

Deep Generative Models: Transformers for Vision and Language

Fall Semester 2024

René Vidal

Director of the Center for Innovation in Data Engineering and Science (IDEAS)

Rachleff University Professor, University of Pennsylvania

Amazon Scholar & Chief Scientist at NORCE



Evolutions of Transformers

Natural Language Processing:

- **BERT (Bidirectional Encoder Representations from Transformers)**
- GPT (Generative Pre-trained Transformer)

Computer Vision:

- **Vision Transformer**
- Swin Transformer, Pyramid Vision Transformer

Unifying Vision and Language

- **ViLBERT: Vision and Language BERT**
- **CLIP: Contrastive Language-Image Pre-training**
- **DALL-E: Zero-Shot Text-to-Image Generation**
- **LLaVA: Large Language and Vision Assistant**

Transformer for Vision and Language

- Transformers thus far focus on either NLP or vision. How do we build a model for unified vision and language tasks using transformers?

Captioning:
a cat staring out the window at a group of birds.

FOIL:
a **dog** **cat** staring out the window at a group of birds.

Referring Expression Recognition:
Bird to the left of the feeder.

Visual Question Answering:
Q: How many birds are there?
A: four



A blue arrow points from the text "Bird to the left of the feeder." to the birdfeeder in the image.

NLVR:
Q: Left image has twice as many cats as the right image, and at least two cats are black. **A:** True

Visual Dialog:

A-Bot: Image shows a cat staring out the window at a group of birds.

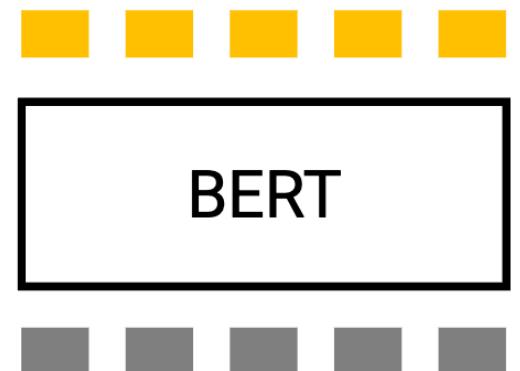
Q-Bot: How many cats are there ?
A-Bot: 1

Q-Bot: Can you see its face? [it = cat; visual coreference]
A-Bot: no

Q-Bot: I think we were talking about **Image 2**.

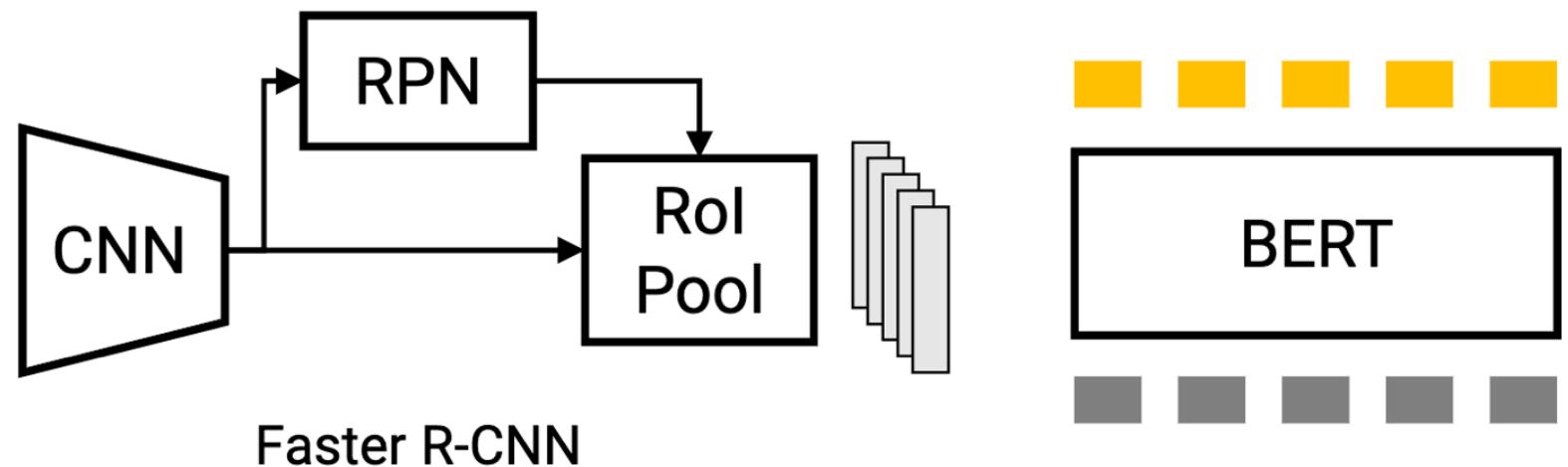
ViLBERT: Vision and Language BERT

- ViLBERT is a transformer-based model that extends the BERT architecture to process both visual and language inputs, marking one of the first successful applications of transformers for multimodal tasks.
 - Goal: To enable joint understanding of images and text, supporting tasks like image captioning, visual question answering (VQA), and cross-modal retrieval.
- Uses two parallel transformer streams—one for images and one for text.
 - Start with a pretrained BERT model and finetune it.



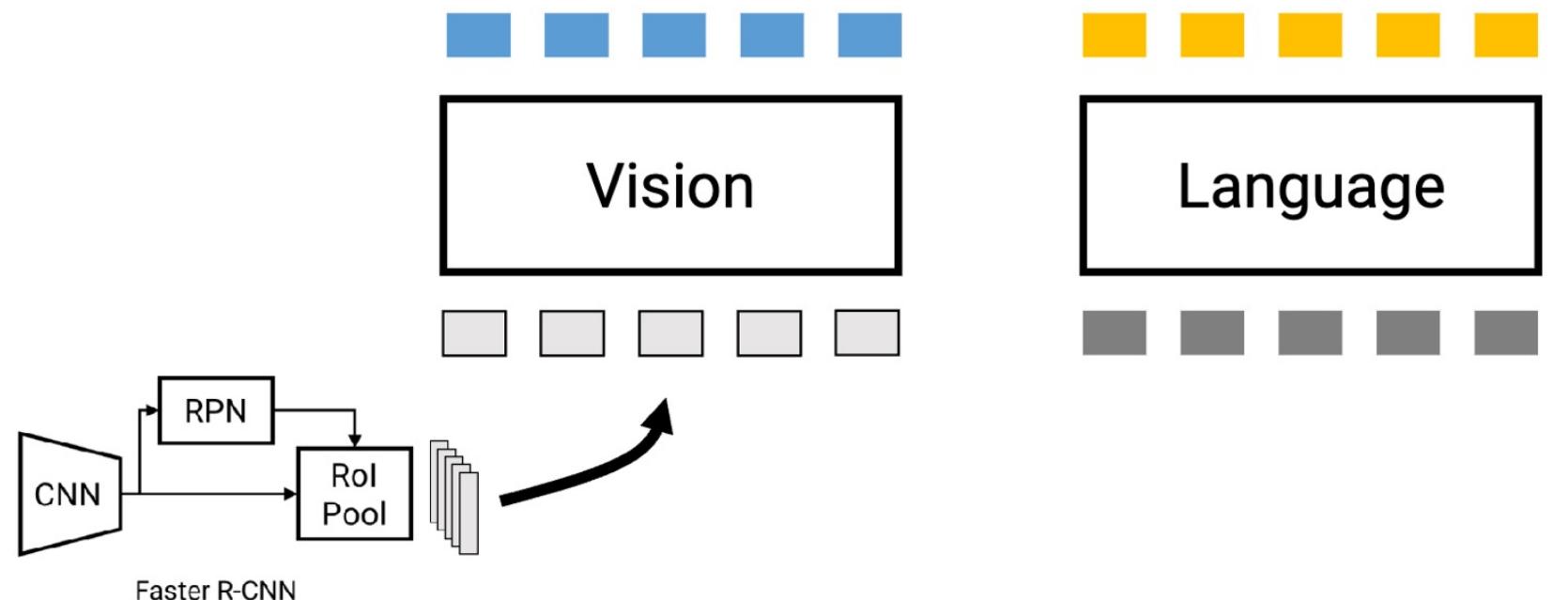
ViLBERT: Vision and Language BERT

- Uses two parallel transformer streams—one for images and one for text.
 - Start with a pretrained BERT model and finetune it.
 - Extract regions from an image using pre-trained detector.



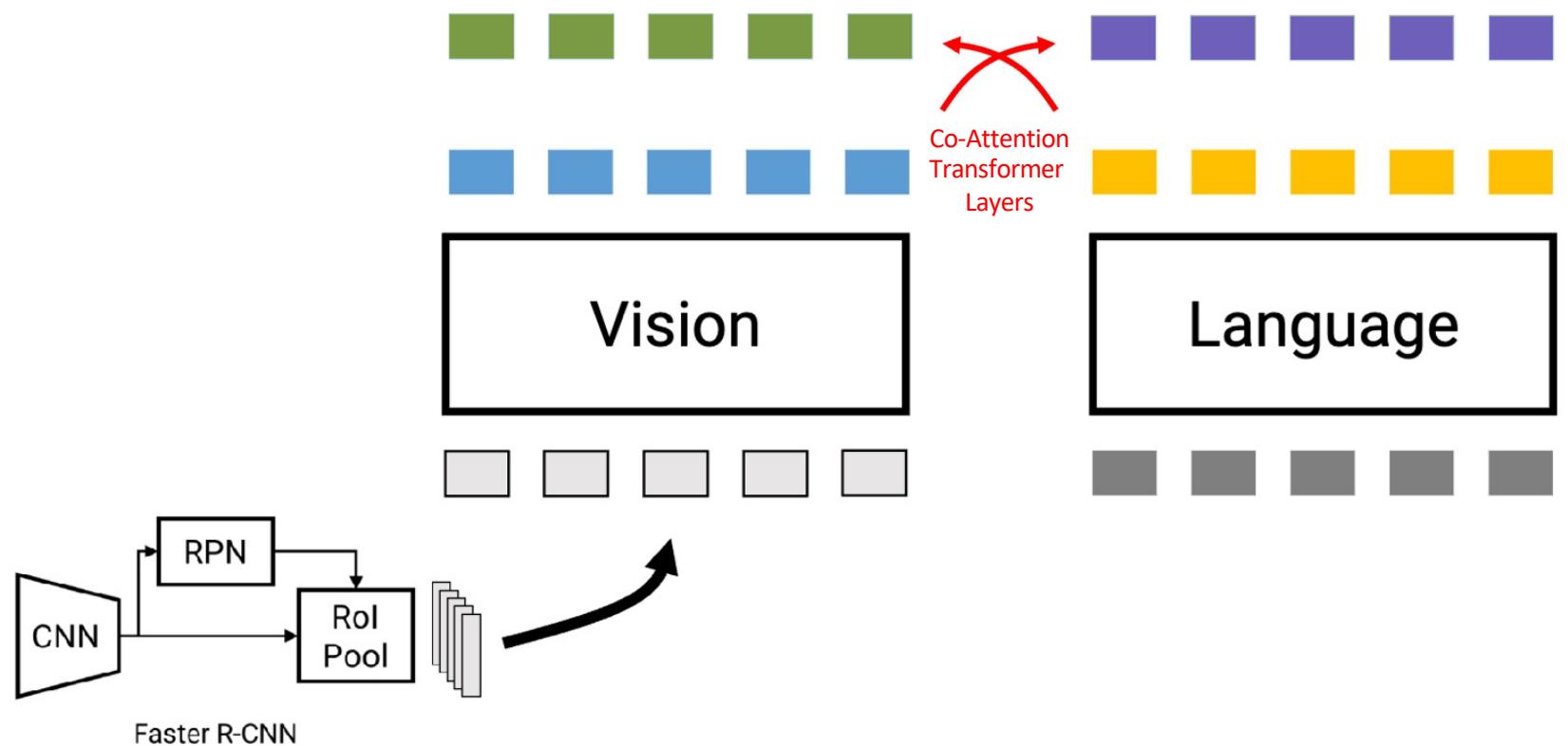
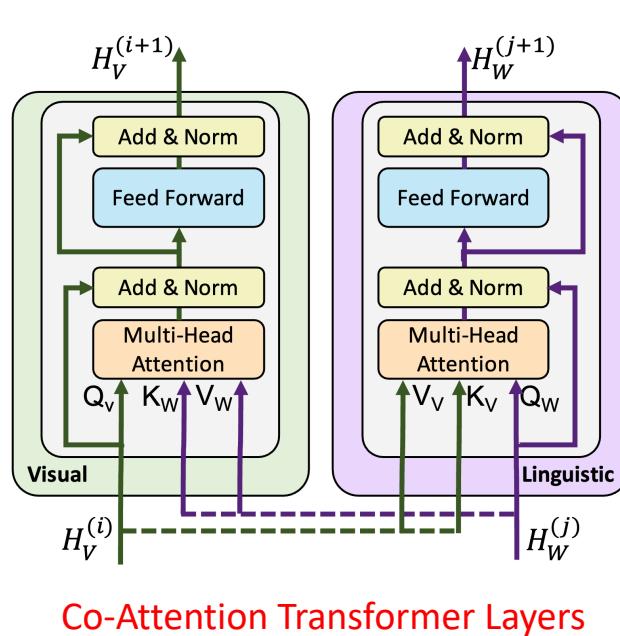
ViLBERT: Vision and Language BERT

- Uses two parallel transformer streams—one for images and one for text.
 - Start with a pretrained BERT model and finetune it.
 - Extract regions from an image using pre-trained detector.
 - Use another BERT-like model to process the visual tokens.
 - For each detected object, the visual token is defined as the mean-pooled convolutional feature from that region.
 - It introduces spatial positional encoding, constructing a 5-d vector from detected region position (top-left and bottom-right coordinates) and the fraction of image area covered. This is then linearly projected to match dimension of visual tokens.



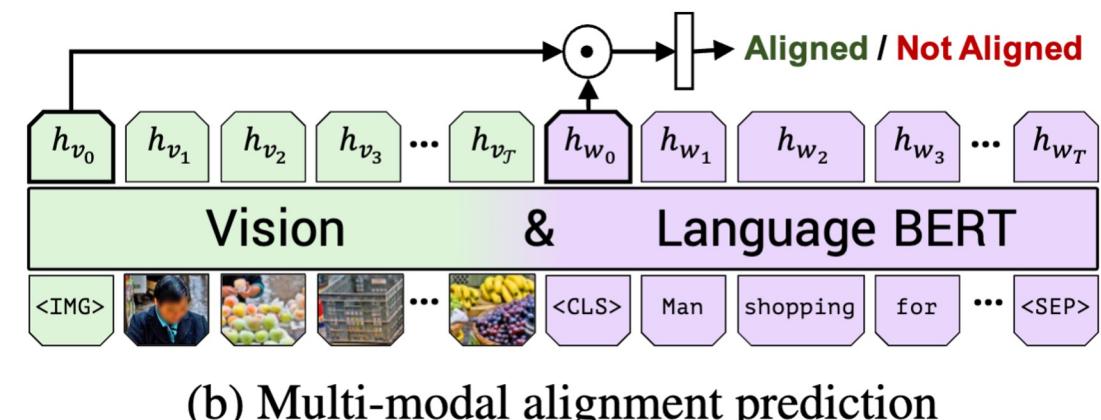
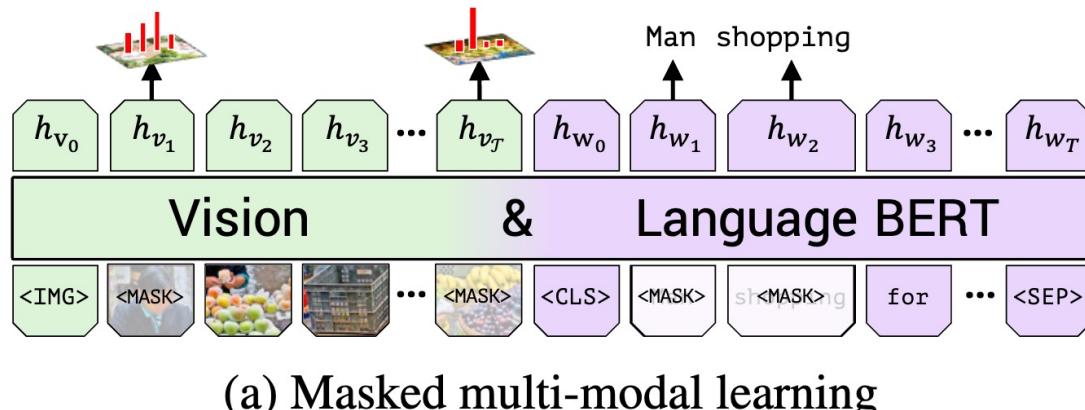
ViLBERT: Vision and Language BERT

- Uses two parallel transformer streams—one for images and one for text.
 - Connect the vision and language processing by a co-attention mechanism.
 - By exchanging keys and values between streams in multi-headed attention, each stream integrates features from the other, enhancing visual understanding with language features and vice versa – in effect performing image-conditioned language attention in the visual stream and language-conditioned image attention in the linguistic stream.



