10-423/623: Generative AI Lecture 14 – Visual-Language Models

Henry Chai & Matt Gormley

10/21/24

Front Matter

- Announcements:
  - HW3 released 10/7, due 10/24 at 11:59 PM
    - Please be mindful of your grace day usage!
  - Project team formation due 10/25 at 11:59 PM
    - Each team should only submit one PDF; see handout for instructions on how to make group submissions in Gradescope
    - Reminder: **you may not take grace days on any project deliverables**
  - Quiz 4 on 10/28 (Monday)
    - Will cover Lectures 12 – 15

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

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

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

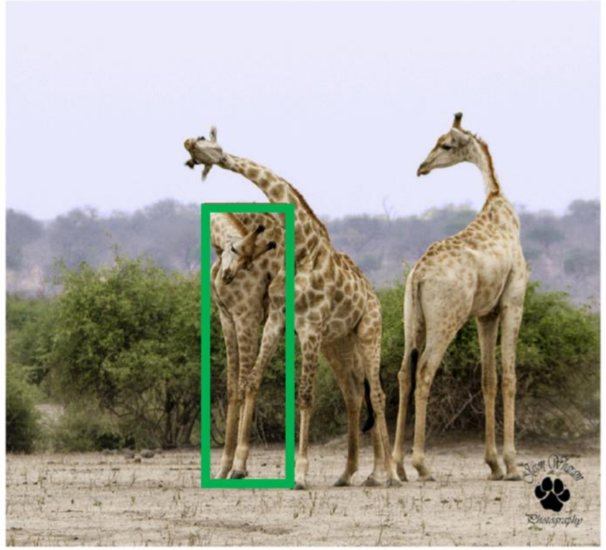
The left image contains twice the number of dogs as the right image, and at least two dogs in total are standing.

One image shows exactly two brown acorns in back-to-back caps on green foliage.

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

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.

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.

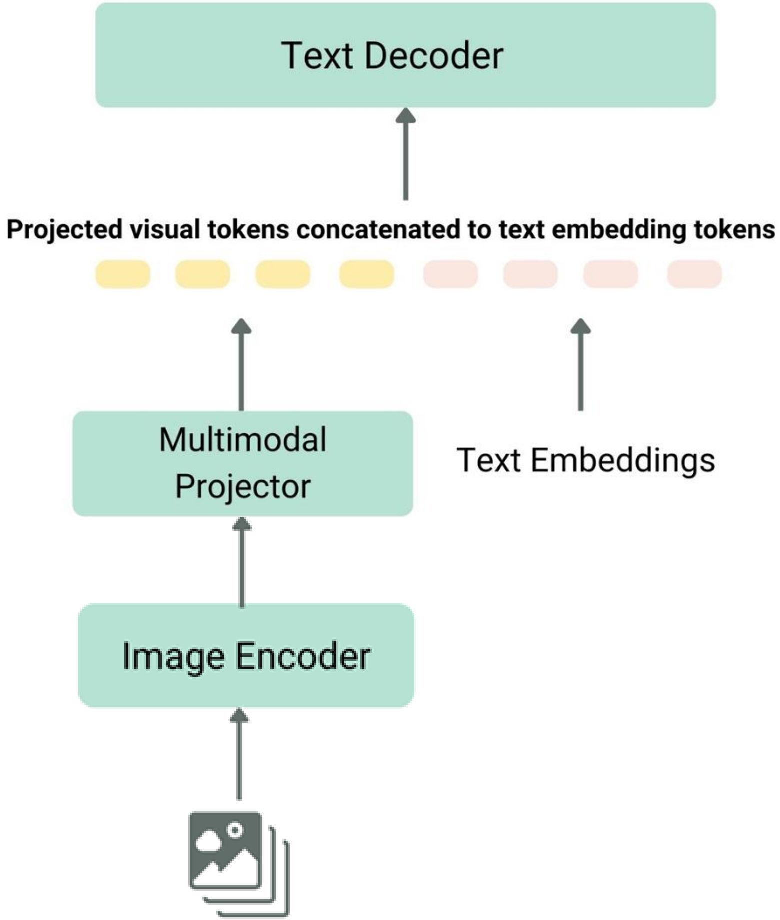
Generated Caption: A woman in a black shirt and sunglasses smiles.

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

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

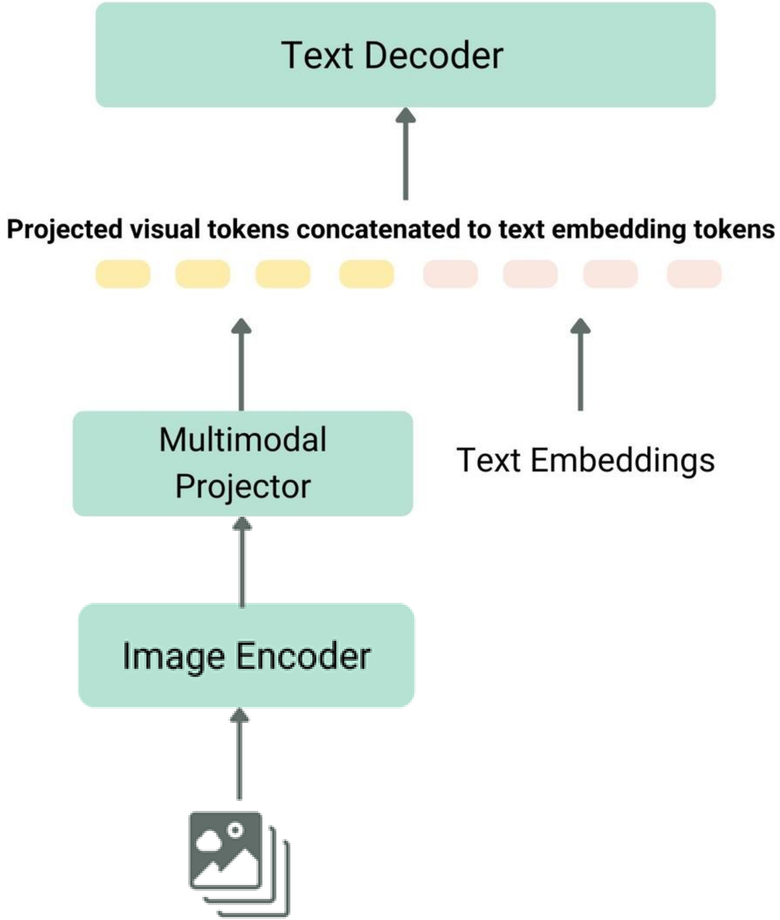


- Two common encoders:
  - VQ-VAE encoder followed by an embedding layer that converts the discrete tokens into dense numerical vectors
  - CLIP encoder, that directly learns an embedding vector using a contrastive pre-training objective

