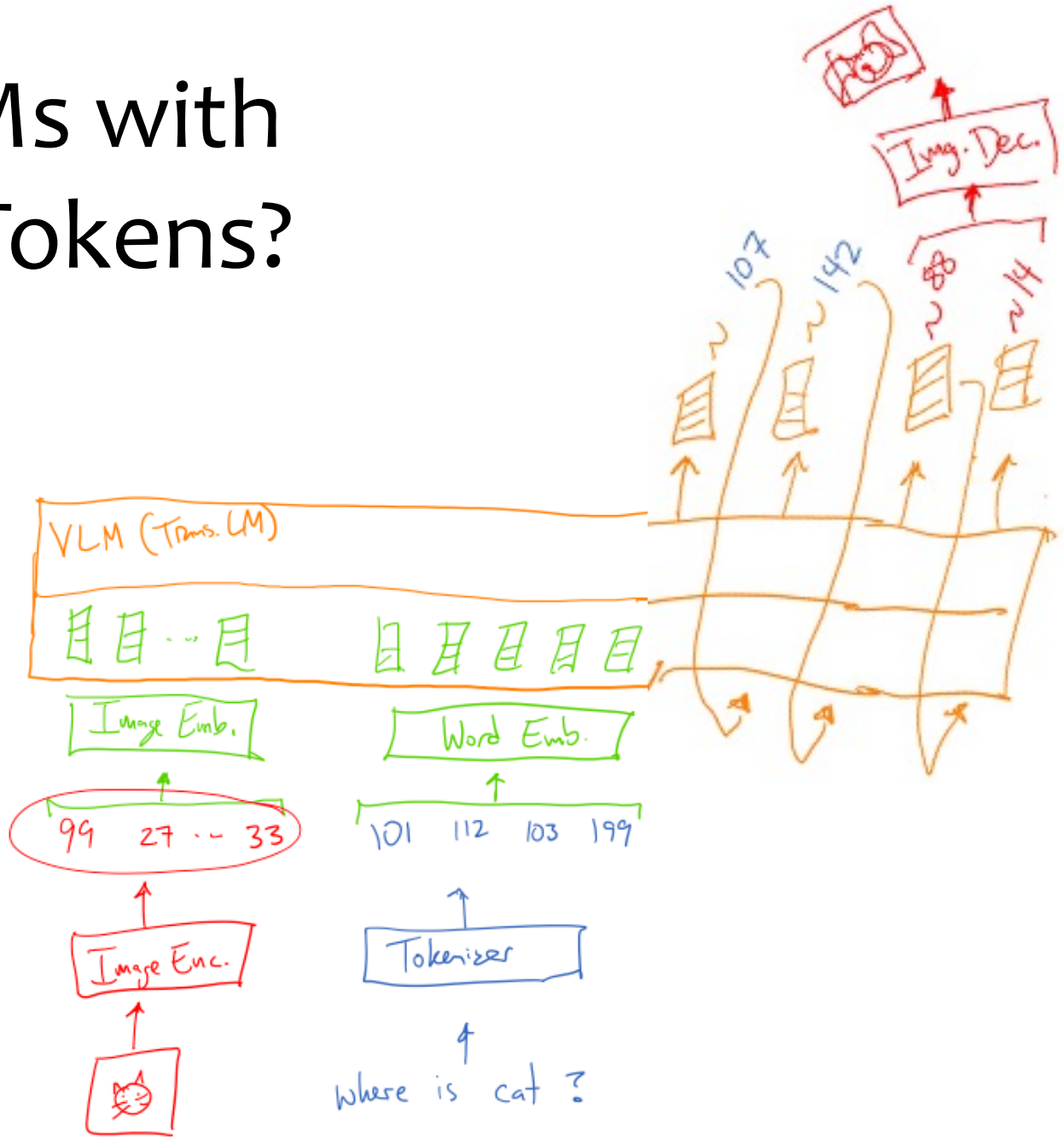
- VLMs with VQ-VAE encoders (or any vector quantized image model) can also generate images in addition to text by defining a loss over the image codebook tokens
- CLIP does not discretize its image embedding so VLMs with CLIP-based encoders cannot (naturally) define a loss over images and thus, can only output text
- However, CLIP embeddings are more expressive than the discrete VQ-VAE encodings so can lead to improved performance in some settings

Slide from Henry Chai

(VLMs that read text/images and write text/images)

VLMS WITH TEXT AND IMAGE DECODERS

Why VLMs with Integer Tokens?