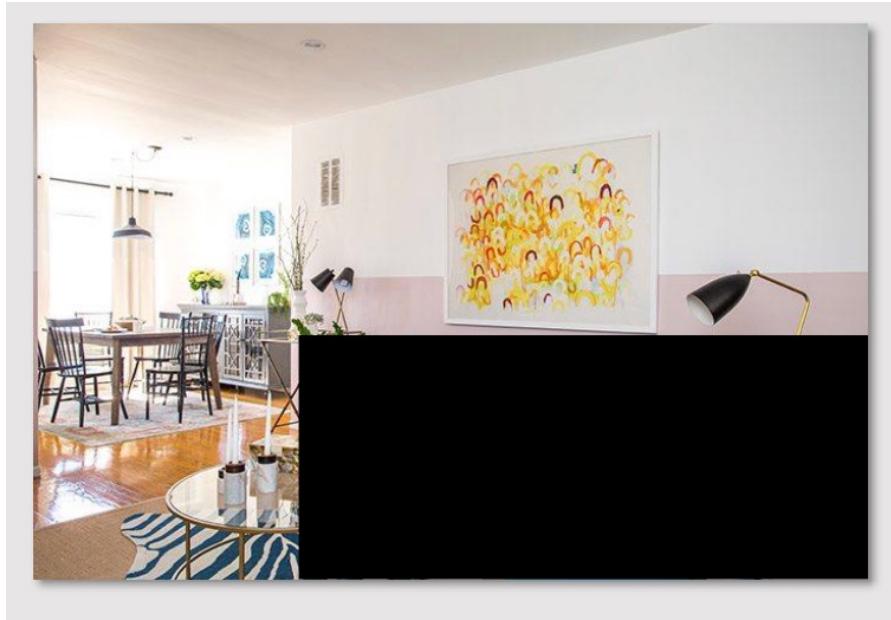
ViLBERT Pretraining and Transfer Tasks

- ViLBERT is trained on the Conceptual Captions dataset under two training tasks:
 - Masked Multi-Modal Learning: It follows the MLM task in standard BERT. It learns to reconstruct image region categories or words for masked inputs given the observed inputs.
 - Multi-Modal Alignment Prediction: It learns to predict whether or not the caption describes the image content.
- After pretraining, ViLBERT is adapted to various established vision-and-language tasks through minimal modifications.
 - Fine-tuning generally involves adding a task-specific classification layer, enabling end-to-end training without major architecture changes.

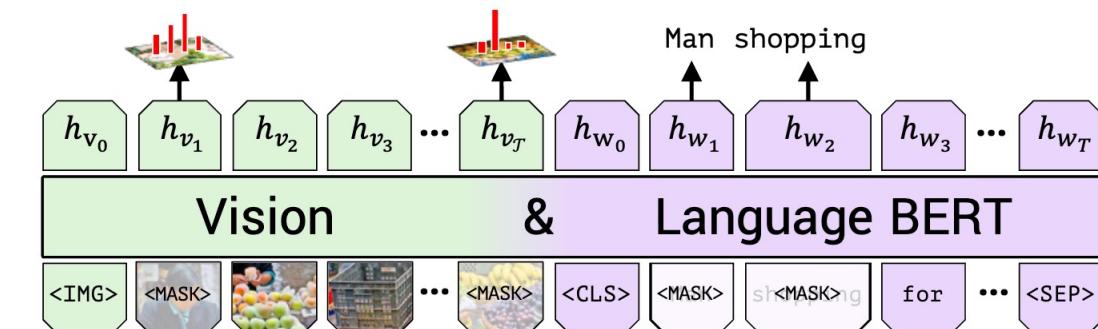


ViLBERT Pretraining

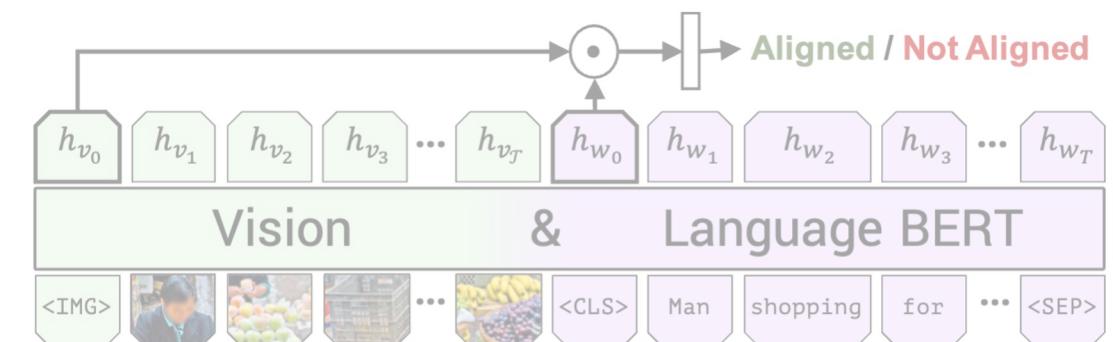
- **Masked Multi-Modal Learning:** It learns to reconstruct image region categories or words for masked inputs given the observed inputs.
 - For visual tokens, instead of directly regressing the masked feature values, it predicts a distribution over semantic classes for the corresponding image region.
 - To supervise this, the model minimizes the KL divergence between its predicted distribution and the output distribution from Faster R-CNN for the same region.



blue sofa in the living
room.



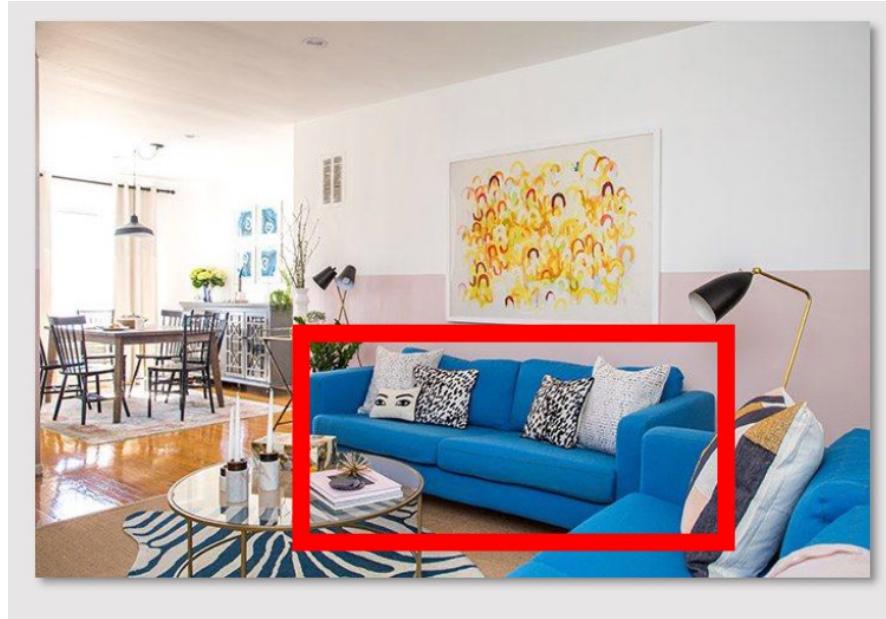
(a) Masked multi-modal learning



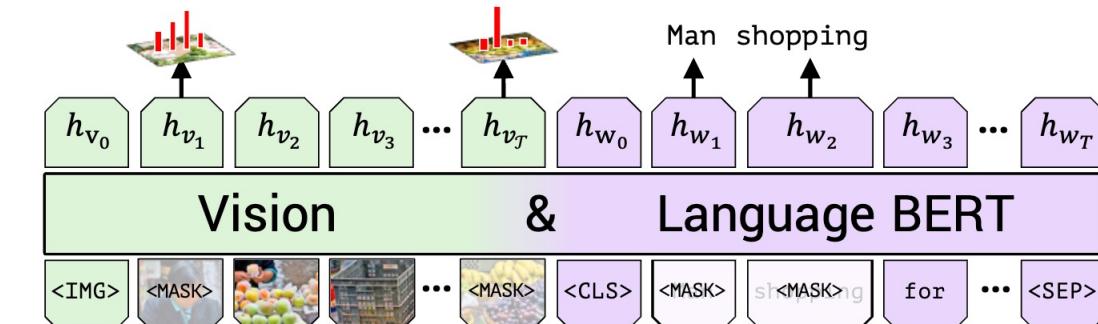
(b) Multi-modal alignment prediction

ViLBERT Pretraining

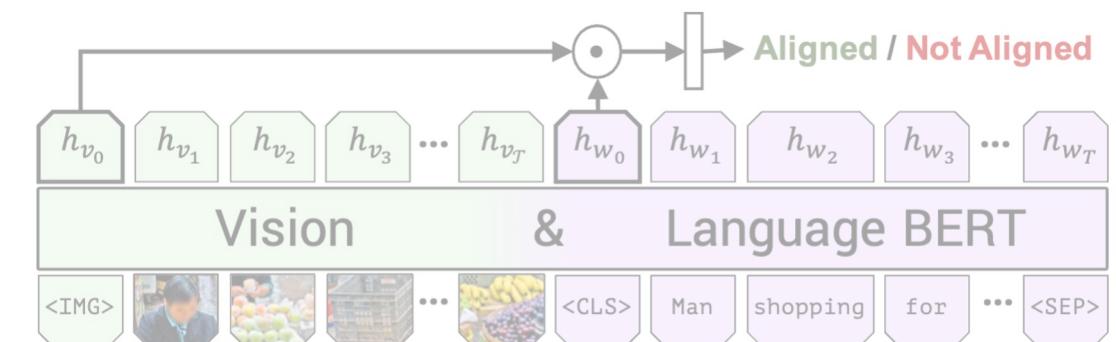
- **Masked Multi-Modal Learning:** It learns to reconstruct image region categories or words for masked inputs given the observed inputs.
 - For visual tokens, instead of directly regressing the masked feature values, it predicts a distribution over semantic classes for the corresponding image region.
 - To supervise this, the model minimizes the KL divergence between its predicted distribution and the output distribution from Faster R-CNN for the same region.



██████████ in the living room.



(a) Masked multi-modal learning



(b) Multi-modal alignment prediction

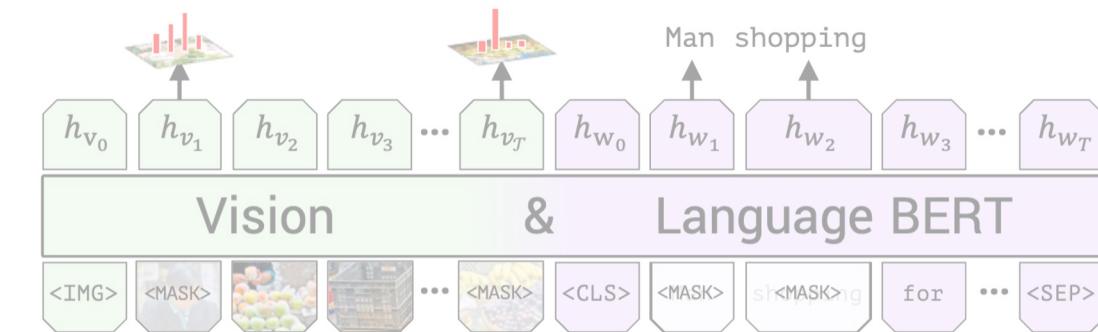
ViLBERT Pretraining

- **Multi-Modal Alignment Prediction:** It learns to predict whether or not the caption describes the image content.
 - The model predicts whether an image-text pair is aligned by learning holistic visual (h_{v_0}) and linguistic (h_{w_0}) representations, similar to the [CLS] token in BERT and ViT.
 - It uses an element-wise product of h_{v_0} and h_{w_0} , followed by a binary classifier to determine alignment.

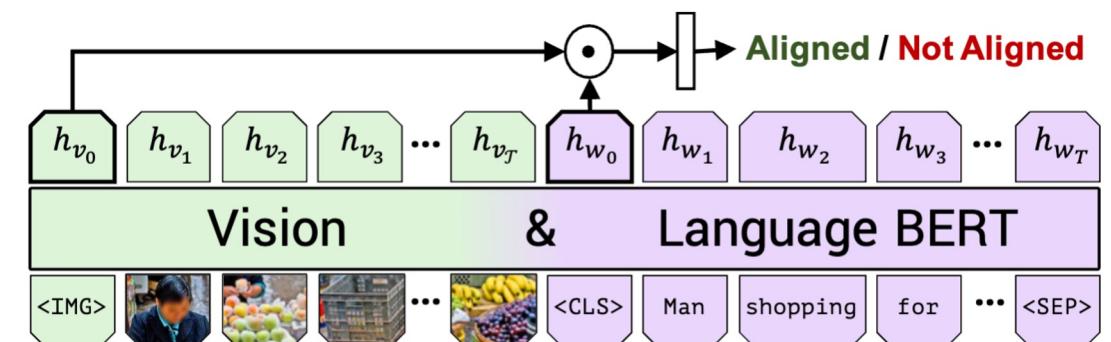


pop artist performs at the festival in a city.

✓ Aligned



(a) Masked multi-modal learning



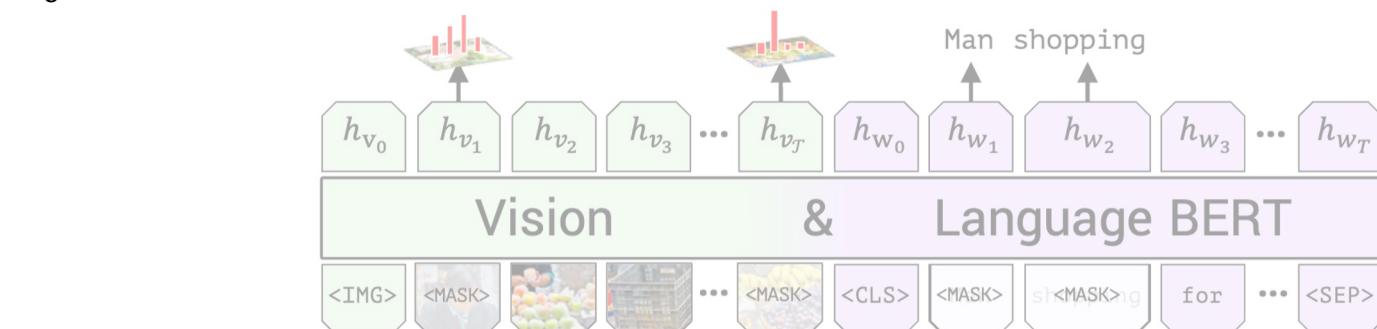
(b) Multi-modal alignment prediction

ViLBERT Pretraining

- **Multi-Modal Alignment Prediction:** It learns to predict whether or not the caption describes the image content.
 - The model predicts whether an image-text pair is aligned by learning holistic visual (h_{v_0}) and linguistic (h_{w_0}) representations, similar to the [CLS] token in BERT and ViT.
 - It uses an element-wise product of h_{v_0} and h_{w_0} , followed by a binary classifier to determine alignment.

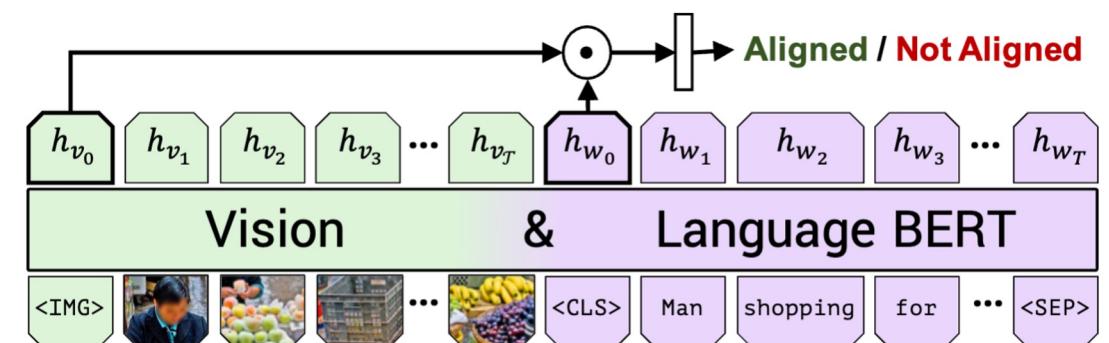


a man kicks a ball



✗ Not Aligned

(a) Masked multi-modal learning



(b) Multi-modal alignment prediction

ViLBERT Transfer Tasks

- A pretrained ViLBERT is transferred to a set of vision-and-language tasks: such as **Visual Question Answering (VQA)**, **Visual Commonsense Reasoning (VCR)**, **Grounding Referring Expressions**, and **Caption-Based Image Retrieval**.

Large-scale Web Data
(Conceptual Captions)



Blue sofa in the living room.

Embodied Visual Navigation
(Room-to-Room)



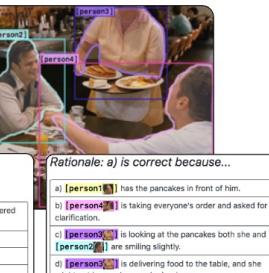
Walk through the bedroom and out of the door into the hallway. Walk down the hall along the banister rail through the open door. Continue into the bedroom with a round mirror on the wall and butterfly sculpture.