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

VLM: Architecture

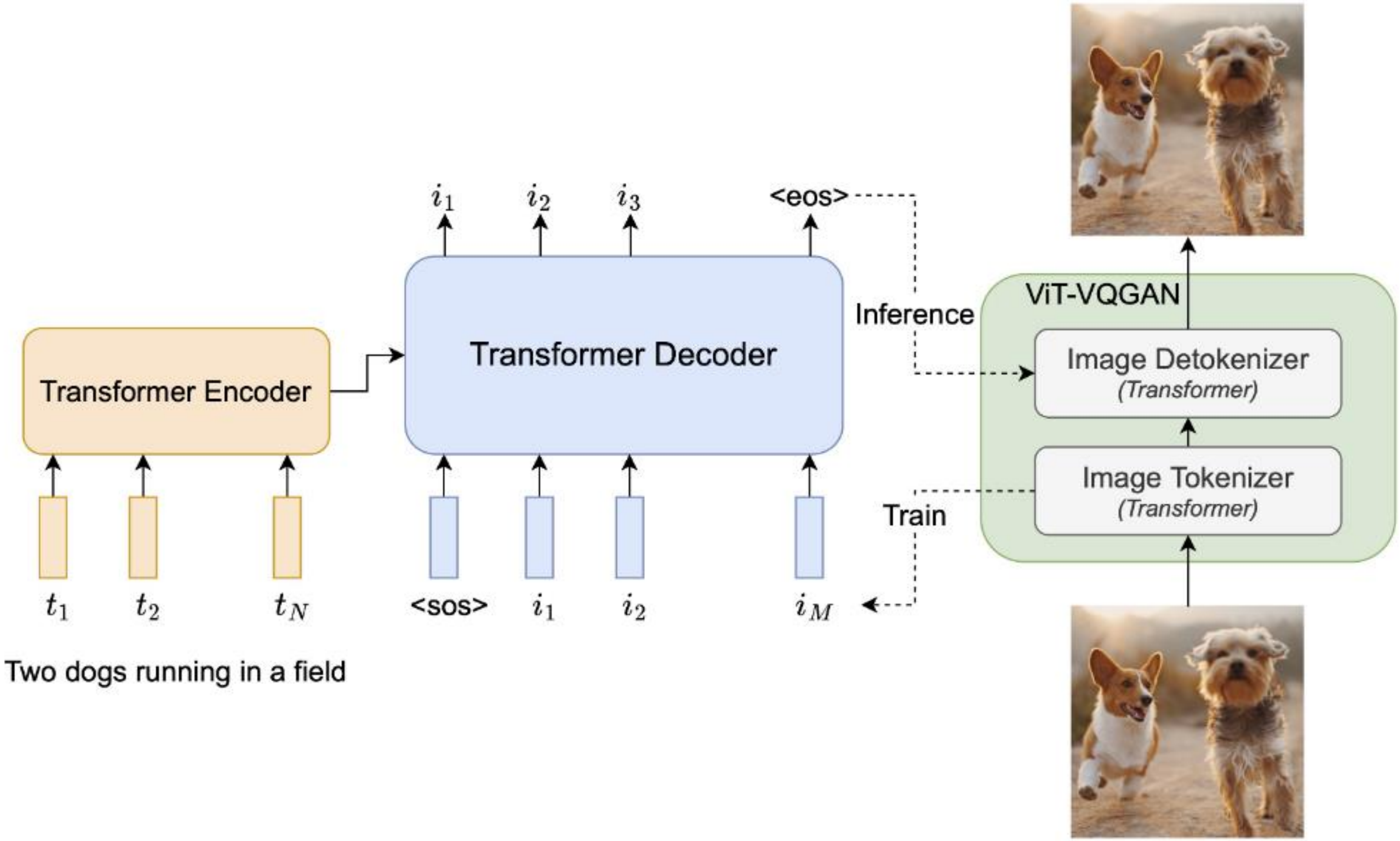
- High-level idea: convert both the image and the text inputs into embedding vectors, then pass those vectors into a decoder-only transformer and do next (text) token prediction



- 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

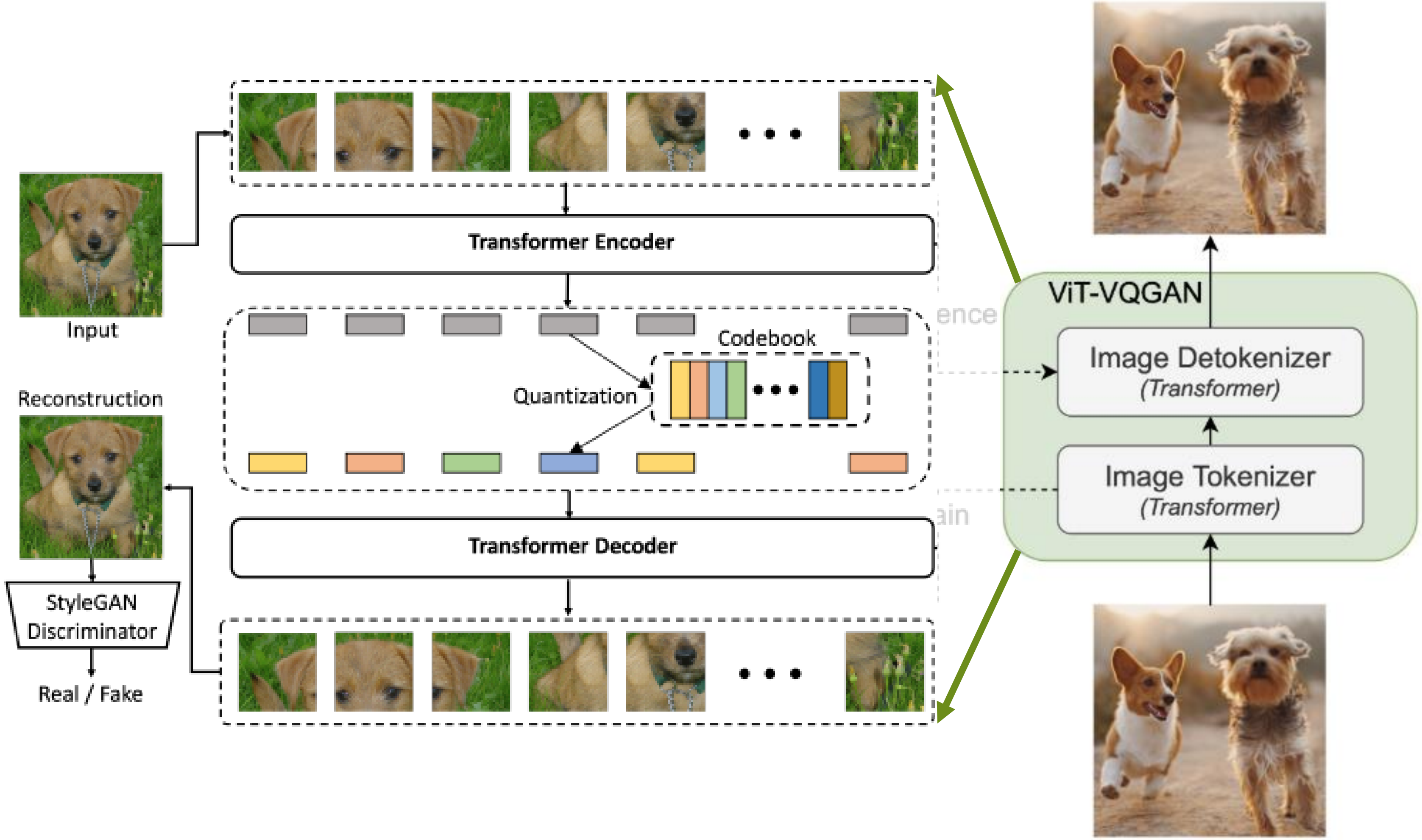
Source: https://huggingface.co/blog/vlms