Large World Model

- The Large World Model (LWM) is an example of a Transformer LM that works with both discrete text tokens and discrete image tokens
- Key idea:
  - pretrain your image tokenizer/de-tokenizer (VQGAN)
  - any images in your data can be converted to their discrete representations ahead of time
  - then train your Transformer LM as you would any other LM on discrete tokens
  - at test time, whenever you see a sequence of image tokens, convert them back to an image

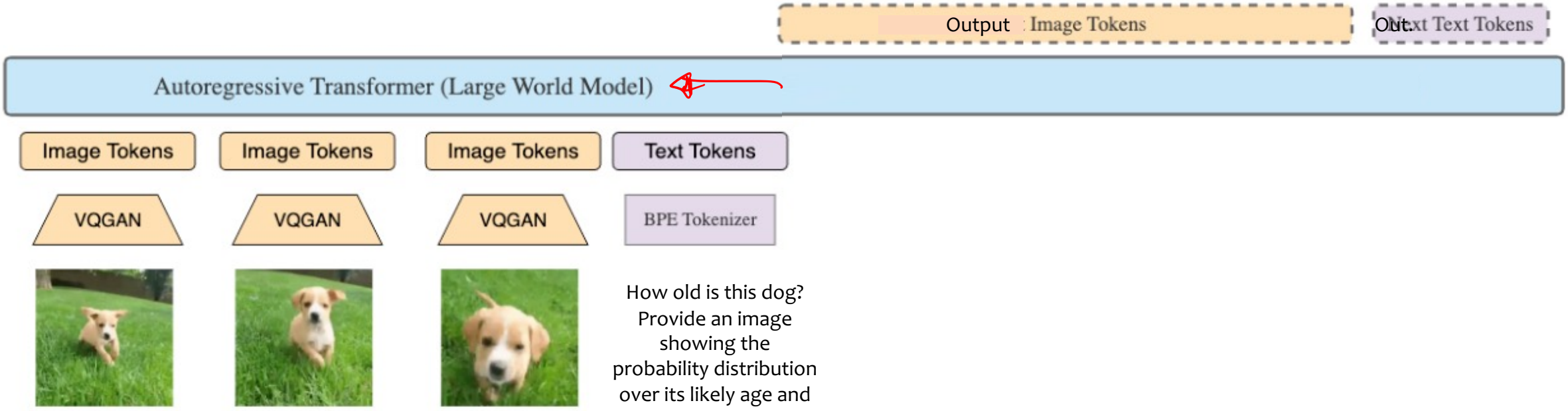


Figure from http://arxiv.org/abs/2402.08268

Large World Model

H More Image Generation Examples

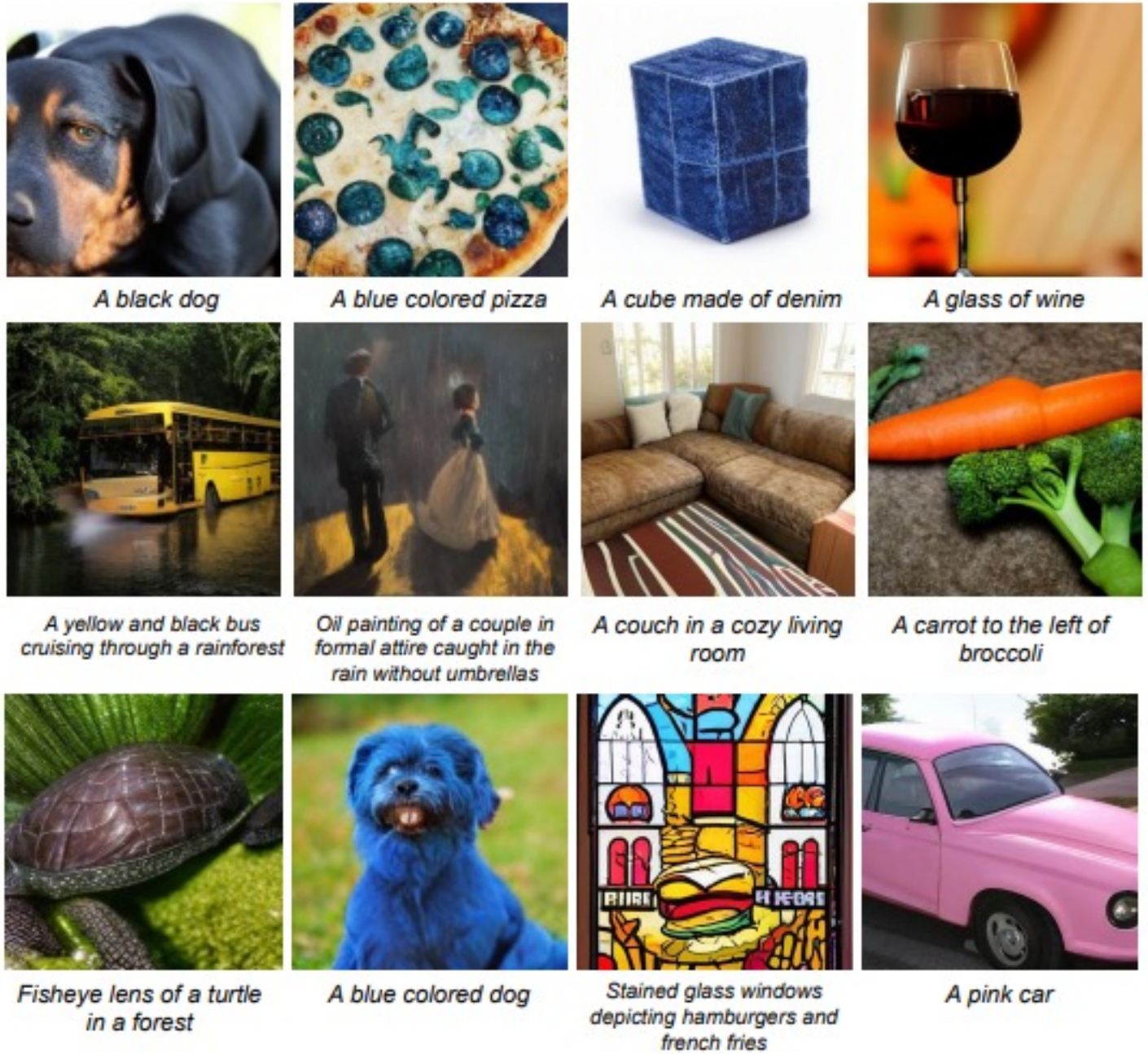
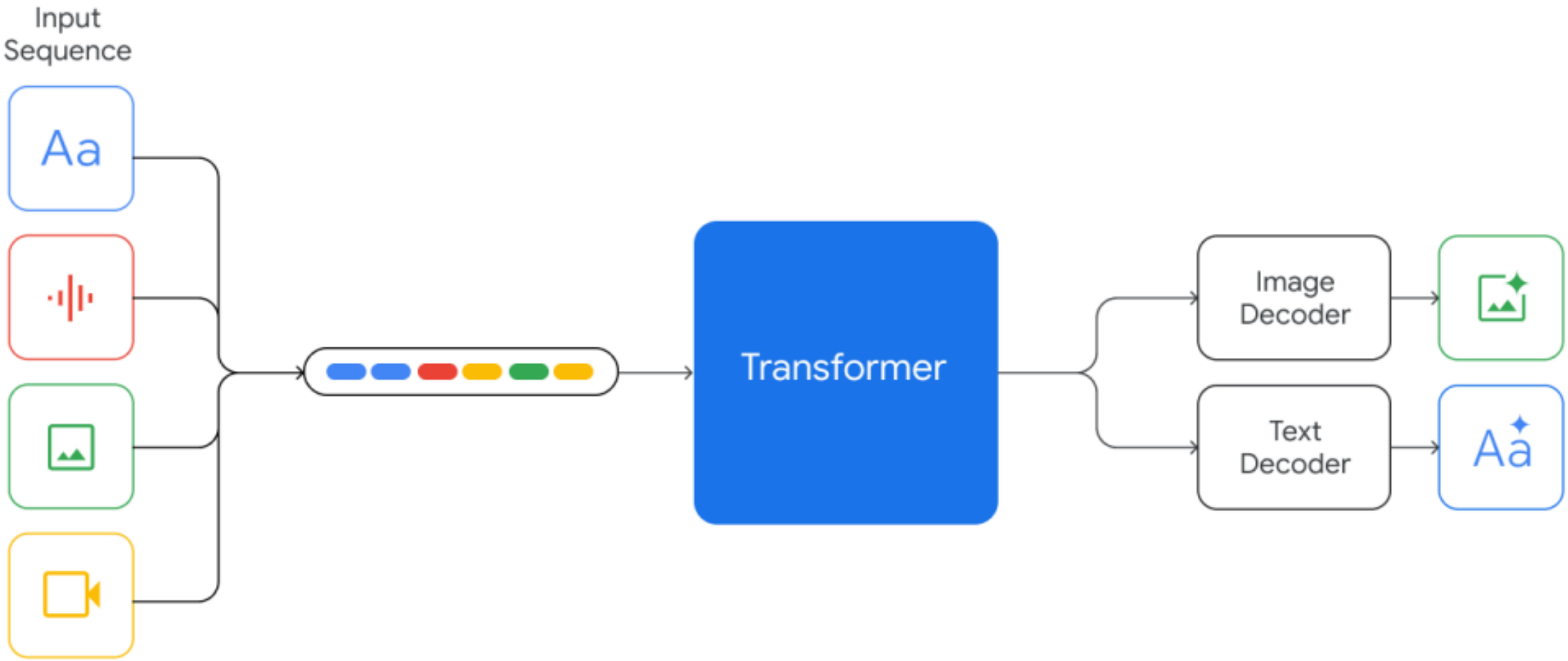