VQA



VCR Q→A



VCR QA→R



Referring Expressions



Caption-Based Image Retrieval

ViLBERT: Summary

- **ViLBERT is a multi-modal model that extends BERT to jointly process visual and linguistic information, solving tasks that require understanding both modals.**
- **Architecture:**
 - Dual Stream: Uses two parallel BERT streams – one for visual inputs and one for language inputs – connected via Co-Attention Transformer Layers.
 - Co-Attention Mechanism: Enables each stream to attend to the other by exchanging key-value pairs, allowing image-conditioned language features and vice versa.
- **Pretraining Tasks:**
 - Masked Multi-Modal Learning: Masks out words or visual regions, training the model to predict masked words of semantic class of visual regions given the observed tokens.
 - Multi-Modal Alignment Prediction: Determines if a given image-caption pair is aligned.
- **Applications:**
 - ViLBERT achieves strong performance on vision-and-language tasks like Visual Question Answering (VQA), Visual Commonsense Reasoning, and Image-Text Retrieval.

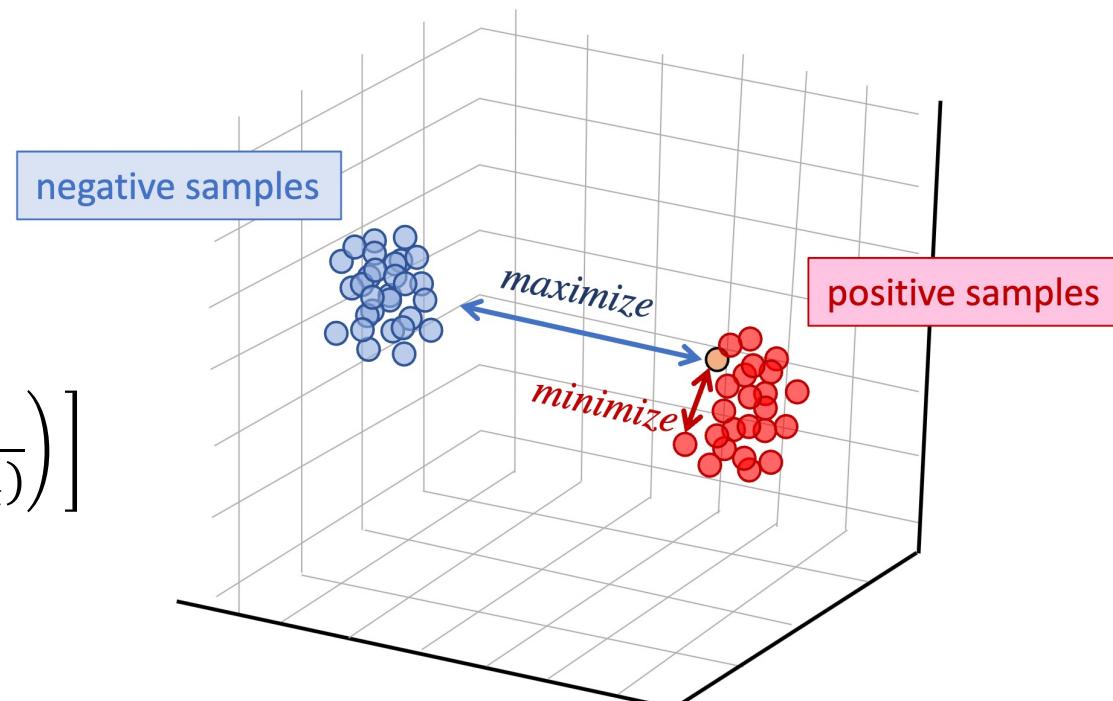
CLIP: Contrastive Language-Image Pre-training

- CLIP (Contrastive Language-Image Pre-training) is a vision and language model that learns to understand images and texts together using natural language descriptions to supervise image classification.
 - CLIP learns to embed both text and image into a shared representation space.
- Generalized Understanding:
 - CLIP learns diverse visual concepts through natural language that is capable of zero-shot learning.
- Zero-shot learning:
 - CLIP can classify unseen categories without specific training, leveraging contrastive learning.
 - Traditional models need labeled data for each category.
 - CLIP is adaptable to new tasks and data without additional labeling.

Contrastive Learning in CLIP

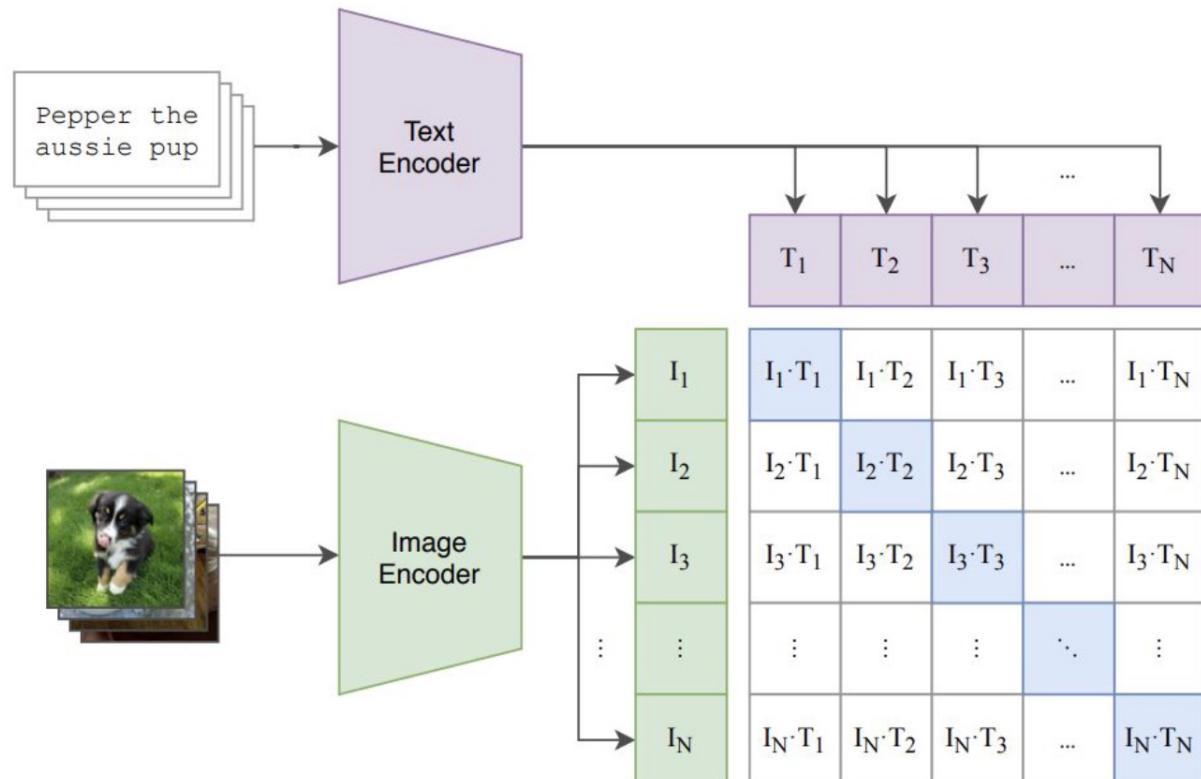
- What is Contrastive Learning?
 - A type of self-supervised learning that focuses on grouping related items closer while pushing unrelated items apart in the learned representation space.
- CLIP utilizes a contrastive learning loss that maximizes similarity between paired image and text embeddings (positive pairs) and minimizes similarity for all other pairs within a training batch (negative pairs).
 - For each sample image I_i ●
 - Minimize its distance with its positive pair T_i ●
 - Maximize its distance with its negative pair T_j ○
 - Goal: $d(f(I_i), f(T_i)) \ll d(f(I_i), f(T_j))$

$$\mathcal{L}_{contrast}(I, T) = \sum_i \left[-\log \left(\frac{\exp(I_i \cdot T_i)}{\sum_j \exp(I_j \cdot T_i)} \right) - \log \left(\frac{\exp(I_i \cdot T_i)}{\sum_k \exp(I_i \cdot T_k)} \right) \right]$$



CLIP Architecture Overview

- Dual-Encoder Structure:
 - CLIP employs two parallel encoders, ViT for images and GPT-2 like Transformer for text, simultaneously processing both visual and textual information.
 - Both encoders project their outputs into a shared multimodal embedding space, effectively pairing them in a common representation space.
- Contrastive Pre-training:
 - During training, the contrastive learning objective encourages the encoders to maximize the similarity between the correct image-text pairs while minimizing it for incorrect pairs.
 - CLIP is trained using very large batch sizes (e.g., 32,768) on 400M image-text pairs to ensure efficient contrastive pairing of image-text pairs and mitigating the effects of false negatives problem.



CLIP Architecture Overview

- Algorithm:

- Minibatch of aligned images I and texts T are passed through an Image Encoder (ViT) and Text Decoder (Transformer), respectively, to produce image features I_f and text features T_f .

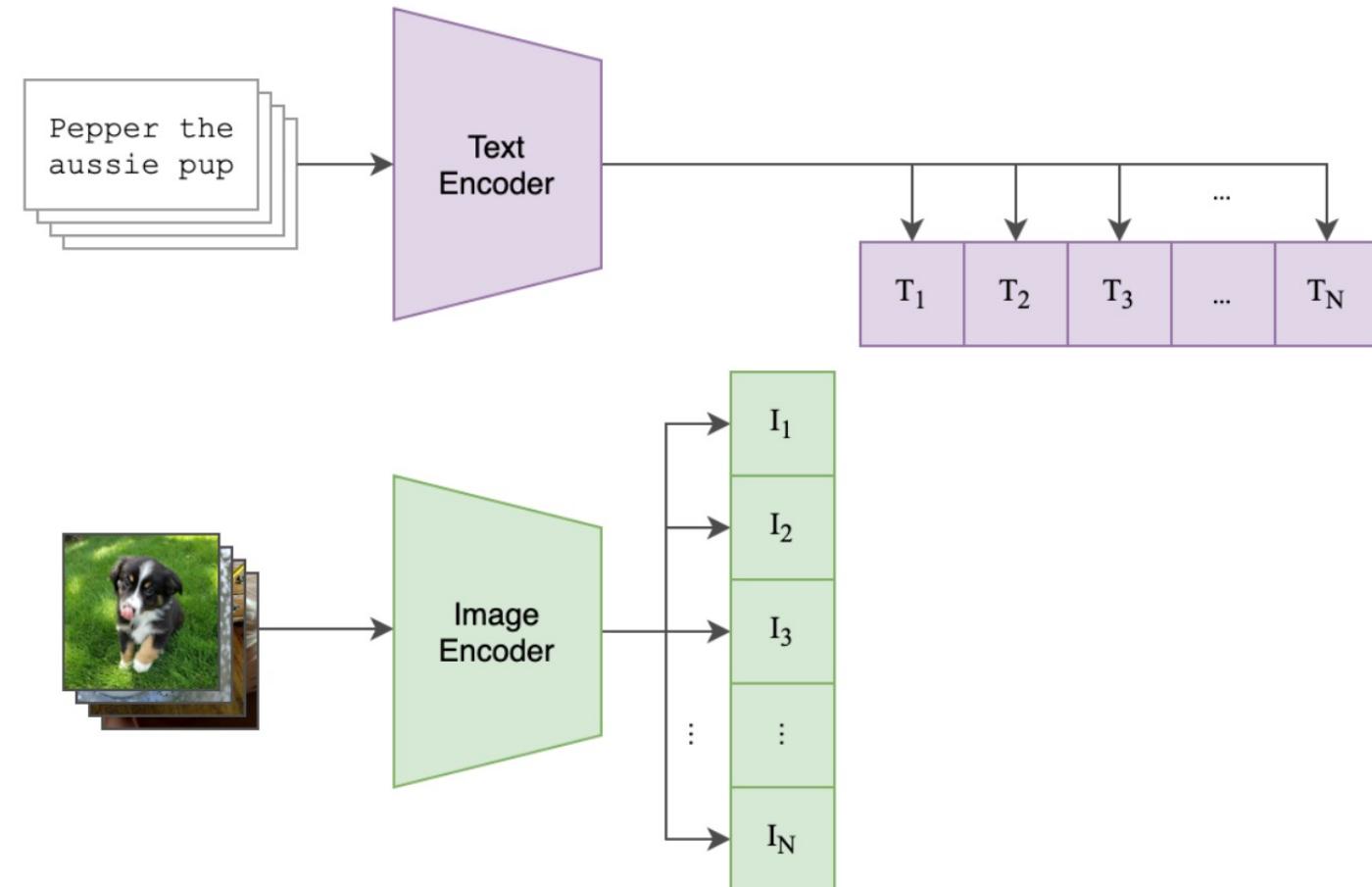
```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```



CLIP Architecture Overview

- Algorithm:

- The representations I_f and T_f are projected into a shared embedding space using learned projection matrices
- These projected embeddings are then L2-normalized and the similarity between each image embedding and each text embedding is calculated using cosine similarity.

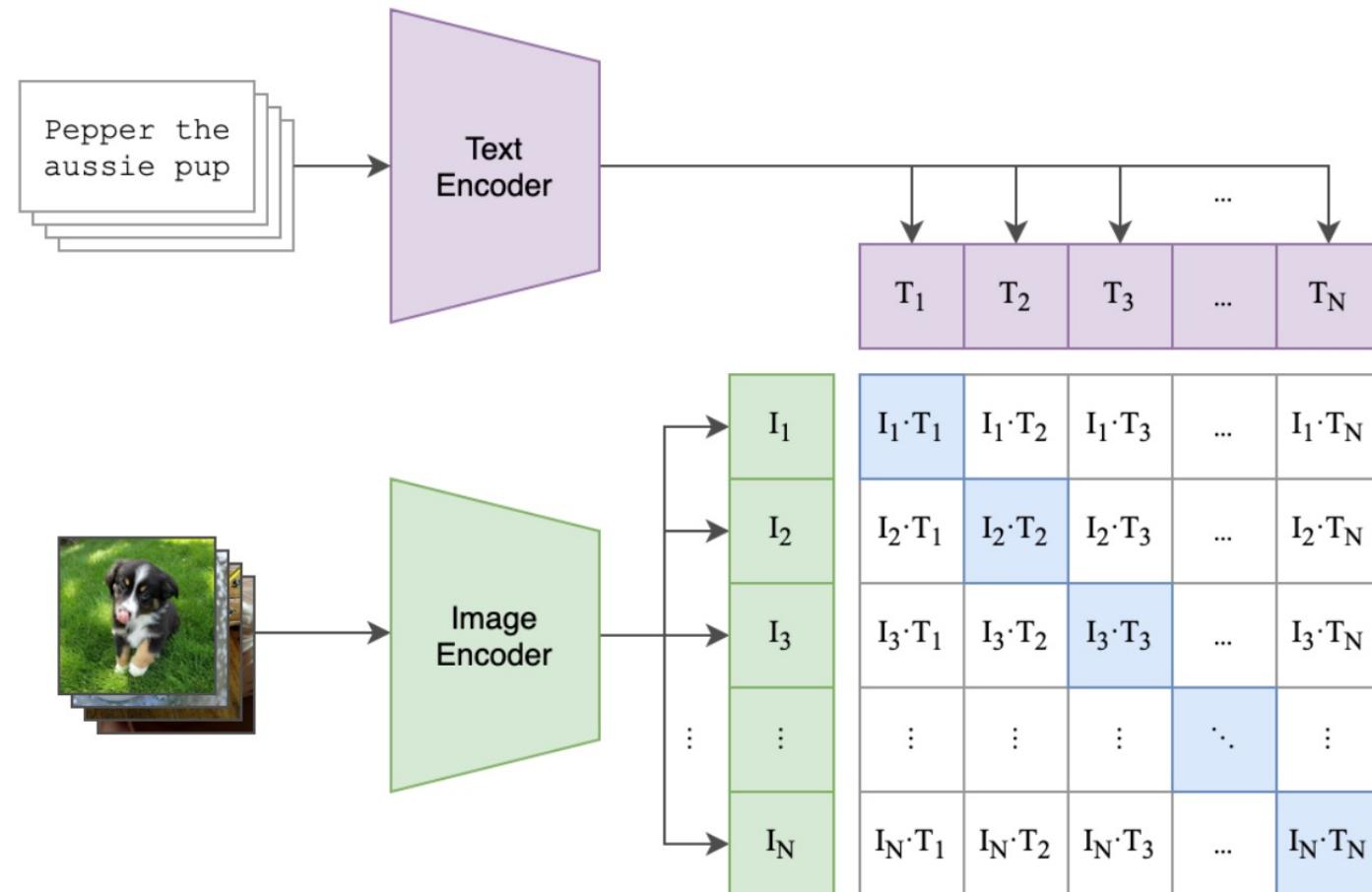
```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```



CLIP Architecture Overview

- Algorithm:

- Since each image I_i should only match its corresponding text T_i , we set the ground truth labels for this task as labels = $[1, \dots, N]$ which indicates that the correct match for each sample lies along the diagonal of the logits matrix.