Recall: Parti



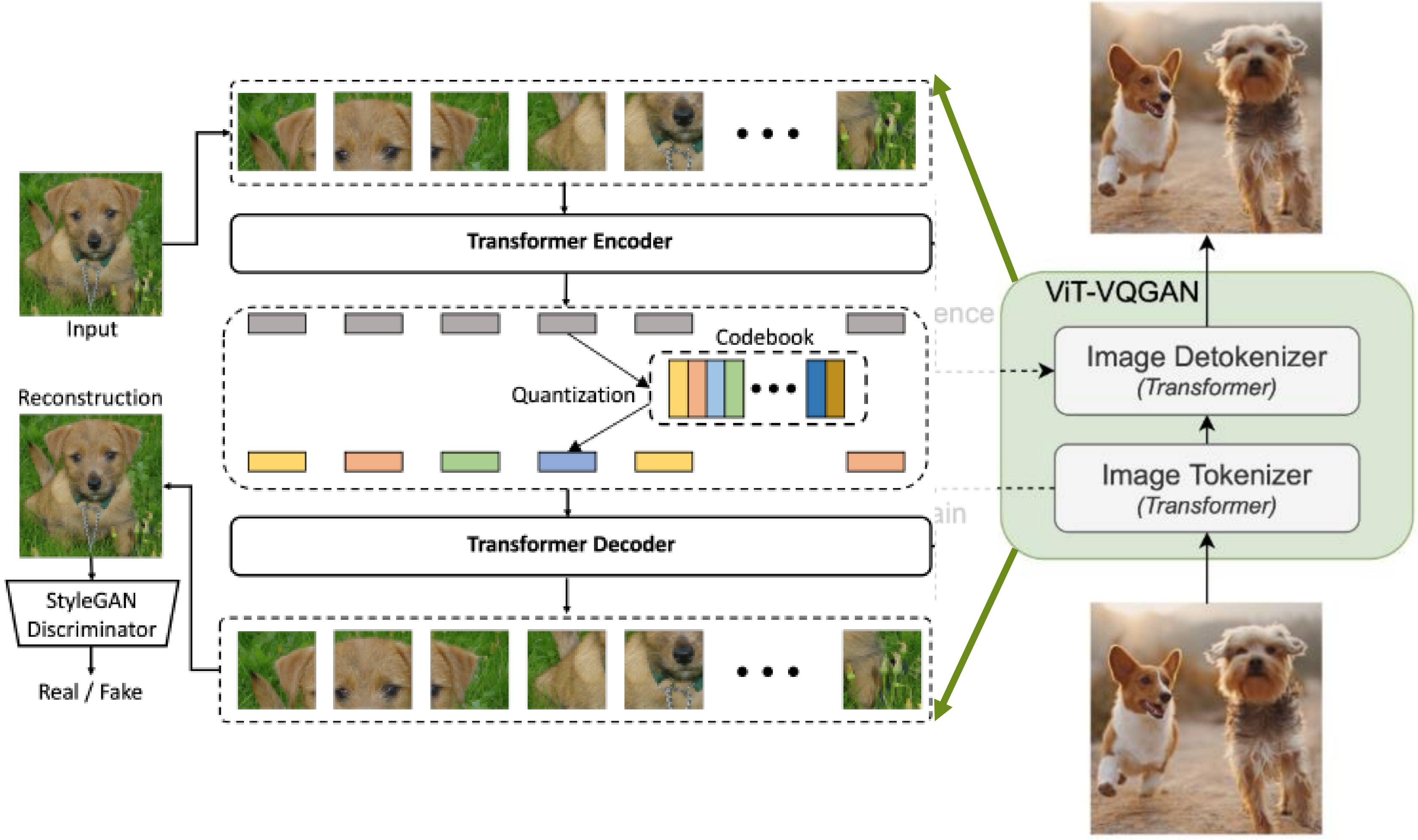
Source: https://arxiv.org/pdf/2206.10789