Figure from http://arxiv.org/abs/2402.08268

Gemini


Input
Sequence
Aa
Transformer
Image
Decoder
Text
Decoder
Aa

Gemini

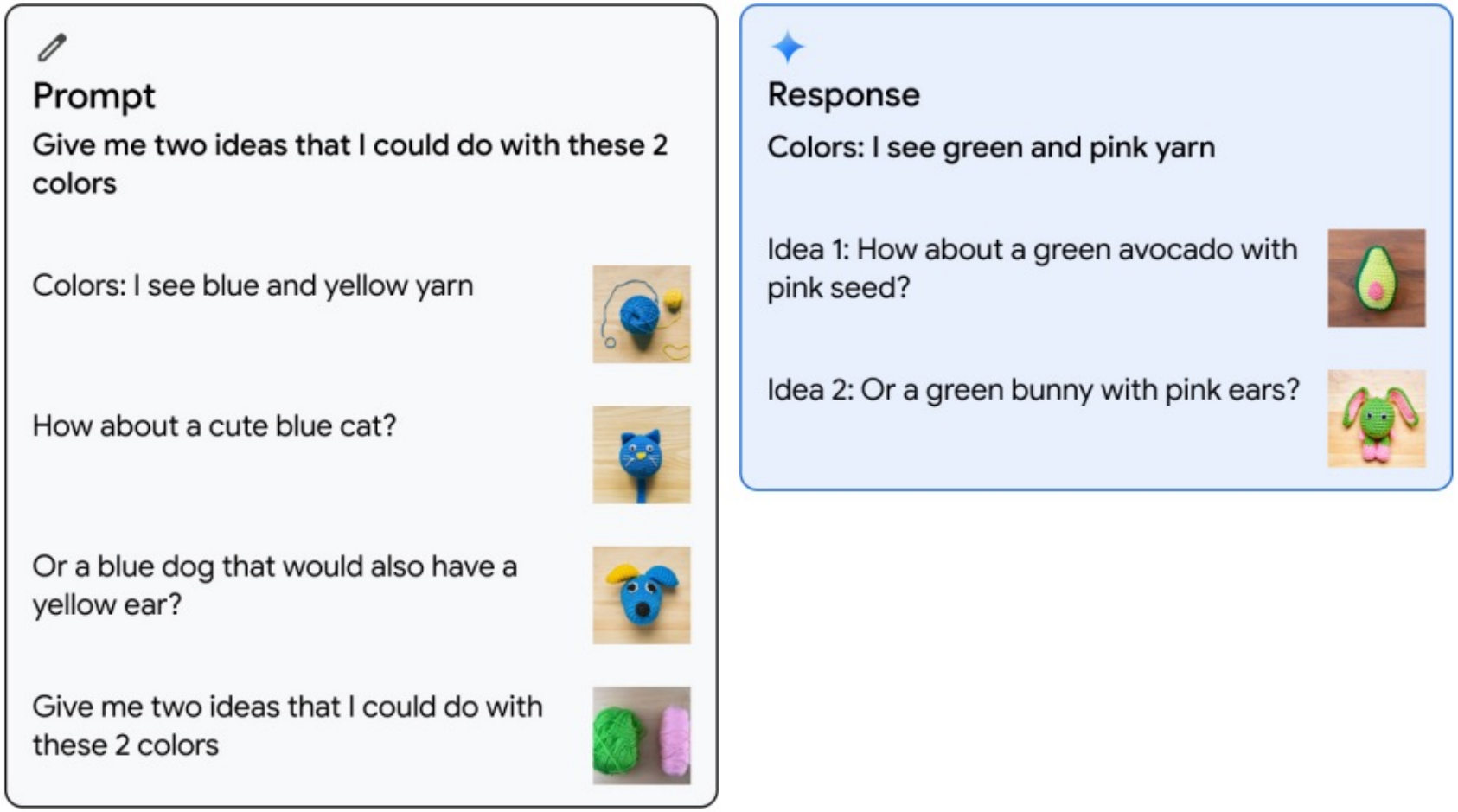


Figure 6 | **Image Generation.** Gemini models can output multiple images interleaved with text given a prompt composed of image and text. In the left figure, Gemini Ultra is prompted in a 1-shot setting with a user example of generating suggestions of creating cat and dog from yarn when given two colors, blue and yellow. Then, the model is prompted to generate creative suggestions with two new colors, pink and green, and it generates images of creative suggestions to make a cute green avocado with pink seed or a green bunny with pink ears from yarn as shown in the right figure.