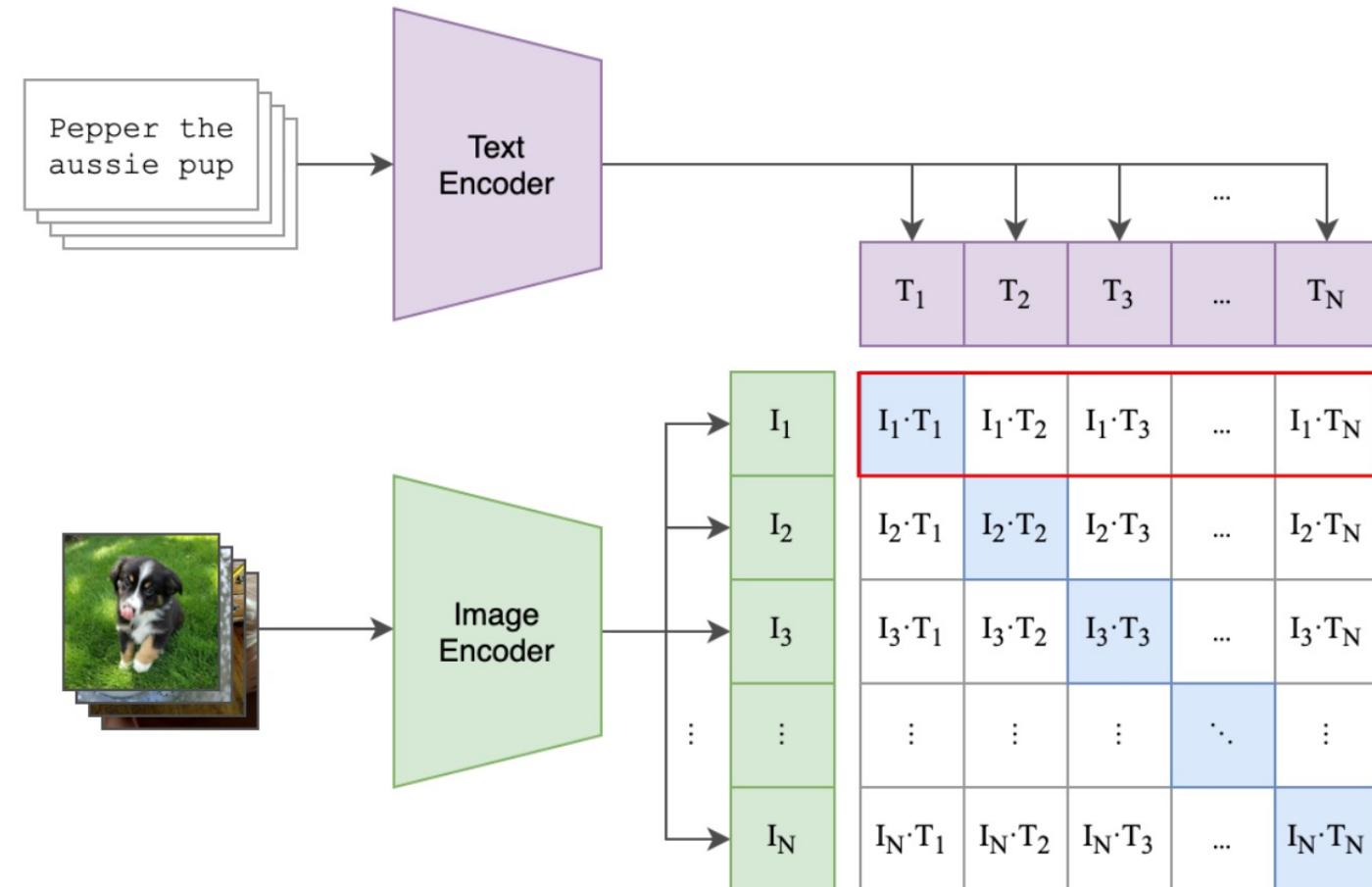
```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```



CLIP Architecture Overview

- Algorithm:

- For each image I_i , the model tries to predict the correct text T_i among all texts in the batch.
- This is done by applying cross-entropy loss across each row of the logits matrix.
 - Treating it as a classification problem where the correct class is the corresponding text.

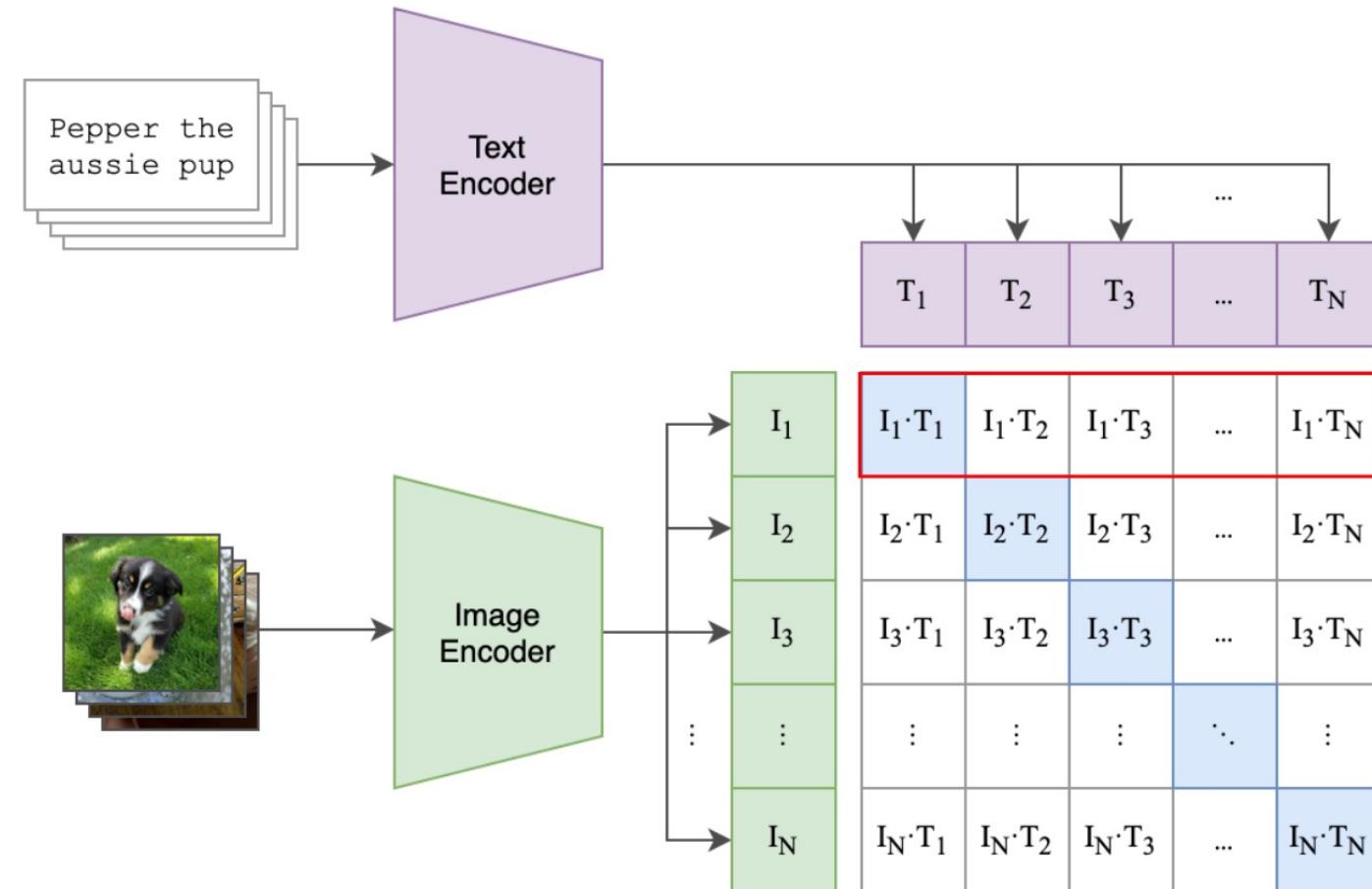
```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```



CLIP Architecture Overview

- Algorithm:

- Similarly, for each text T_i , the model tries to predict the correct image I_i among all images in batch.
- This is also done by applying cross-entropy loss across each column of the logits matrix.