Recall: Image Tokenization

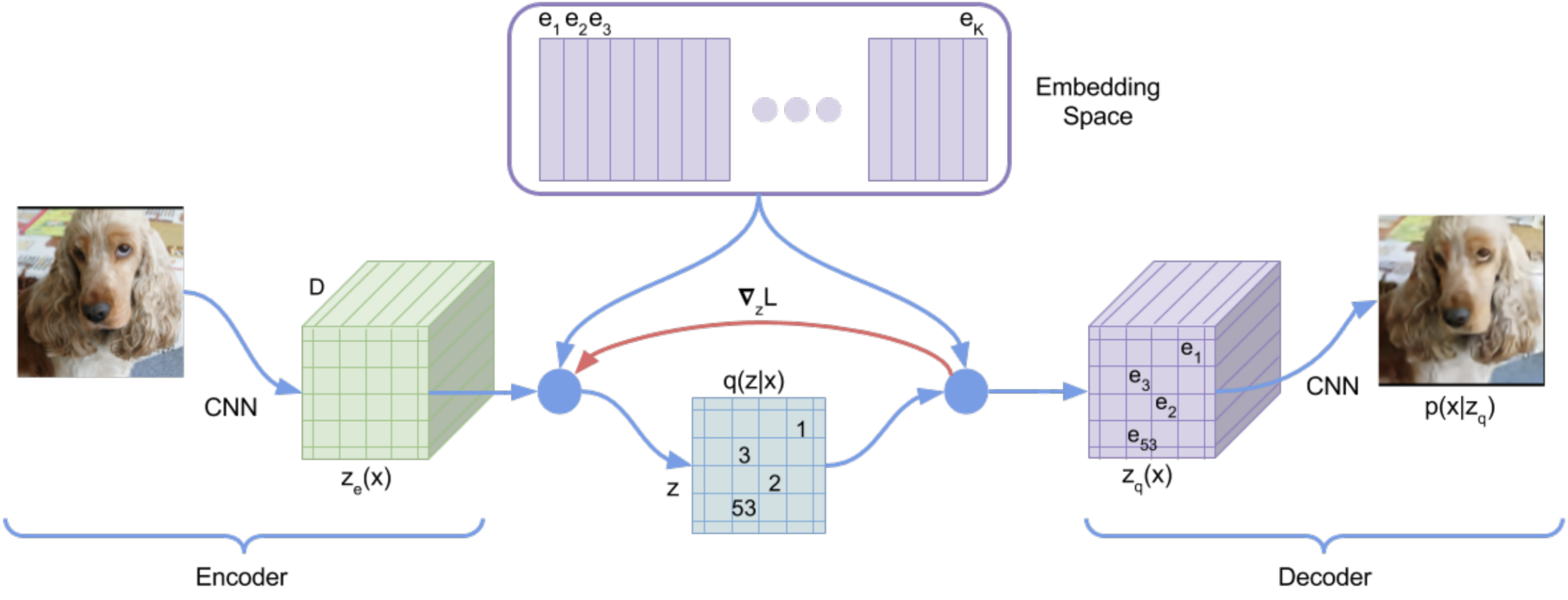


Source: https://arxiv.org/pdf/2110.04627

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

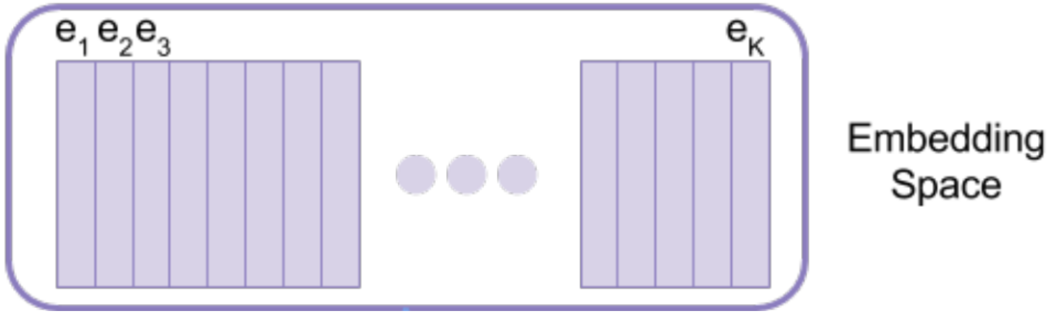


Source: https://arxiv.org/pdf/2110.04627



Vector-Quantized VAEs

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

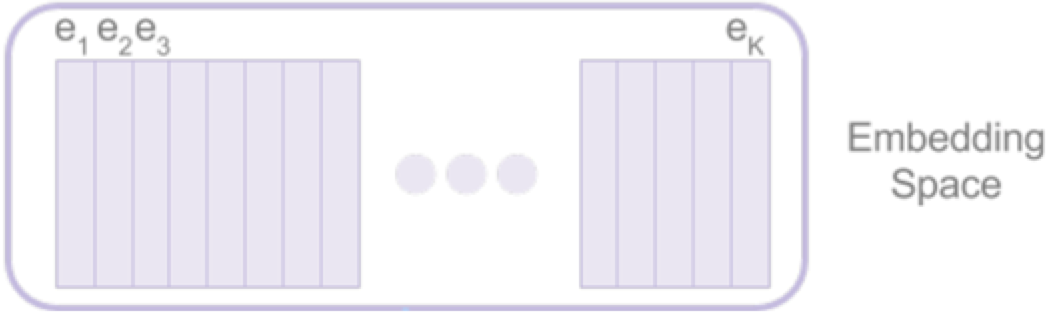


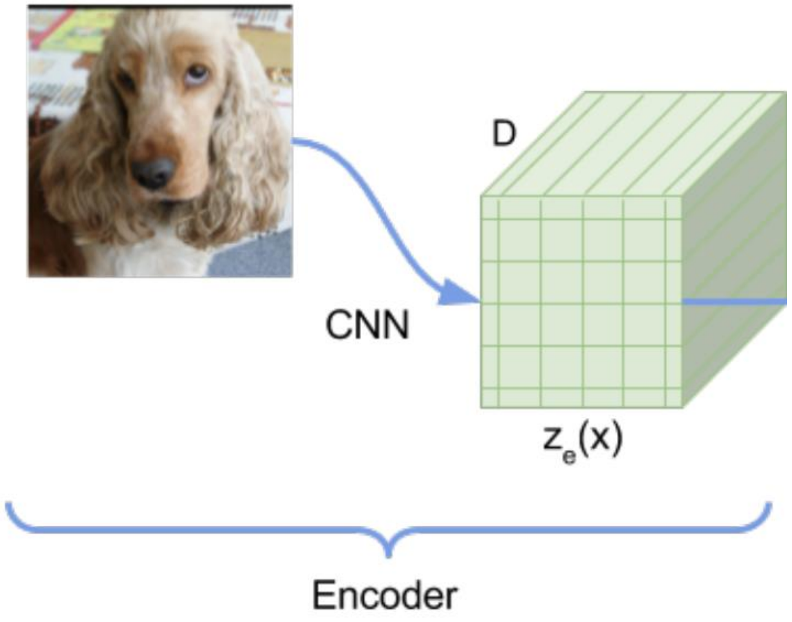
- Embedding space consists of K D-dimensional latent vectors $\{e_1, \dots, e_K\}$ which are learned during training
- The indices $[1, \dots, K]$ of each latent vector correspond to the "image tokens" in some fixed-length codebook

Vector-Quantized VAEs

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

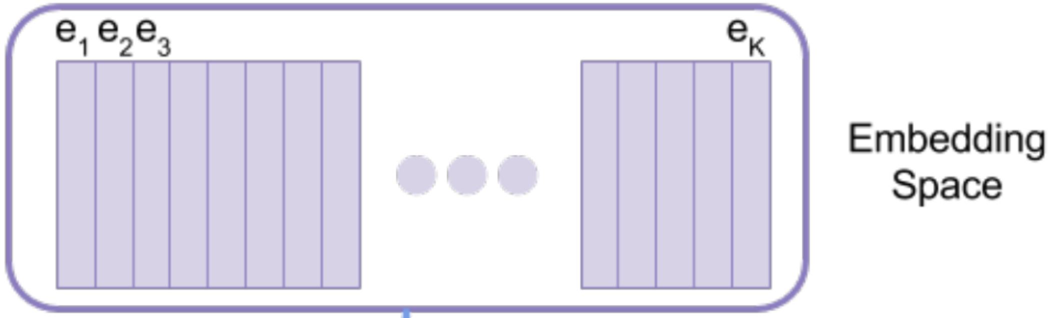
Vector-Quantized VAEs

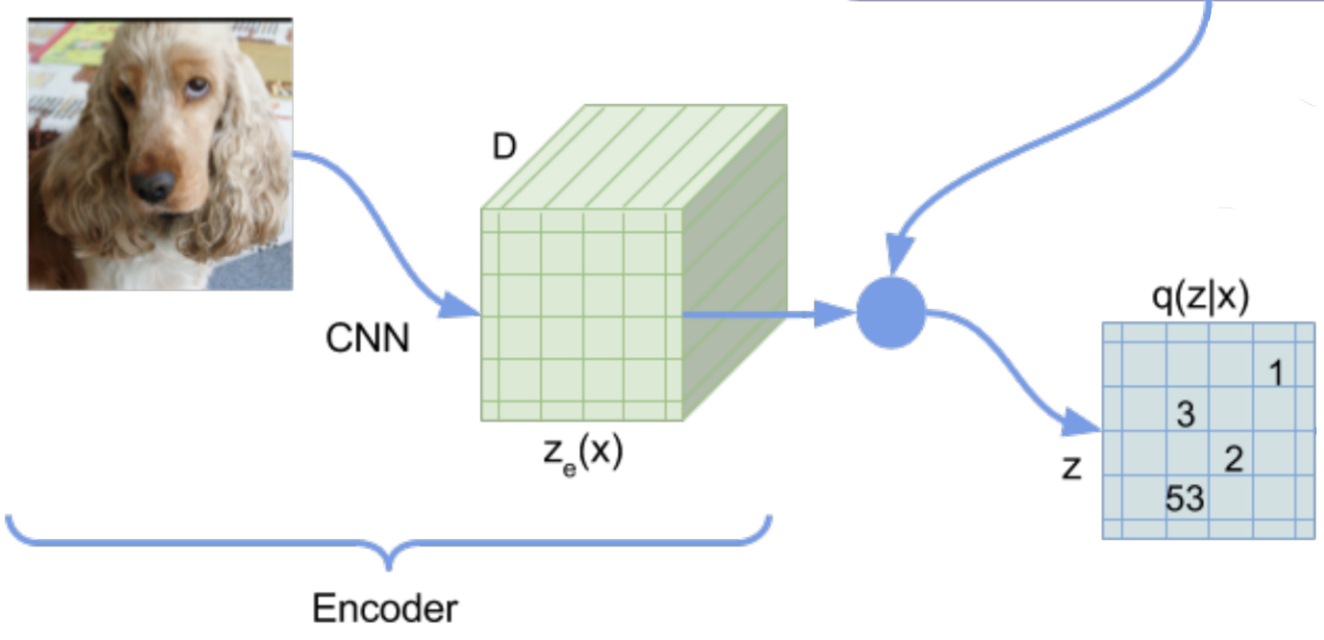




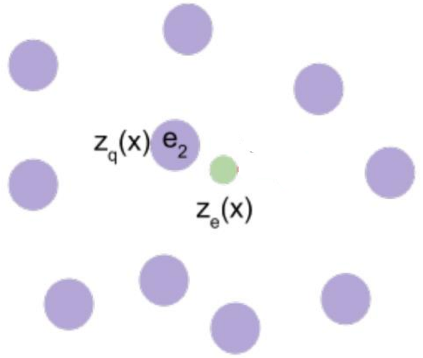
- The encoder (e.g., a ResNet-like CNN) maps images to N D-dimensional vectors