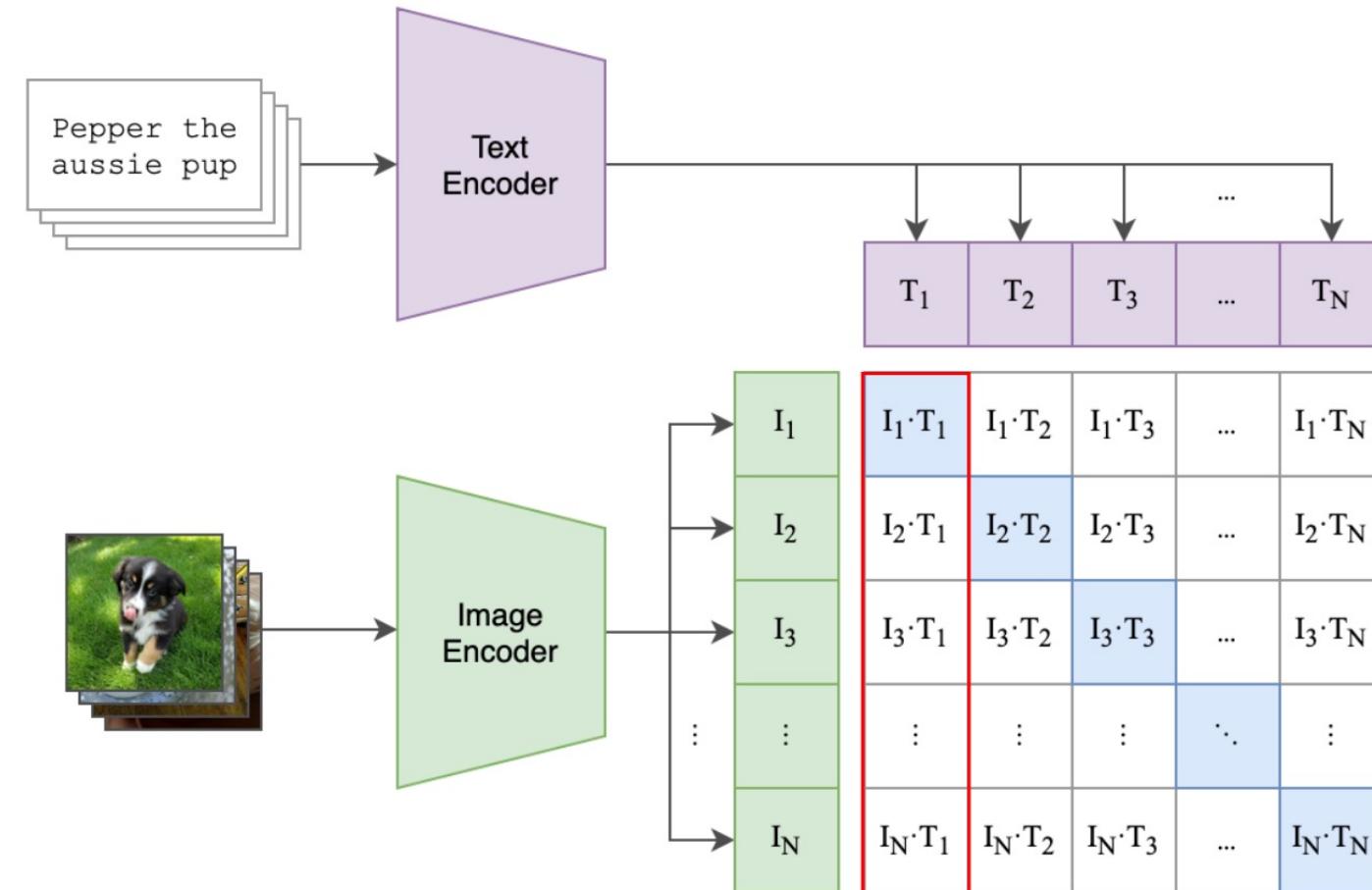
```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

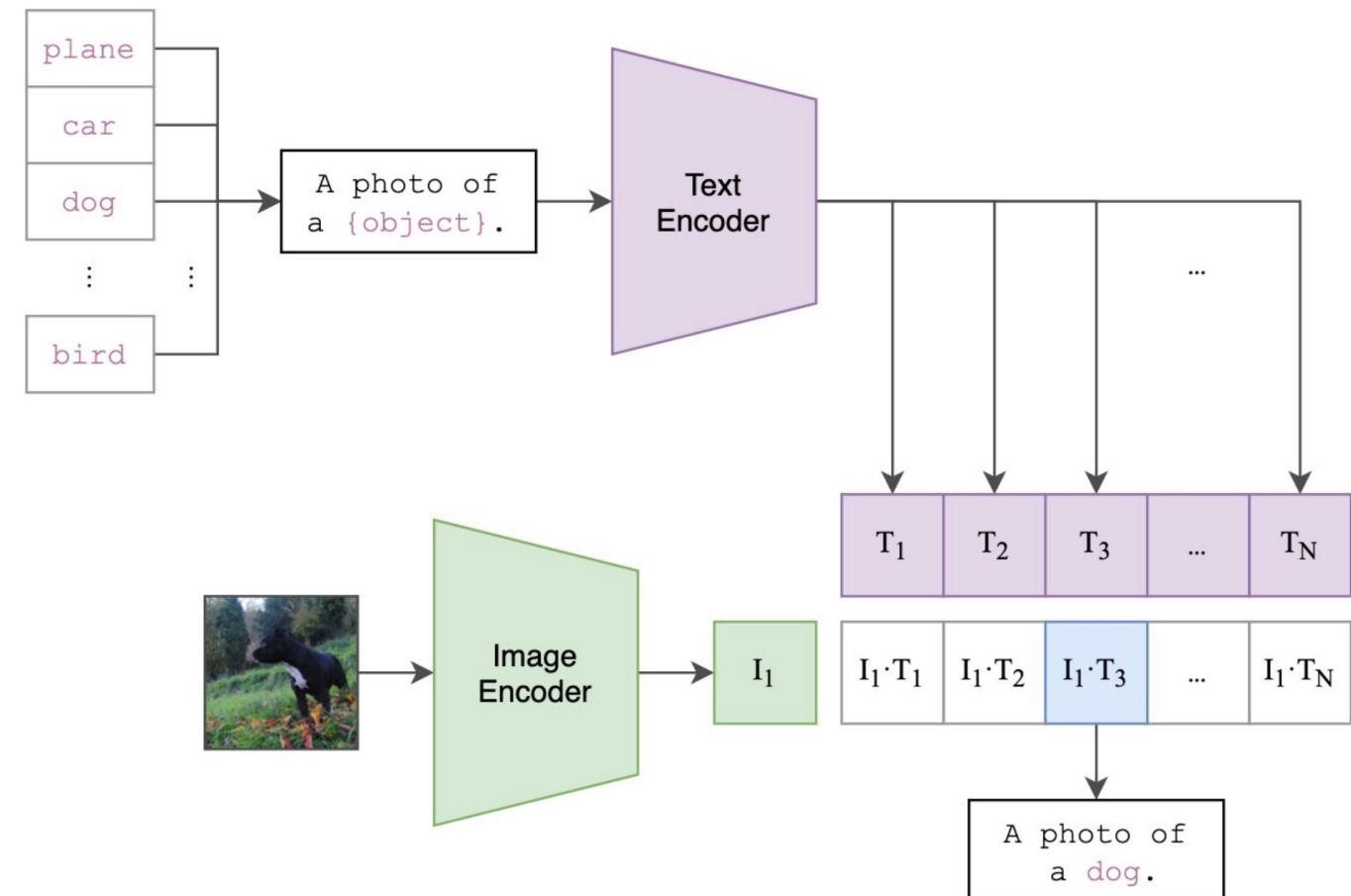
# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```



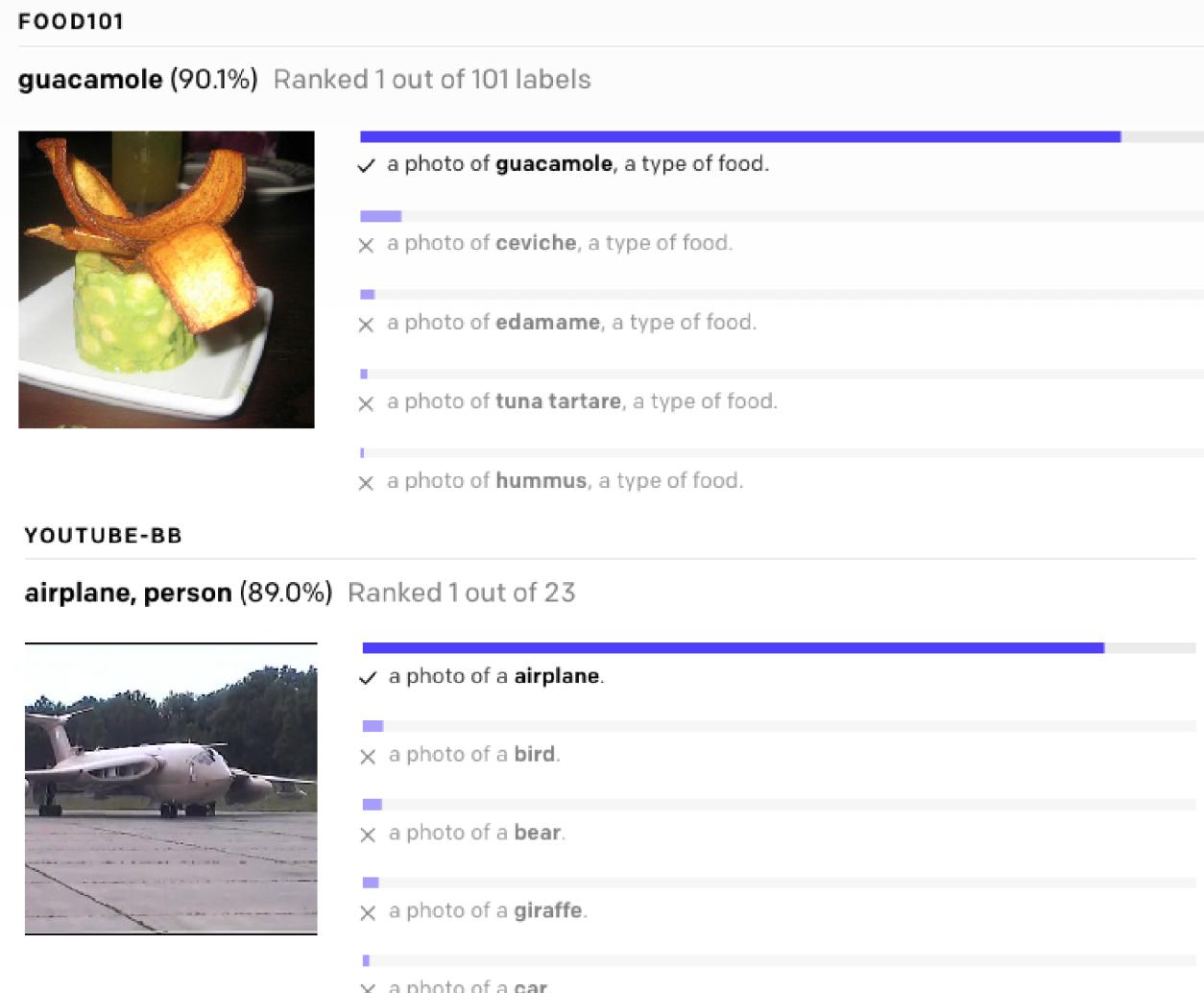
CLIP: Zero-shot classification

- At test time, the learned text synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.
- **Text Prompts as Input Categories:**
 - Categories like "plane," "car," "dog," and "bird" are each turned into descriptive phrases ("A photo of a {object}").
- **Encoding:**
 - The phrases and input image are passed to the text encoder and image encoder, respectively, to produce text and image features.
- **Calculating Similarities:**
 - A similarity score is calculated using the dot product between the image feature and each text feature.
 - The label with the highest similarity score is selected as the classification result.



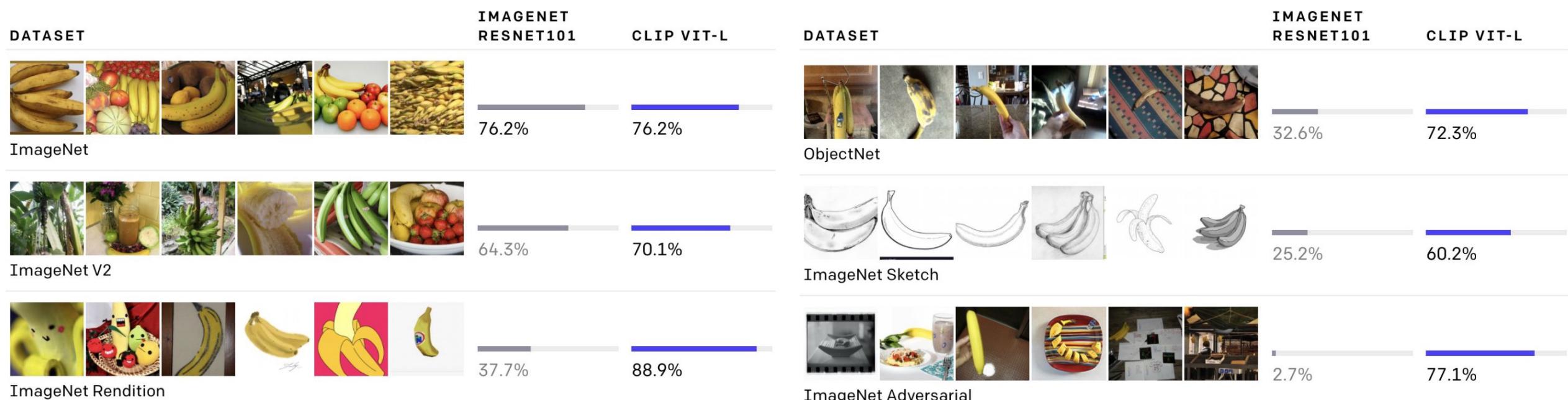
CLIP: Zero-shot classification

- At test time, the learned text synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.
- **Text Prompts as Input Categories:**
 - Categories like "plane," "car," "dog," and "bird" are each turned into descriptive phrases ("A photo of a {object}").
- **Encoding:**
 - The phrases and input image are passed to the text encoder and image encoder, respectively, to produce text and image features.
- **Calculating Similarities:**
 - A similarity score is calculated using the dot product between the image feature and each text feature.
 - The label with the highest similarity score is selected as the classification result.



Zero-shot CLIP Results

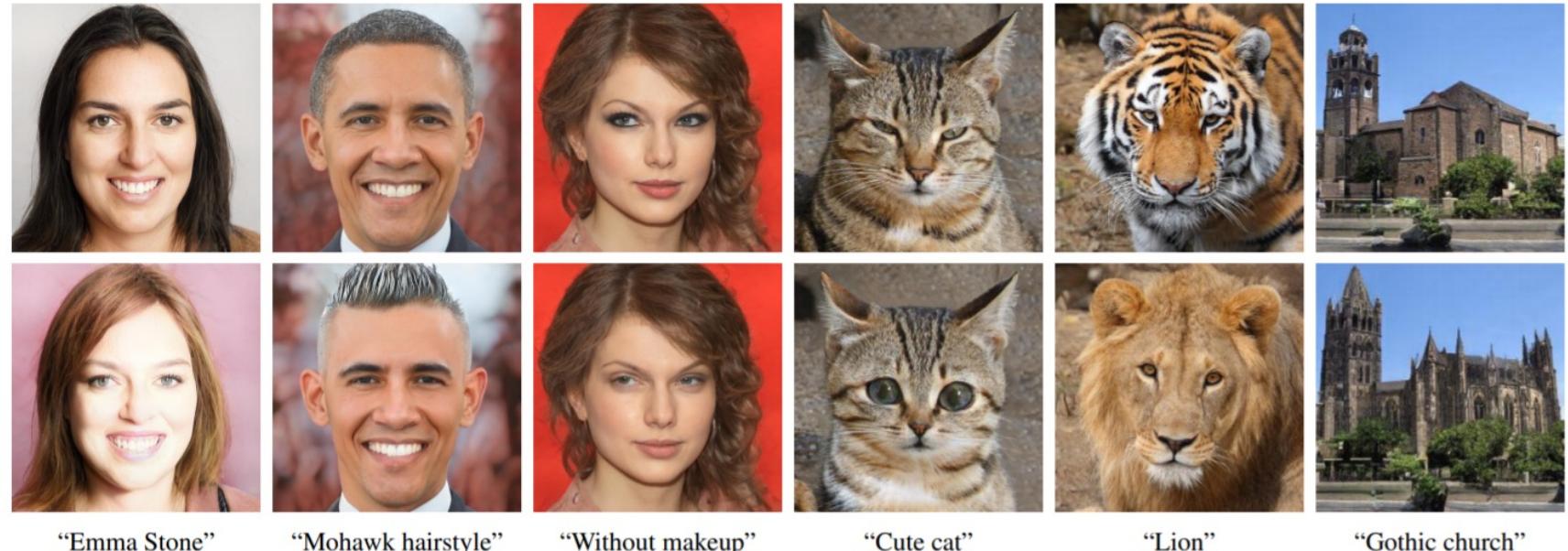
- The performance of the best zero-shot CLIP model, ViT-L/14, is compared with a ResNet-101 that has the same performance on the ImageNet validation set.
- CLIP's features are more robust to distribution shift when compared to models pre-trained on ImageNet.



Applications of CLIP

StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery

- Introduces an optimization scheme that utilizes a CLIP-based loss to modify an input latent vector in response to a user-provided text prompt.*



ClipCap: CLIP Prefix for Image Captioning

- CLIPCap builds on CLIP by leveraging CLIP's understanding of the visual domain to generate descriptive captions.*



DALL-E: Zero-Shot Text-to-Image Generation

- Can we extend ViLBERT or CLIP for text-to-image generation?
 - Generating high-quality images requires capturing fine-grained details.
 - ViT models struggle with this, often resulting in images that appear patchy and lack fine details.
- DALL-E is a dVAE model with a transformer that autoregressively models the text and image tokens as a single stream of data.
 - 12 billion parameters version of GPT-3.
 - Dataset comprised of 3.3M text-image pairs.
- DALL-E demonstrated zero-shot capabilities:
 - It can generate relevant images from text descriptions not seen during training.

Zero-Shot Text-to-Image Generation

Aditya Ramesh¹ Mikhail Pavlov¹ Gabriel Goh¹ Scott Gray¹
Chelsea Voss¹ Alec Radford¹ Mark Chen¹ Ilya Sutskever¹

Abstract

Text-to-image generation has traditionally focused on finding better modeling assumptions for training on a fixed dataset. These assumptions might involve complex architectures, auxiliary losses, or side information such as object part labels or segmentation masks supplied during training. We describe a simple approach for this task based on a transformer that autoregressively models the text and image tokens as a single stream of data. With sufficient data and scale, our approach is competitive with previous domain-specific models when evaluated in a zero-shot fashion.



Text Prompt an armchair in the shape of an avocado. . . .

AI Generated images

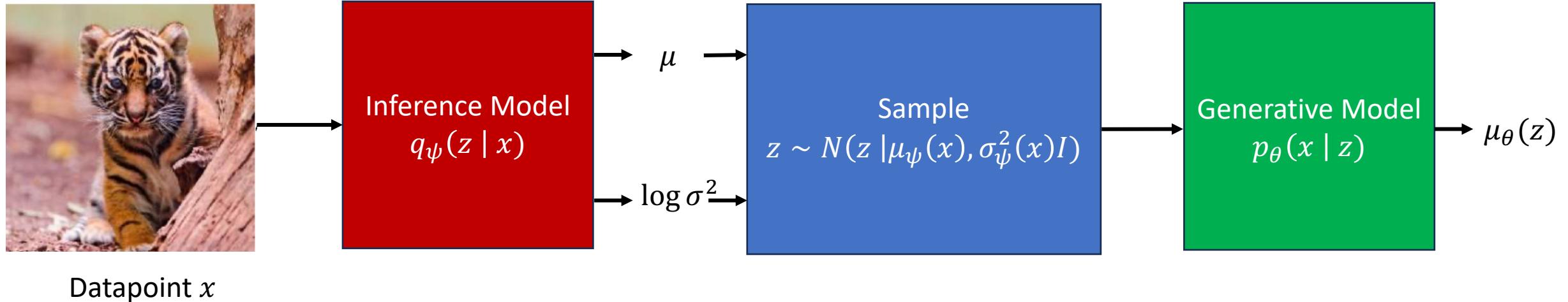


Text Prompt the exact same cat on the top as a sketch on the bottom

AI Generated images



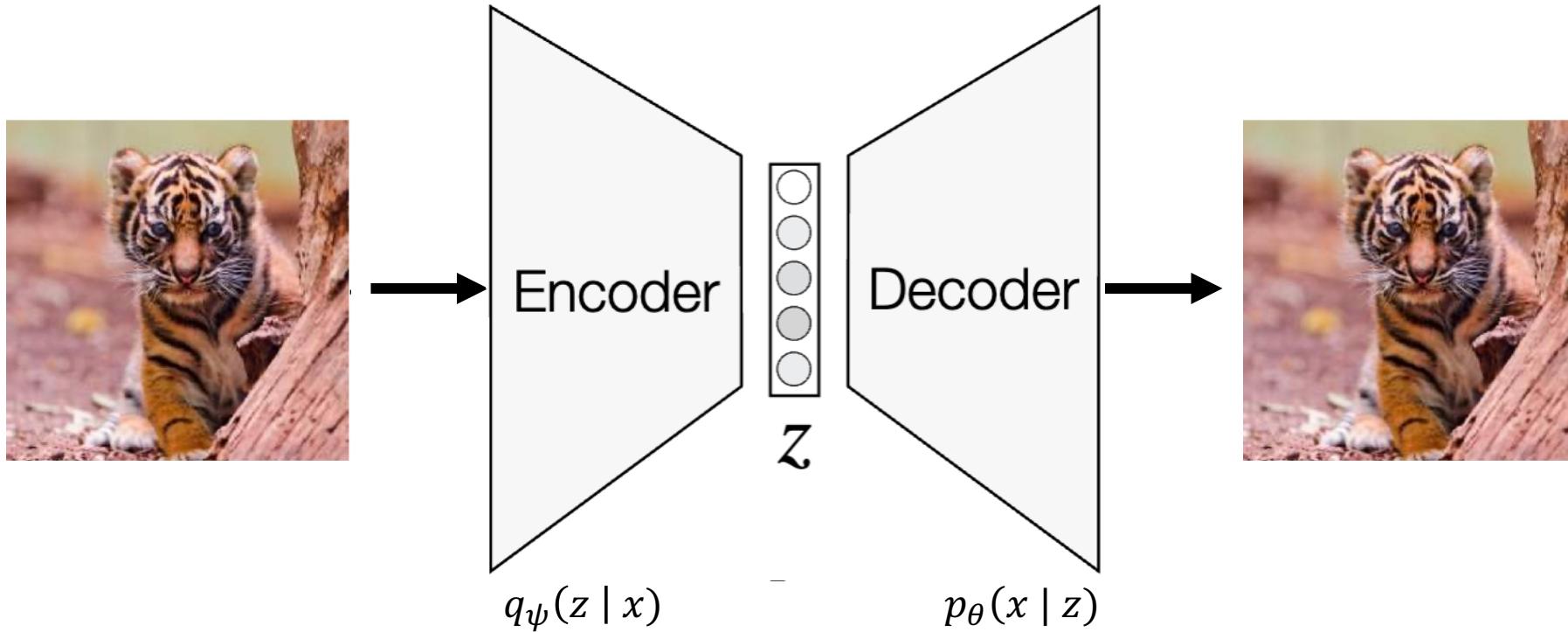
Recall: Variational Autoencoders



ELBO Objective

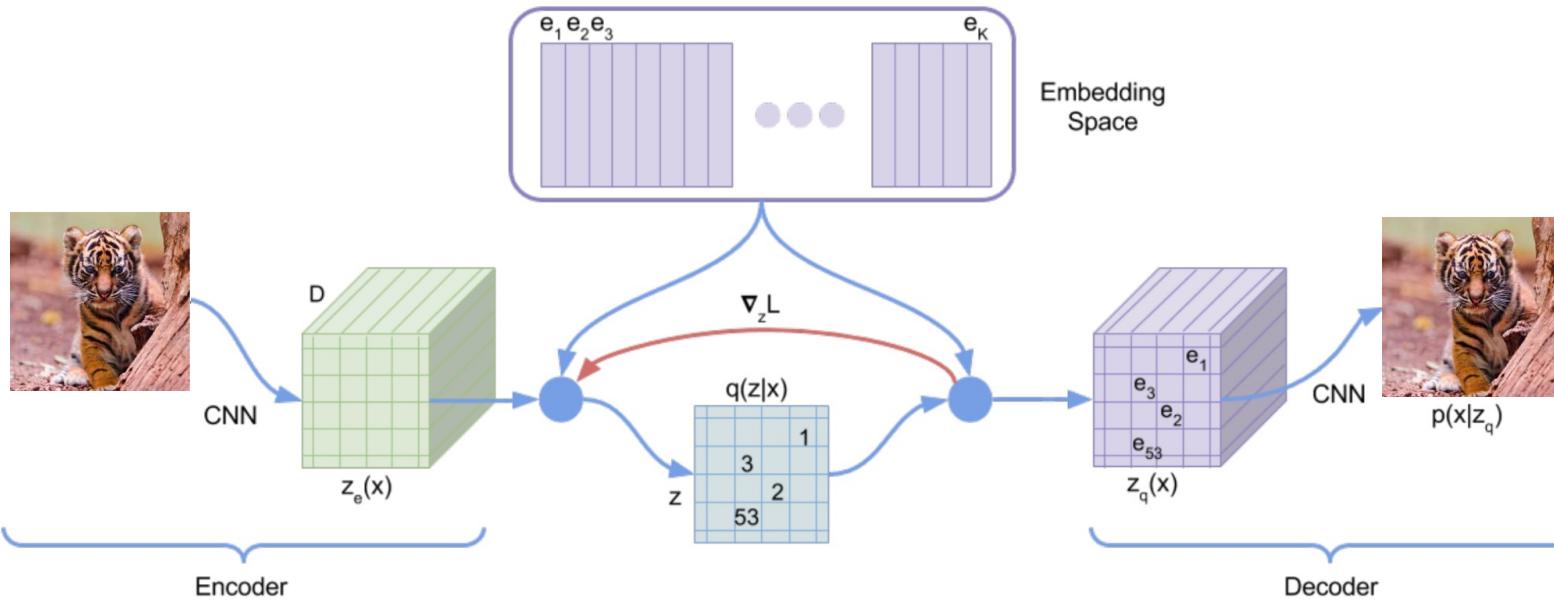
$$E_{z \sim q_\psi(z | x)} [\log p_\theta(x | z) - KL(q_\psi(z | x) || p(z))]$$

Recall: Discrete Variational Autoencoders

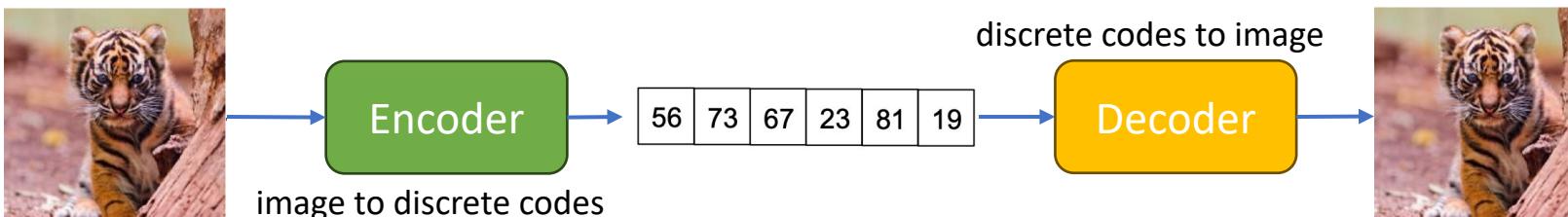


- VAEs usually use a continuous representation for latent z .
 - But a lot of data we encounter in the real-world favors a discrete representation.
 - E.g., images can be described as a collection of objects.
 - Furthermore, transformers are designed to work on discrete tokens.

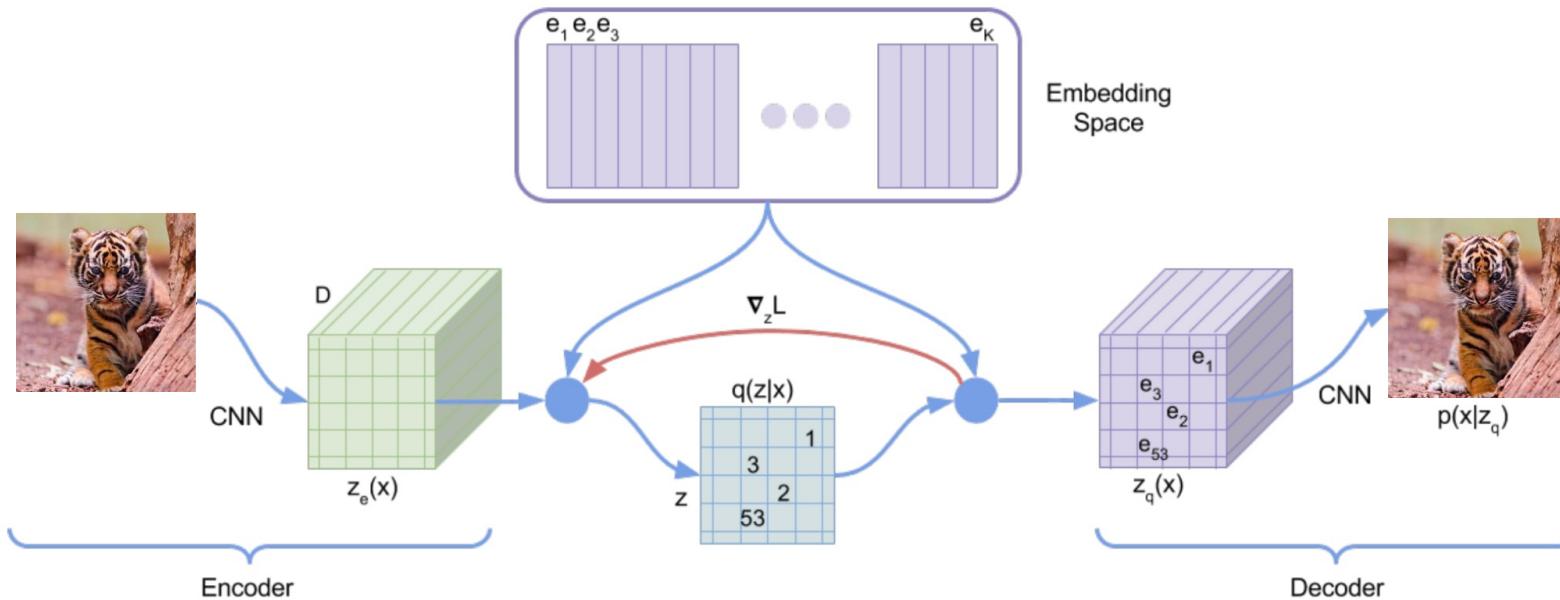
Recall: Discrete Variational Autoencoders



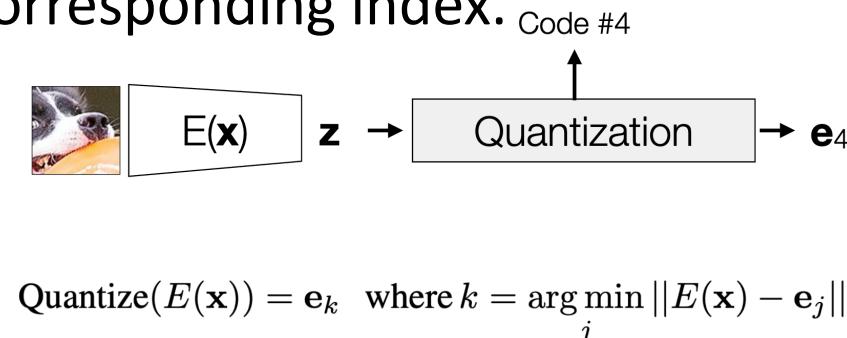
- VAEs usually use a continuous representation for latent z .
 - But a lot of data we encounter in the real-world favors a *discrete* representation.
 - E.g., images can be described as a collection of objects.
 - Furthermore, transformers are designed to work on discrete tokens.
 - Discrete VAE (dVAE): replace Gaussian latent with categorical code.



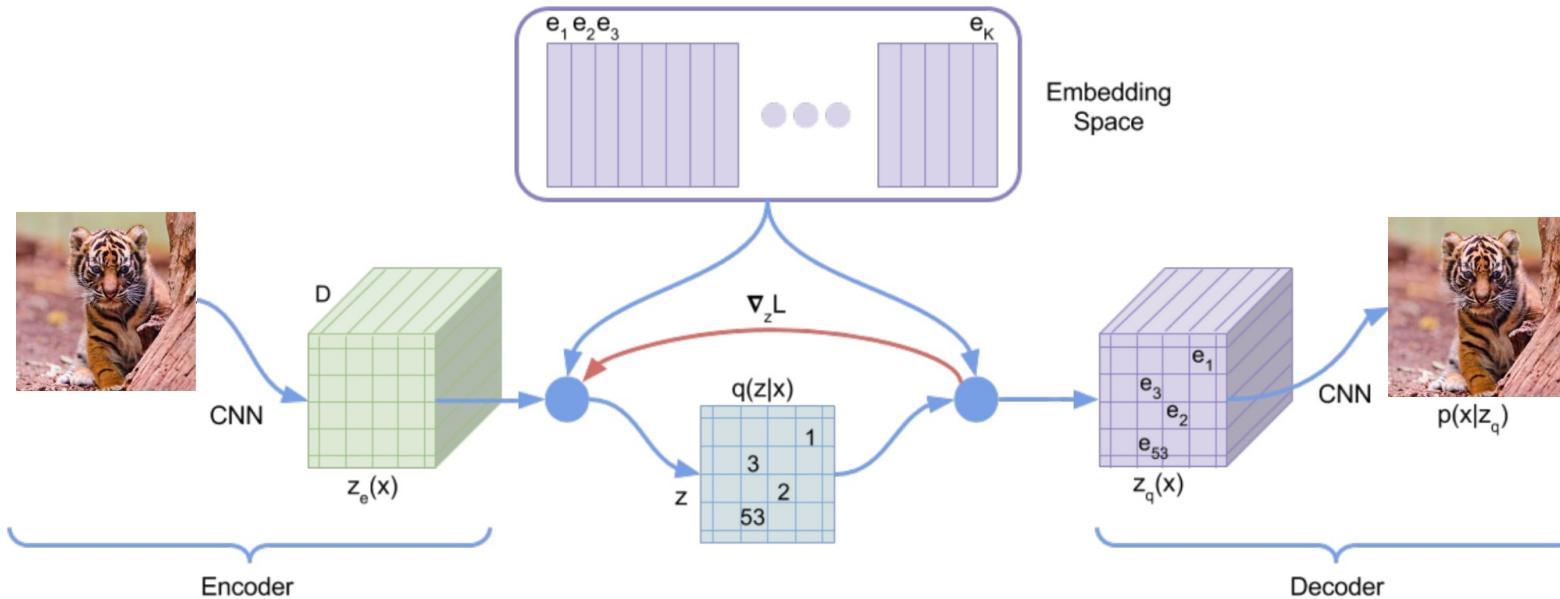
Recall: Discrete Variational Autoencoders



- Discrete VAE (dVAE): replace Gaussian latent with categorical code.
 - It modifies the standard VAE by adding a discrete **codebook** component to the network.
 - By quantizing the **latent space**, dVAE limits the possible values that the latent variables can take to codebook, a finite set of vectors associate with a corresponding index. Code #4
 - The output of the encoder network is compared to all the vectors in the codebook, and the **codebook vectors** closest in Euclidean distance are fed to the decoder to reconstruct the image.



Recall: Discrete Variational Autoencoders



- Discrete VAE (dVAE): Learning the Prior.
 - Once the dVAE is fully trained, we can learn the prior $p(z)$ over the latent codes.
 - We can then generate new data by sampling from the prior and feeding it to the decoder.
 - Given the encoder outputs a sequence of latent codes for each datapoint, we can use any autoregressive model (e.g., RNN or Transformer) to train the prior.
 - The autoregressive factorization is, given all previous latent codes in the sequence, predict the next one:

$$p(z) = p(z_1)p(z_2|z_1)p(z_3|z_1, z_2)p(z_4|z_1, z_2, z_3) \dots$$

DALL-E Model: dVAE with Autoregressive Prior

- In Text-to-Image Generation, we want to generate images x given captions y .
 - The Discrete VAE (dVAE) encodes images x into codes (tokens) z .
 - The Discrete VAE (dVAE) decoder reconstructs images x given z .
 - The Transformer learns to map captions y into codes z , to generate x via the dVAE decoder.
 - DALL-E maximizes the evidence lower bound (ELBO) on the joint likelihood of the model distribution over RGB images x , captions y , and the codes z for the encoded RGB image.

ELBO Objective

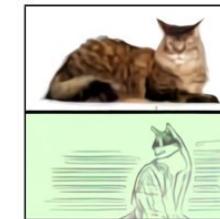