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





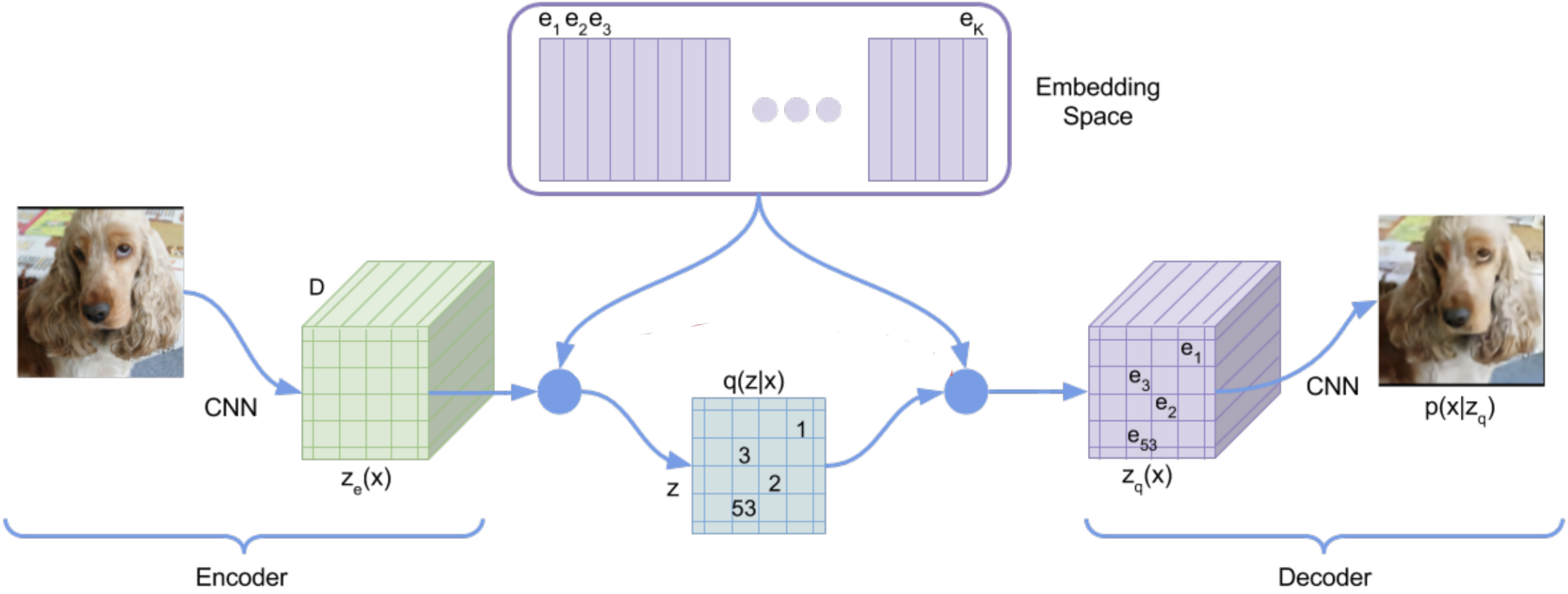
- Each output vector from the encoder is mapped to the nearest latent vector to get the discretized encoding



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

Vector-Quantized VAEs

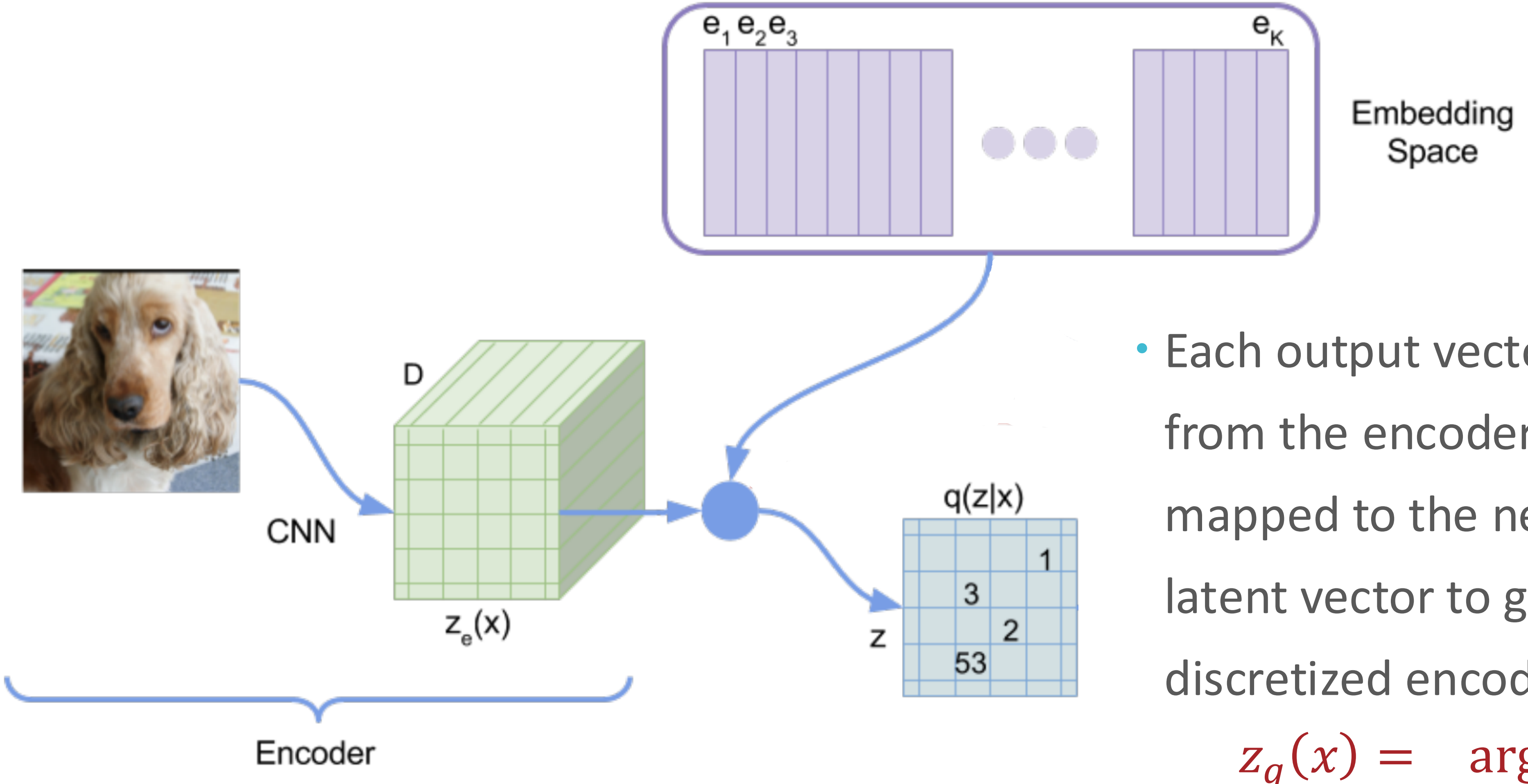
Source: https://arxiv.org/pdf/1711.00937



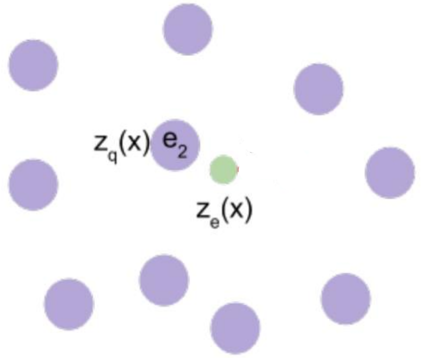
- The decoder takes the discretized representation and recreates the original image