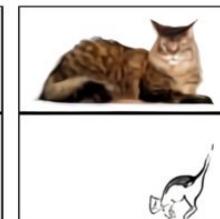
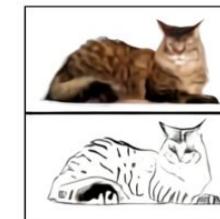
$$p_{\theta, \phi}(x, y, z) = p_{\theta}(x|y, z) p_{\phi}(y, z)$$

$$\log p_{\theta, \phi}(x, y) \geq \mathbb{E}_{z \sim q_{\psi}(z|x)} [\log p_{\theta}(x | y, z) - \beta KL(q_{\psi}(y, z | x) || p_{\phi}(y, z))]$$

- q_{ψ} denotes the distribution of a 32×32 grid of codes generated by **the dVAE encoder** for a 256×256 RGB image.
- p_{θ} denotes the distribution of RGB images generated by **the dVAE decoder** given the codes.
- p_{ϕ} denotes the joint distribution over the text and image codes modeled by **the transformer**.

DALL-E: Two Stage Training

- Stage 1: Train a discrete variational autoencoder (dVAE) to compress each RGB image (256 x 256) into a smaller image token grids (32 x 32).
- Stage 2: Train an autoregressive transformer from the concatenation of text tokens with image tokens.
- Some final samples shown below:



(a) a tapir made of accordion.
a tapir with the texture of an
accordion.

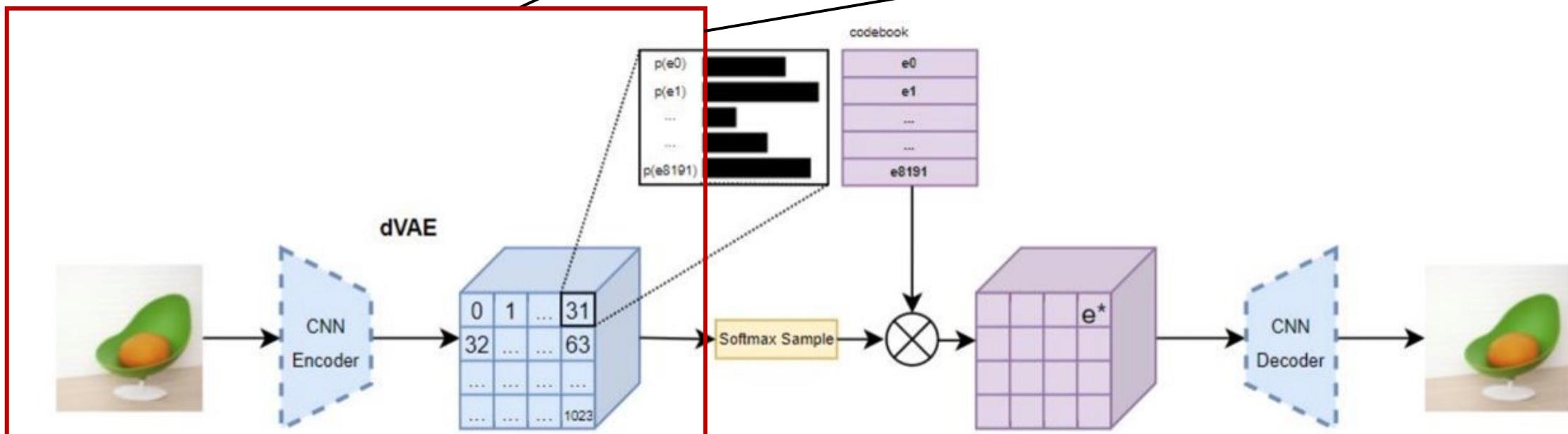
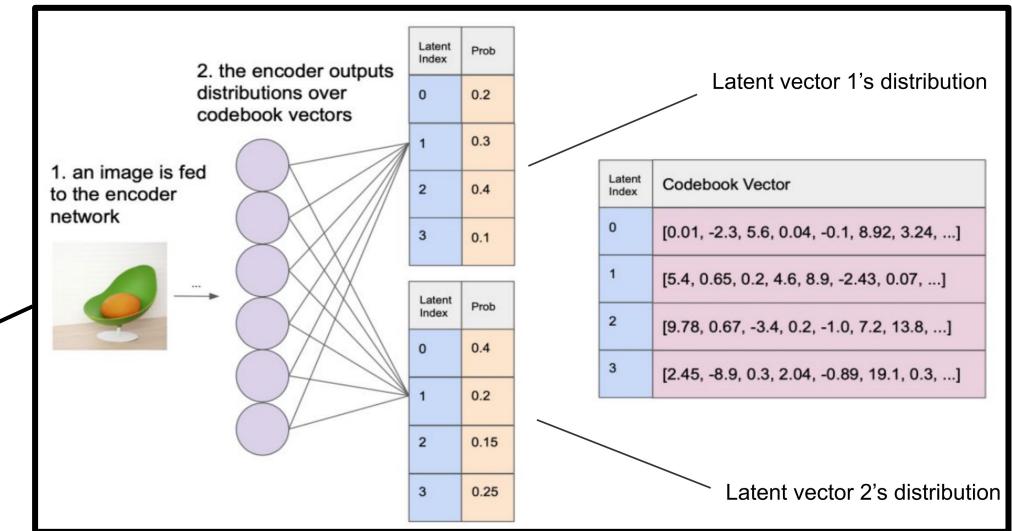
(b) an illustration of a baby
hedgehog in a christmas
sweater walking a dog

(c) a neon sign that reads
"backprop". a neon sign that
reads "backprop". backprop
neon sign

(d) the exact same cat on the
top as a sketch on the bottom

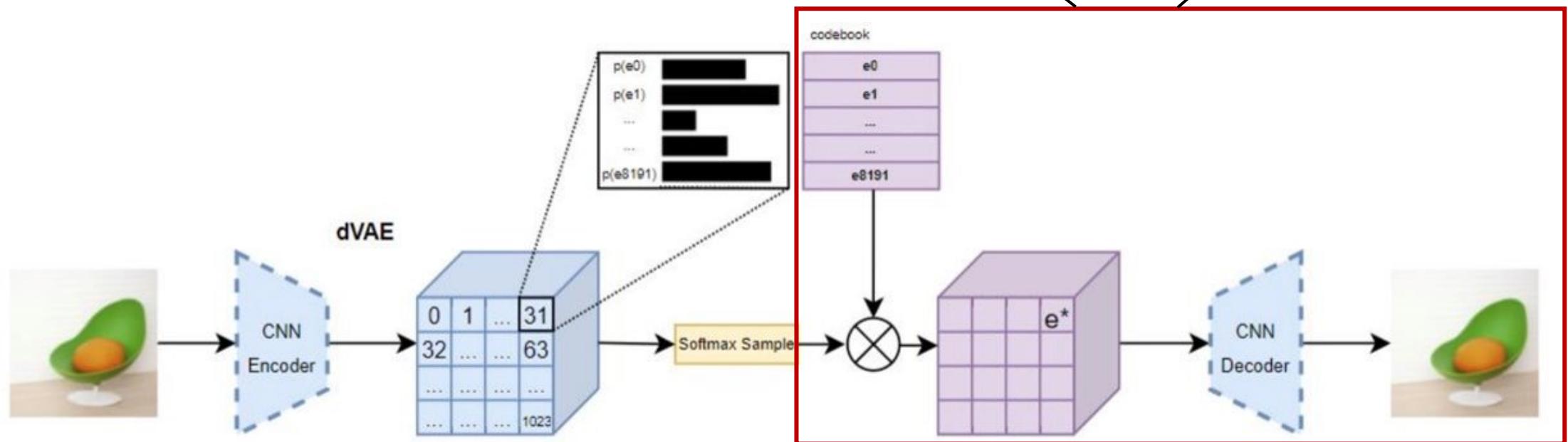
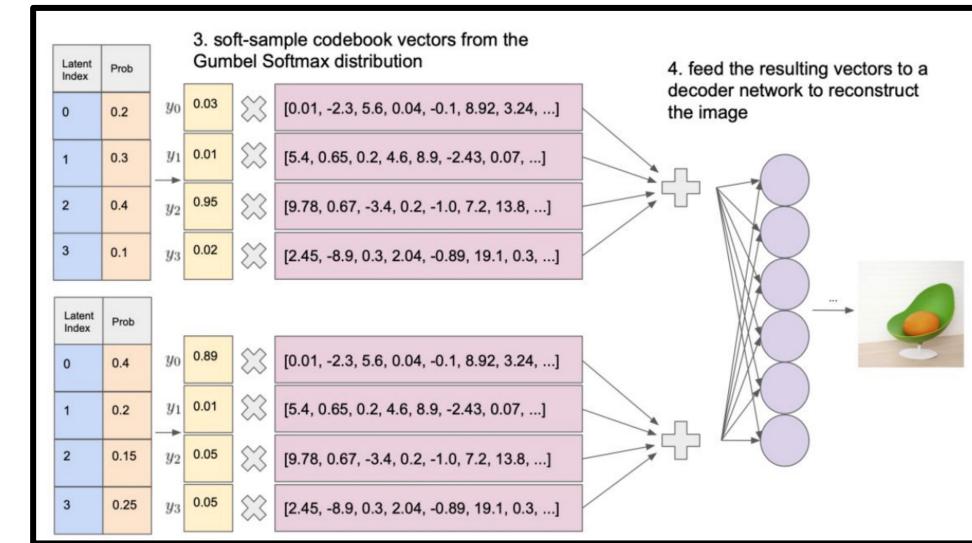
DALL-E Stage 1: Learning the Visual Codebook

- The dVAE is trained to map 256x256 image to a fixed-size grid of codebook vectors. For DALL-E, the grid size is 32x32, and the codebook size is 8192, meaning there are 8192 unique vectors any grid element can be mapped to.



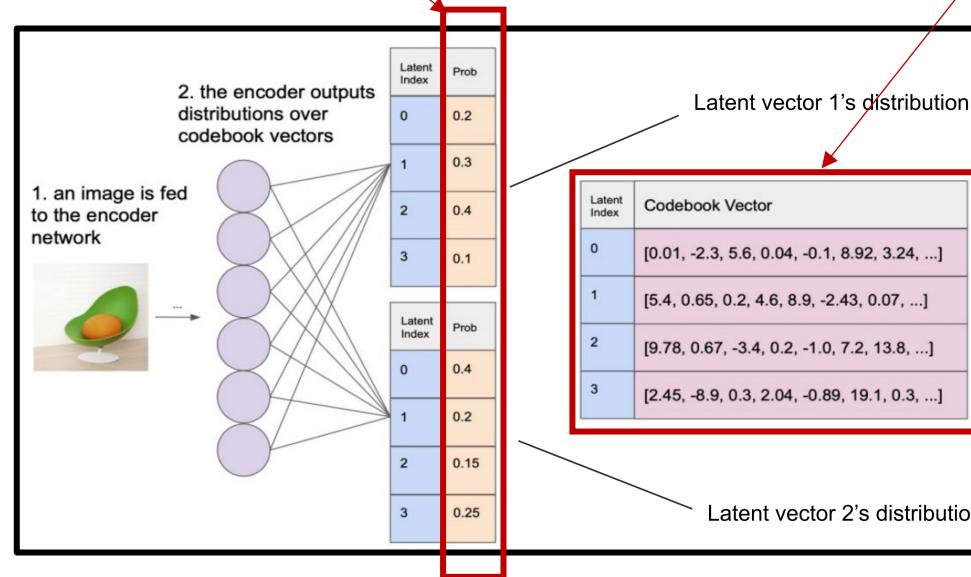
DALL-E Stage 1: Learning the Visual Codebook

- The dVAE decoder takes the distribution of latent index and the codebook vectors to ensemble the latent vector.
- Then it reconstructs the image from this latent vector.



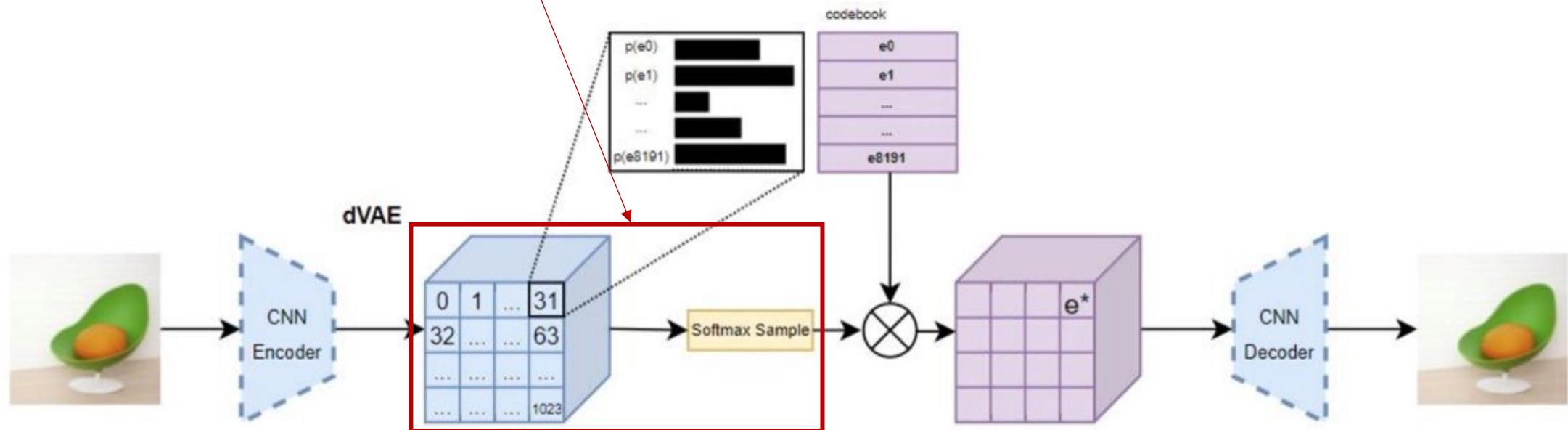
DALL-E Stage 1: Learning the Visual Codebook

- When an image is processed through the encoder of the dVAE, it is transformed into the probability distribution of a set of indices pointing to the codebook vectors, effectively compressing the image.



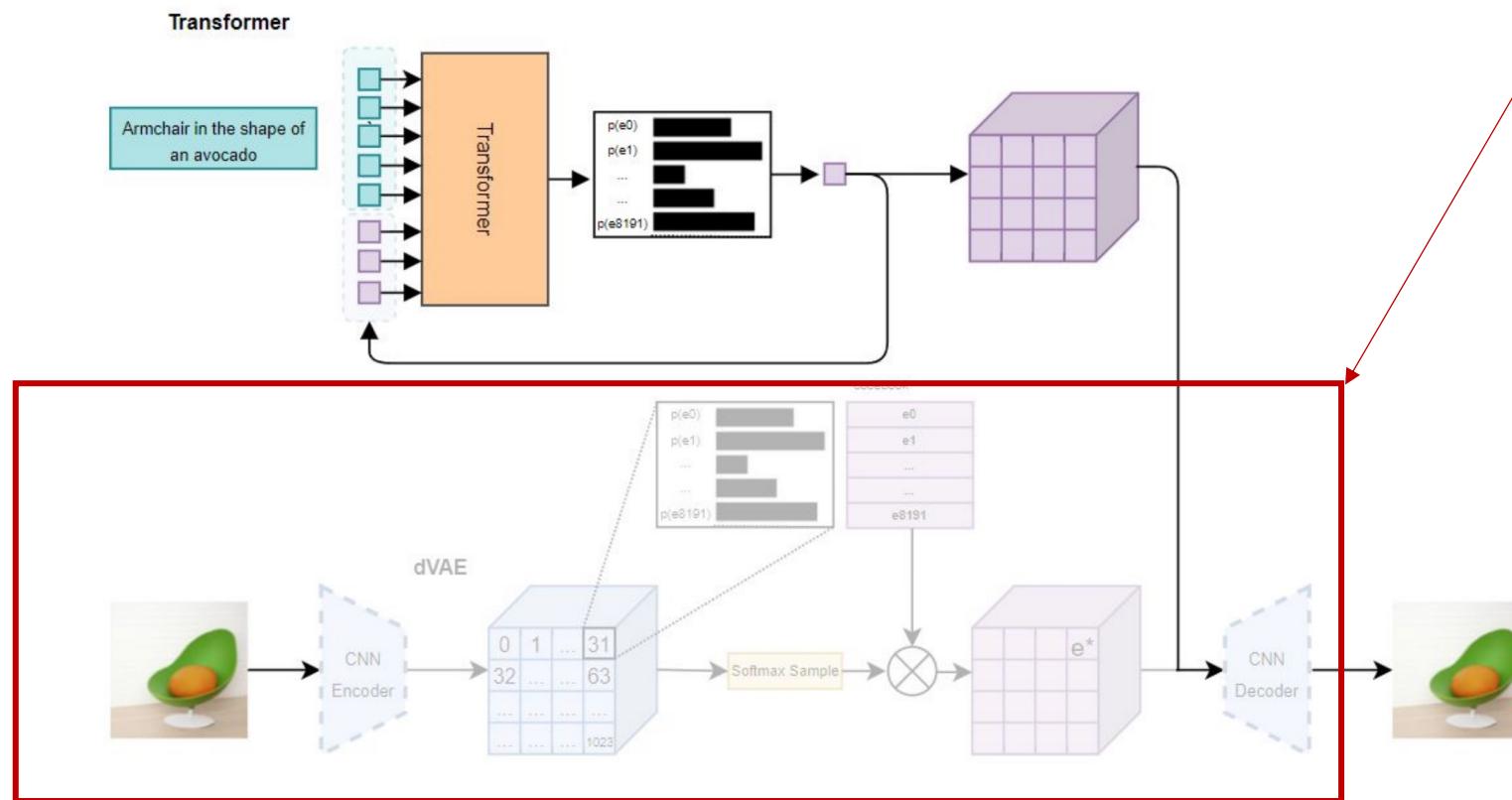
DALL-E Stage 1: Learning the Visual Codebook

- The process of selecting indices from a codebook is inherently discrete and non-differentiable.
- DALL-E uses the Gumbel-Softmax distribution as a continuous relaxation of the discrete distribution.
 - This is done by adding Gumbel noise to the logits (the inputs to the softmax function) and applying a softmax function with a temperature parameter, τ .



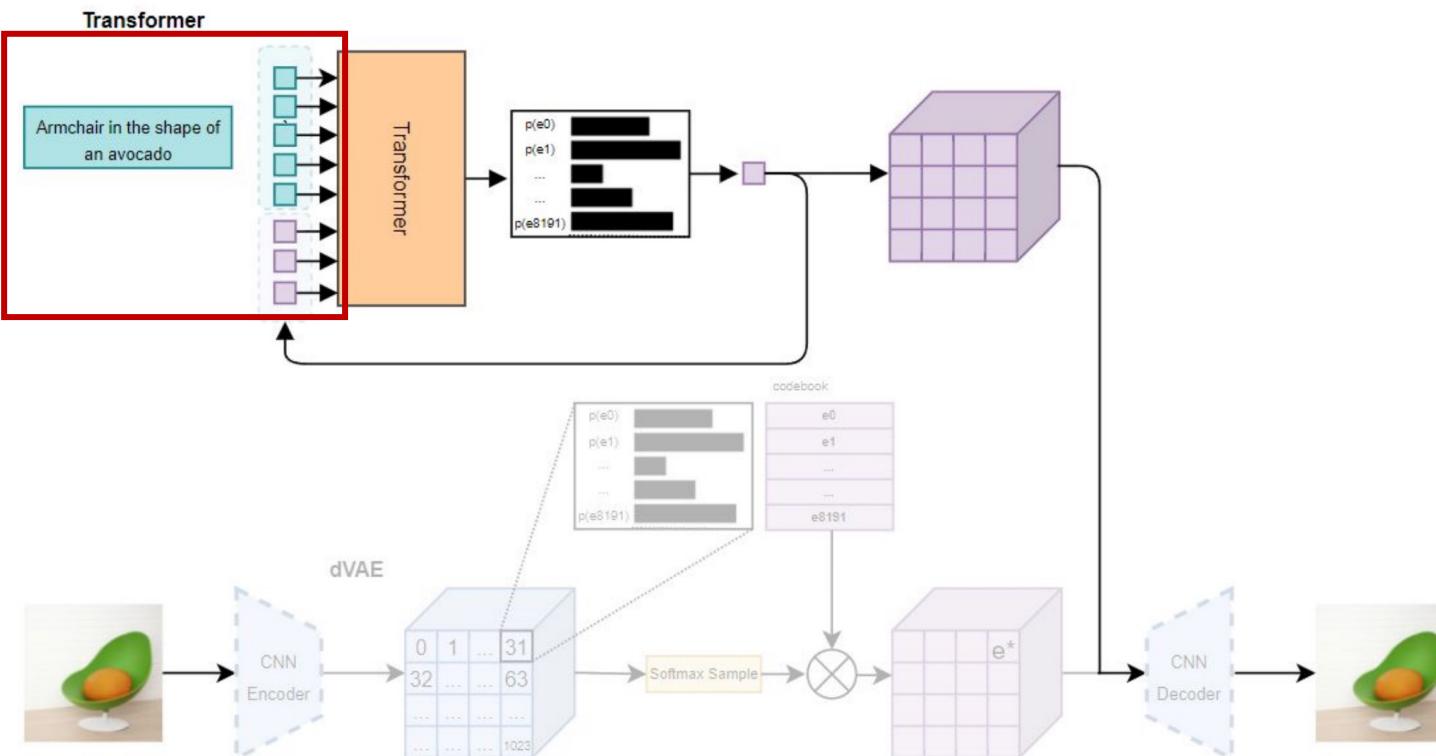
DALL-E Stage 2: Learning the Prior Distribution

- The parameters from Stage 1, which include the encoder (that maps images to tokens) and the decoder (that reconstructs images from tokens), are frozen.
- Indeed, we can now use the frozen decoder to generate images. But how to align with a text (i.e., the user prompt)?



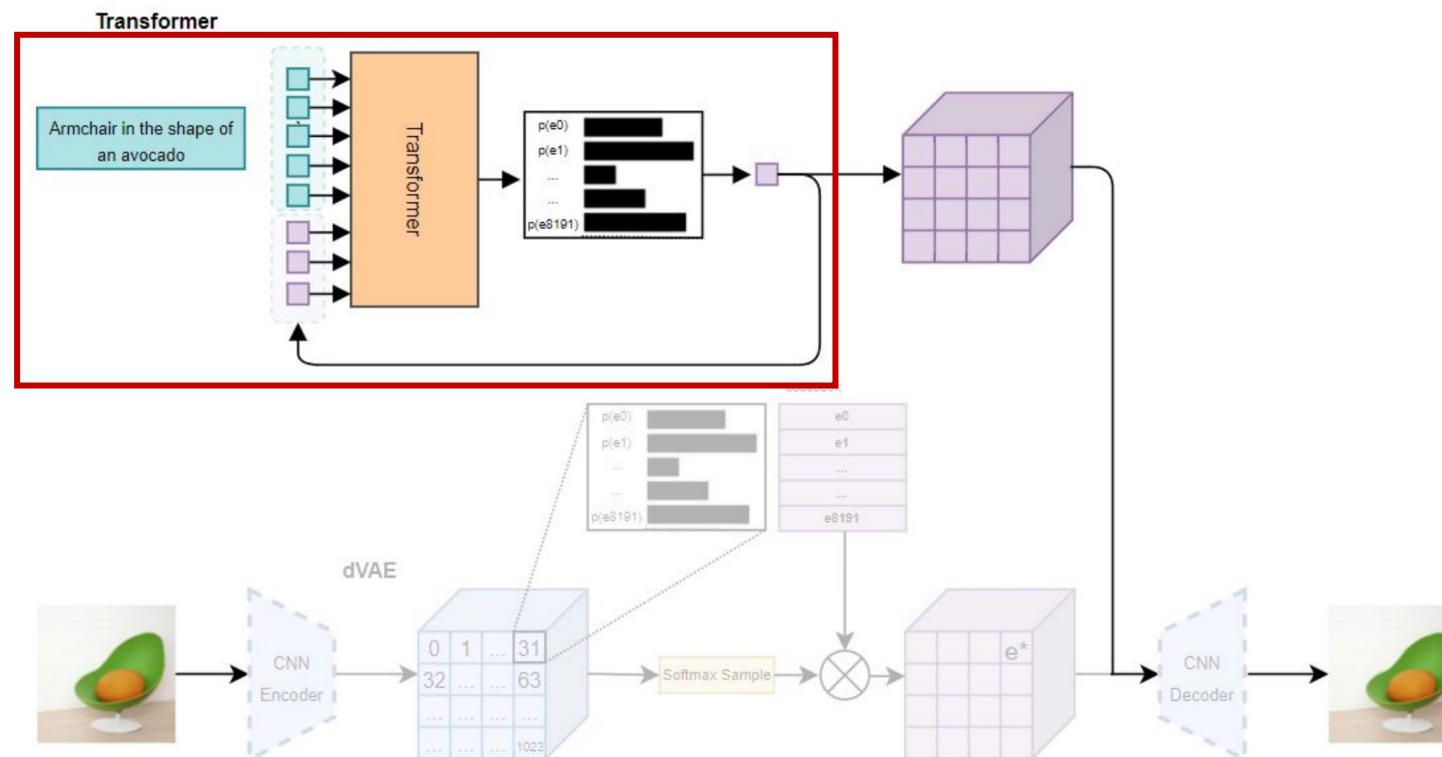
DALL-E Stage 2: Learning the Prior Distribution

- The text prompt is tokenized using Byte Pair Encoding (BPE), which breaks down the text into a sequence of up to 256 tokens.
- For images, the previously trained dVAE encoder is used to represent each image as a sequence of 1024 tokens (which are indices of the learned visual codebook).