Vector-Quantized VAEs

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



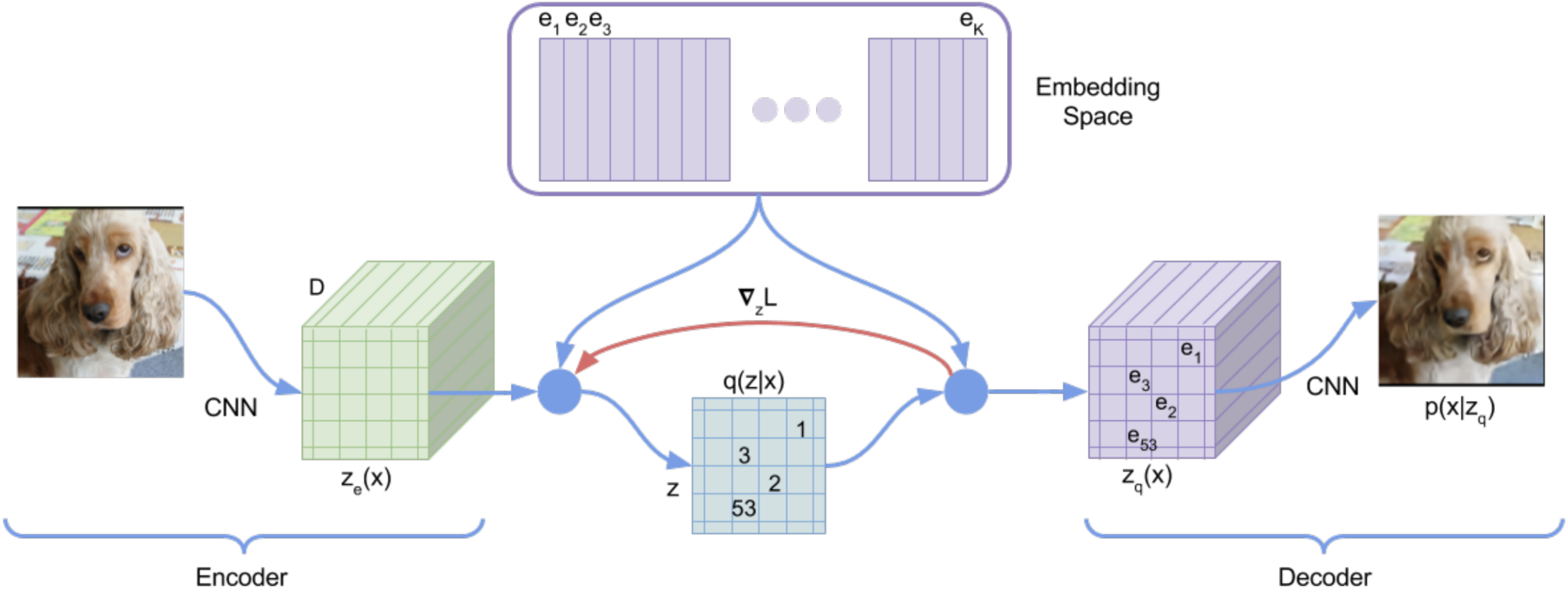
- Each output vector from the encoder is mapped to the nearest latent vector to get the discretized encoding



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

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

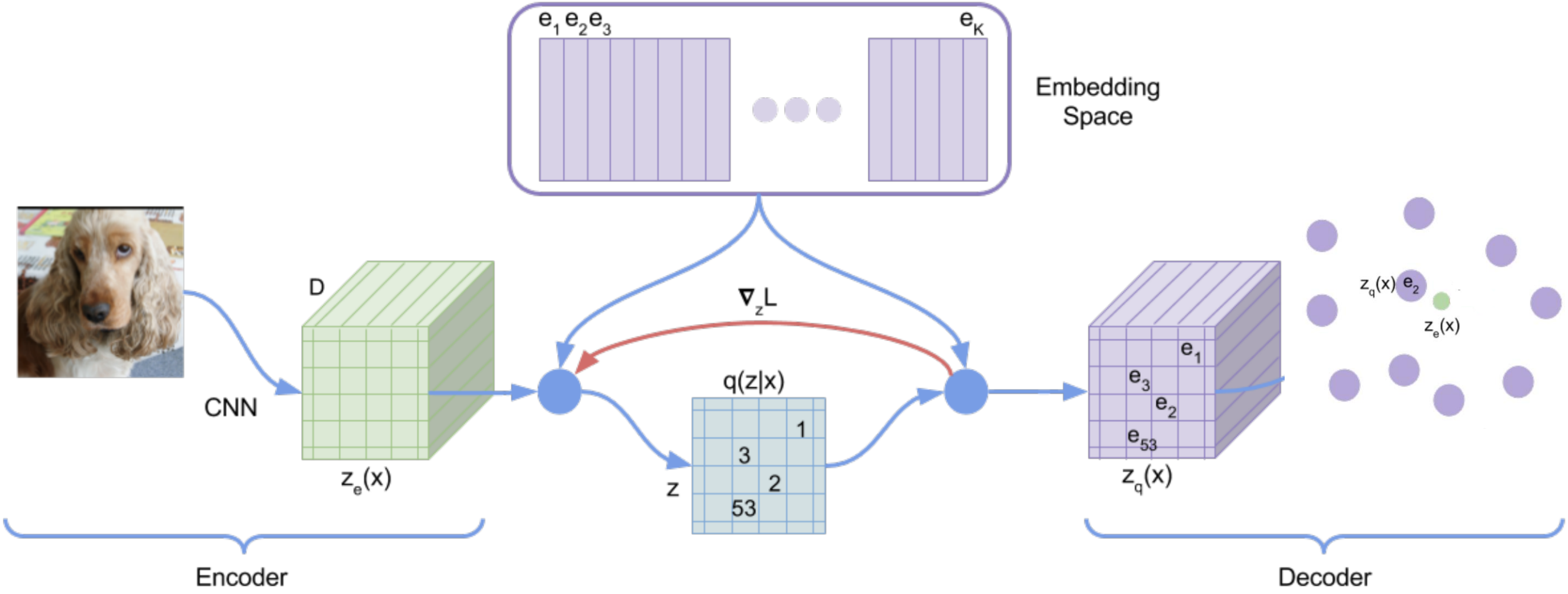
Source: https://arxiv.org/pdf/1711.00937



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

Straight-through Estimator

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

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

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

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

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

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

- The first term is the typical reconstruction error objective

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

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

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

VQ-VAE Objective Function

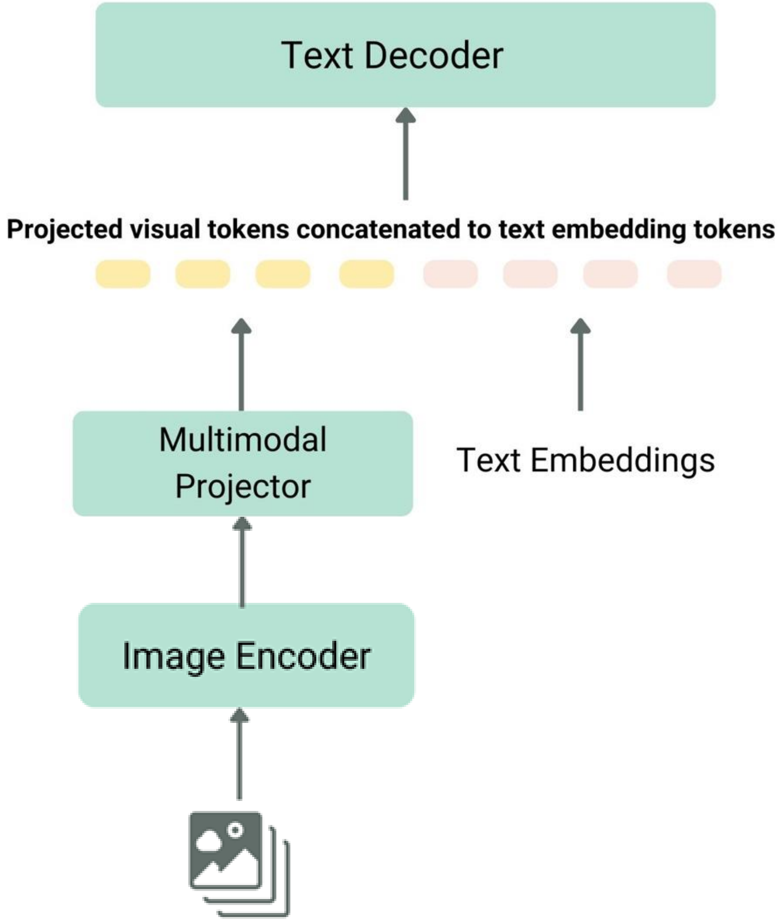
- Intuition: we want the latent vectors to correspond to relevant points in the embedding space i.e., ones that are near the outputs of the encoder
- However, we also want the encoder to respect the latent vectors and not overfit to the training dataset
- Idea: augment the standard VAE objective with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

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

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

VLM: Architecture

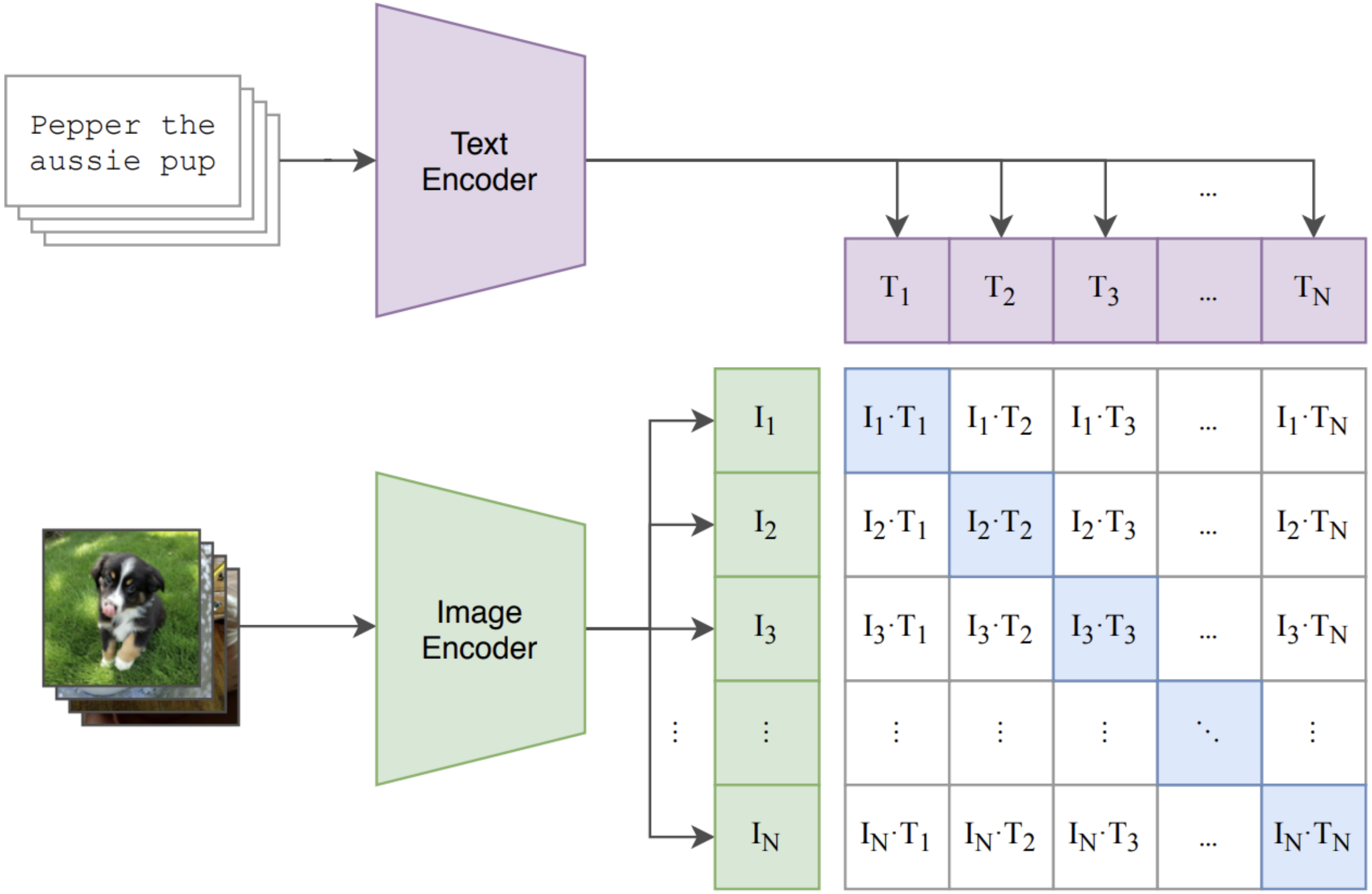
- High-level idea: convert both the image and the text inputs into embedding vectors, then pass those vectors into a decoder-only transformer and do next (text) token prediction



- 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

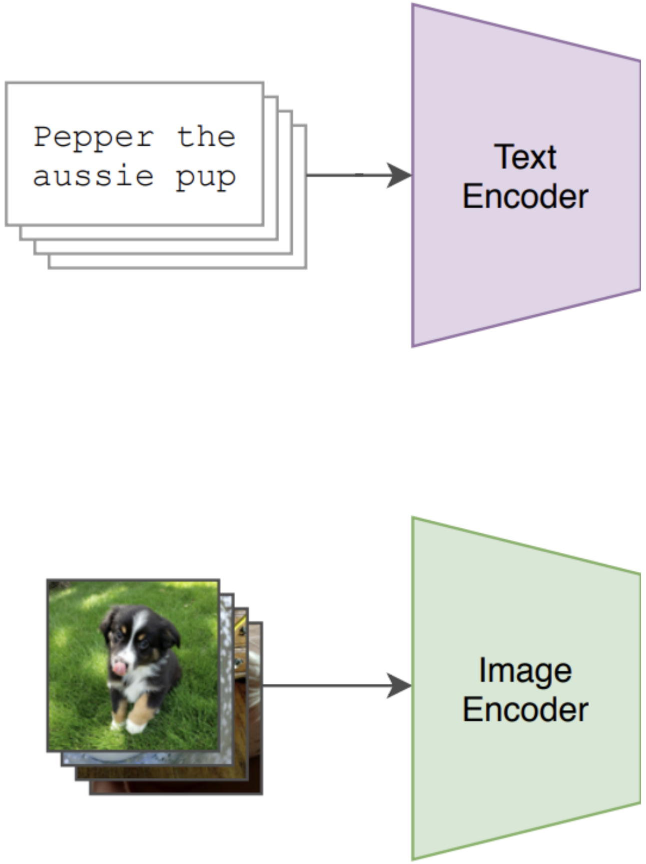
Source: https://huggingface.co/blog/vlms

CLIP



Source: https://arxiv.org/pdf/2103.00020

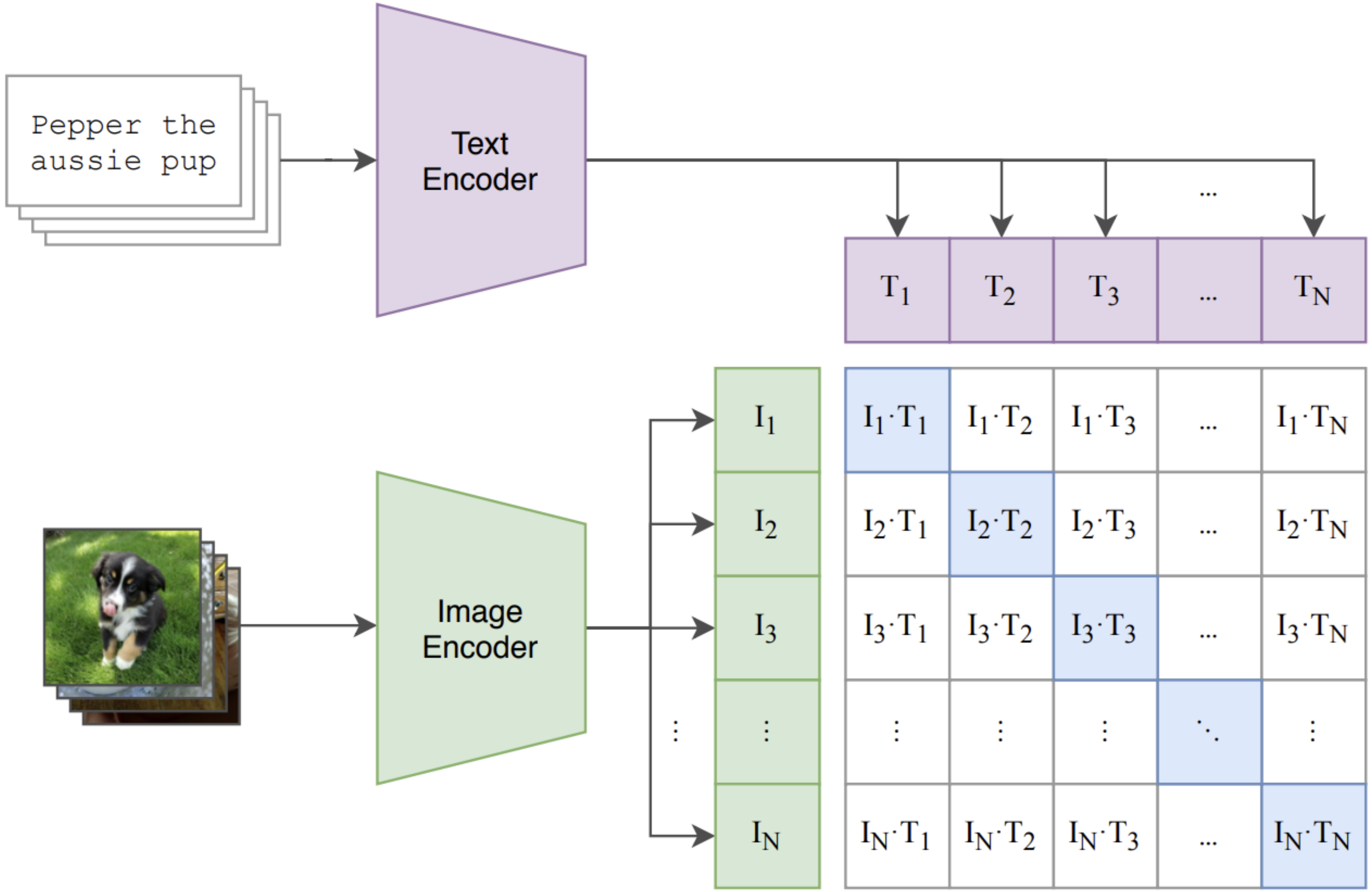
CLIP



- The text encoder (e.g., an encoder-only transformer) and the image encoder (e.g., a ResNet-like CNN or ViT) are both linearly projected into same-dimensional vectors i.e., the multi-modal embedding space

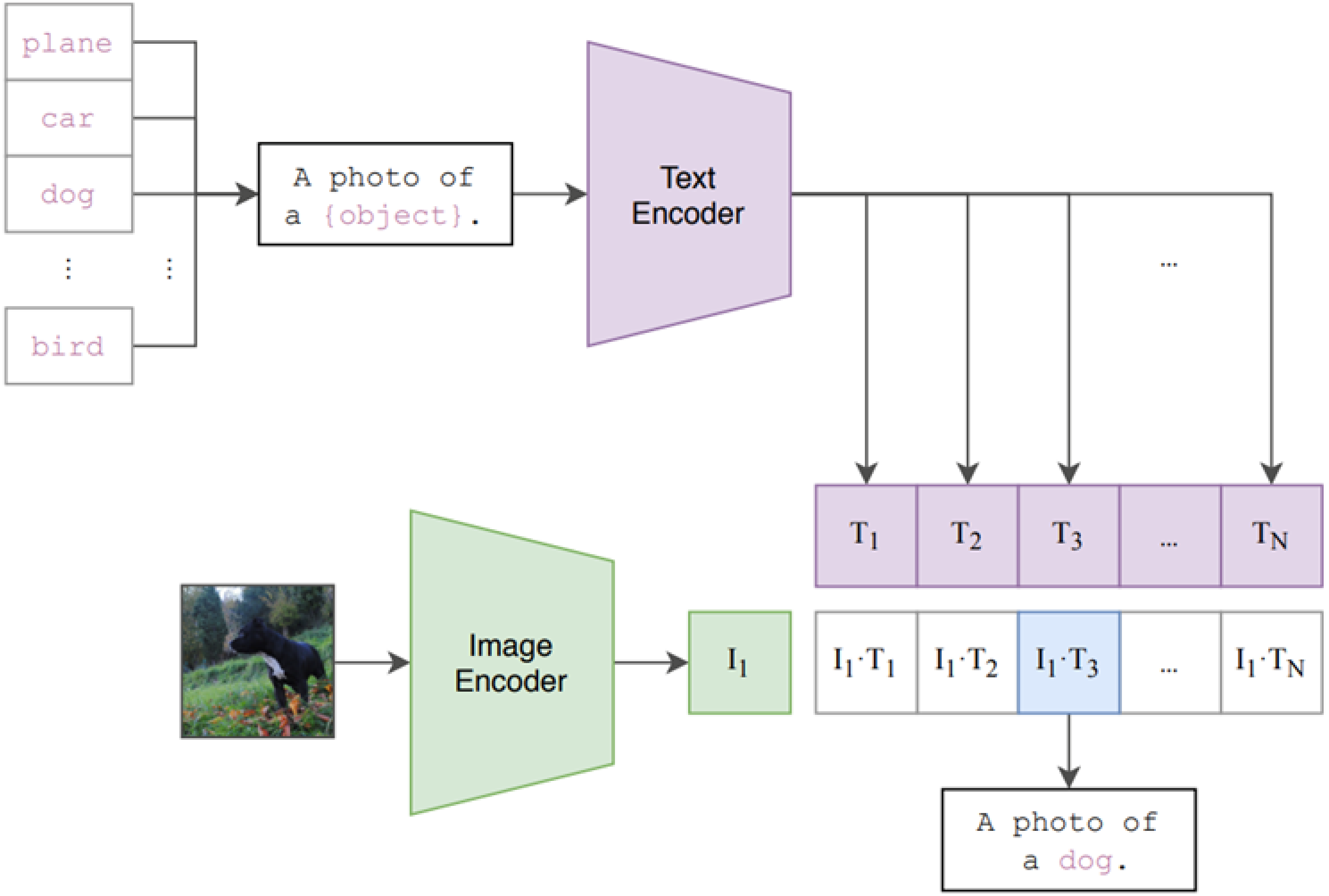
Source: https://arxiv.org/pdf/2103.00020

CLIP



- 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

CLIP: Zero-shot classification



Source: https://arxiv.org/pdf/2103.00020

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