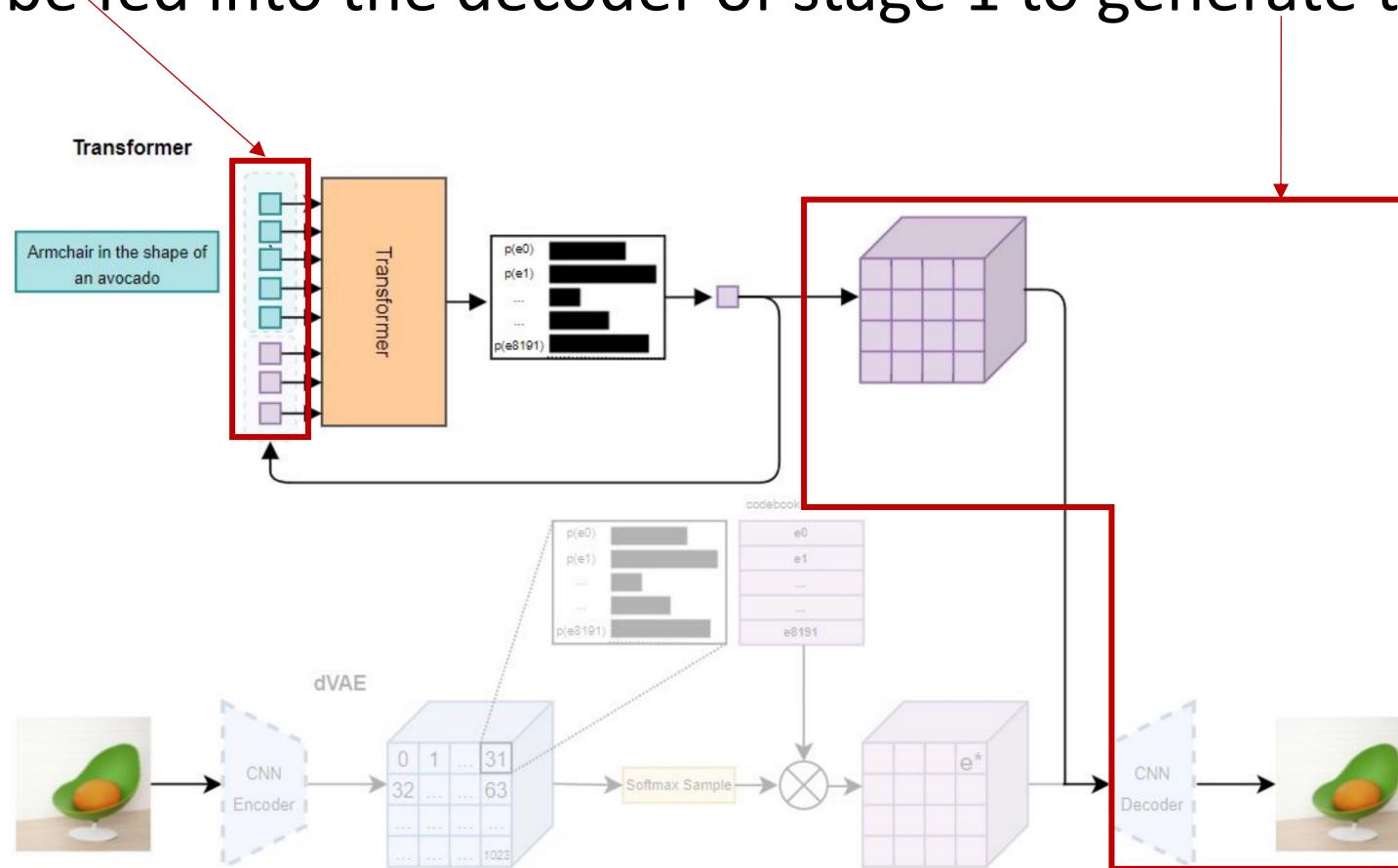
DALL-E Stage 2: Learning the Prior Distribution

- A 12B autoregressive transformer.
- **Input (Concatenated Sequence):** The text and image tokens are concatenated to form a single input sequence to the transformer.
- **Desired Output (Image Tokens):** the model is trained to predict image tokens.



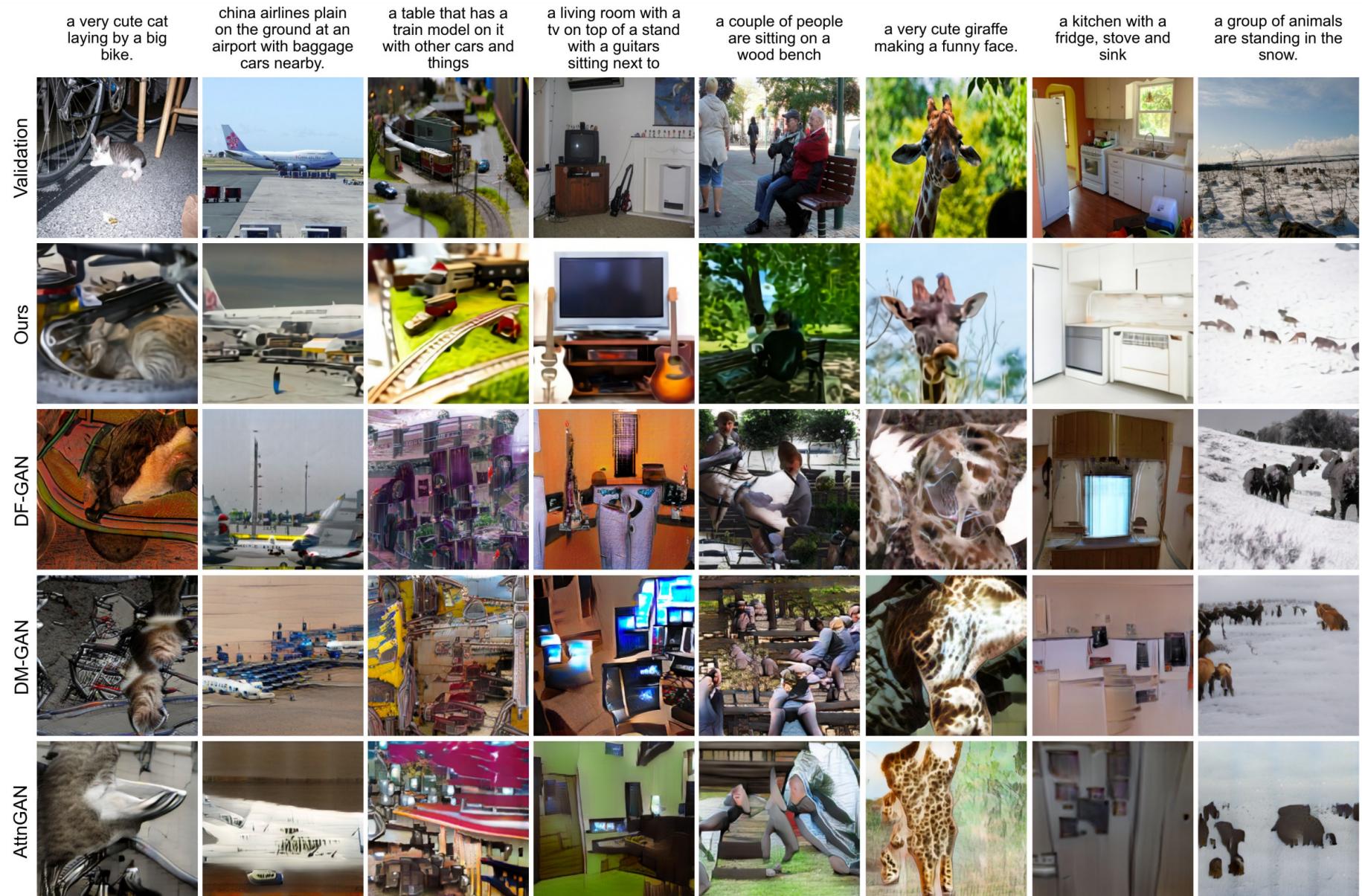
DALL-E Stage 2: Learning the Prior Distribution

- This sequence also includes special tokens for padding, **start-of-text**, and **start-of-image** markers to distinguish different parts of the input.
- In the inference stage, given **text tokens**, the predicted **image tokens** by the transformer will be fed into the decoder of stage 1 to generate the image.



Text to Image Results

DALL-E



Text to Image Results

a snail made of harp. a snail with the texture of a harp.



a sphere made of porcupine. a sphere with the texture of a porcupine.

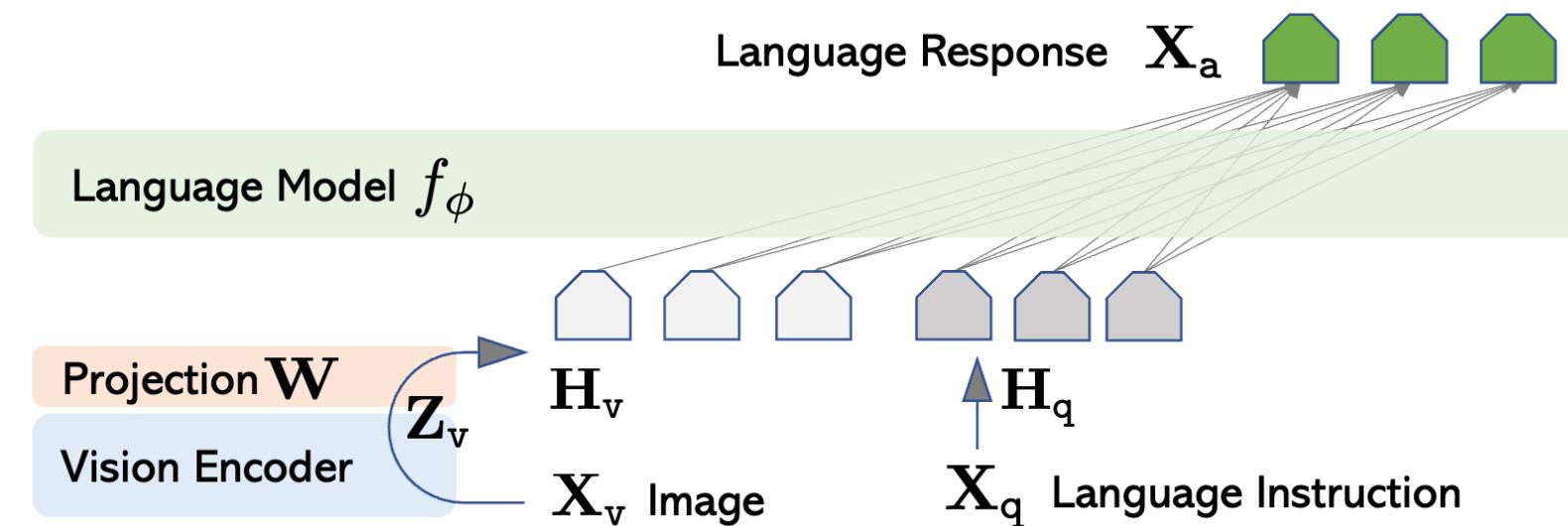


DALL-E: Summary

- DALL-E is a combination a dVAE model with a 12-billion parameter transformer, specifically trained to generate images from textual descriptions.
- Architecture:
 - Discrete VAE: Uses dVAE to encode images into discrete codes and decode images given the discrete codes.
 - Transformer: Uses an autoregressive transfer model, trained with a dataset of text-image pairs.
- Key Features:
 - Text-to-Image Generation: DALL-E generates images based on descriptive text inputs, creating objects, scenes, and imaginative visuals from scratch.
 - Zero-Shot Learning: Capable of generating novel image concepts that were not explicitly part of its training, demonstrating zero-shot reasoning.

LLaVA: Large Language and Vision Assistant

- LLaVA aims to integrate a large language model (LLM) with vision capabilities to understand and generate contextual dialogue based on visual input.
- Model Architecture -- Two Main Components:
 - Visual Encoder: Utilizes a pre-trained image encoder (e.g., CLIP) to extract visual features from images. These features are used to contextualize visual information.
 - Language Decoder: A large language model (e.g., Vicuna) that processes both text and the output from the visual encoder, enabling the generation of human-like dialogue.
- Fusion of Modalities:
 - The integration process involves mapping image features to the LLM's token space, making the visual information compatible for use by the language model.



Instruction Tuning in LLMs

- Instruction-tuning is a training process where an LLM learns to follow natural language instructions by being exposed to a diverse set of queries and responses.
 - This improves a model's ability to understand and respond to a broad variety of user prompts, making interactions more natural, coherent, and context-aware.
- How do we collect the instruction tuning data?
 - Human: high-quality, written by humans – high cost
 - Machine: strong LLM-based teacher like ChatGPT – affordable cost

Instruction

Explain human's behavior.
Behavior: cry.

Answer

There could be many reasons why a person might cry.
They could be feeling sad, scared, angry, or
frustrated...

Recommend a movie for me.

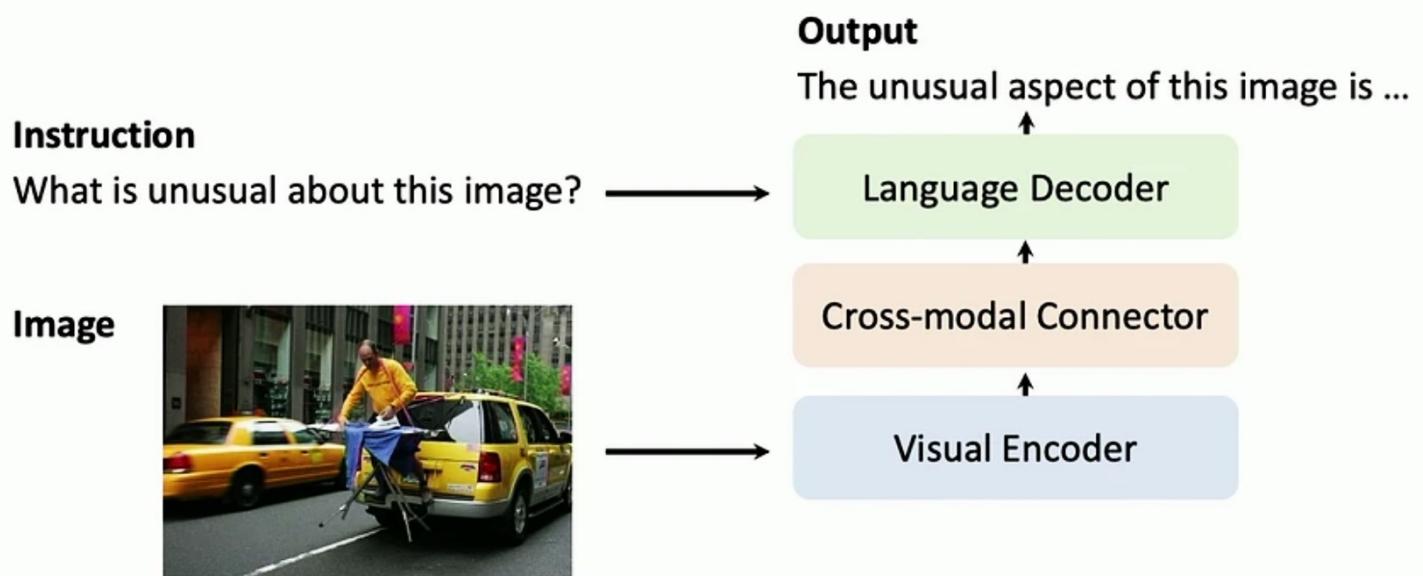
Certainly! Without specific preferences in mind, I'll
recommend a popular and highly-rated movie: "The
Shawshank Redemption." It's a classic drama film ...

...

...

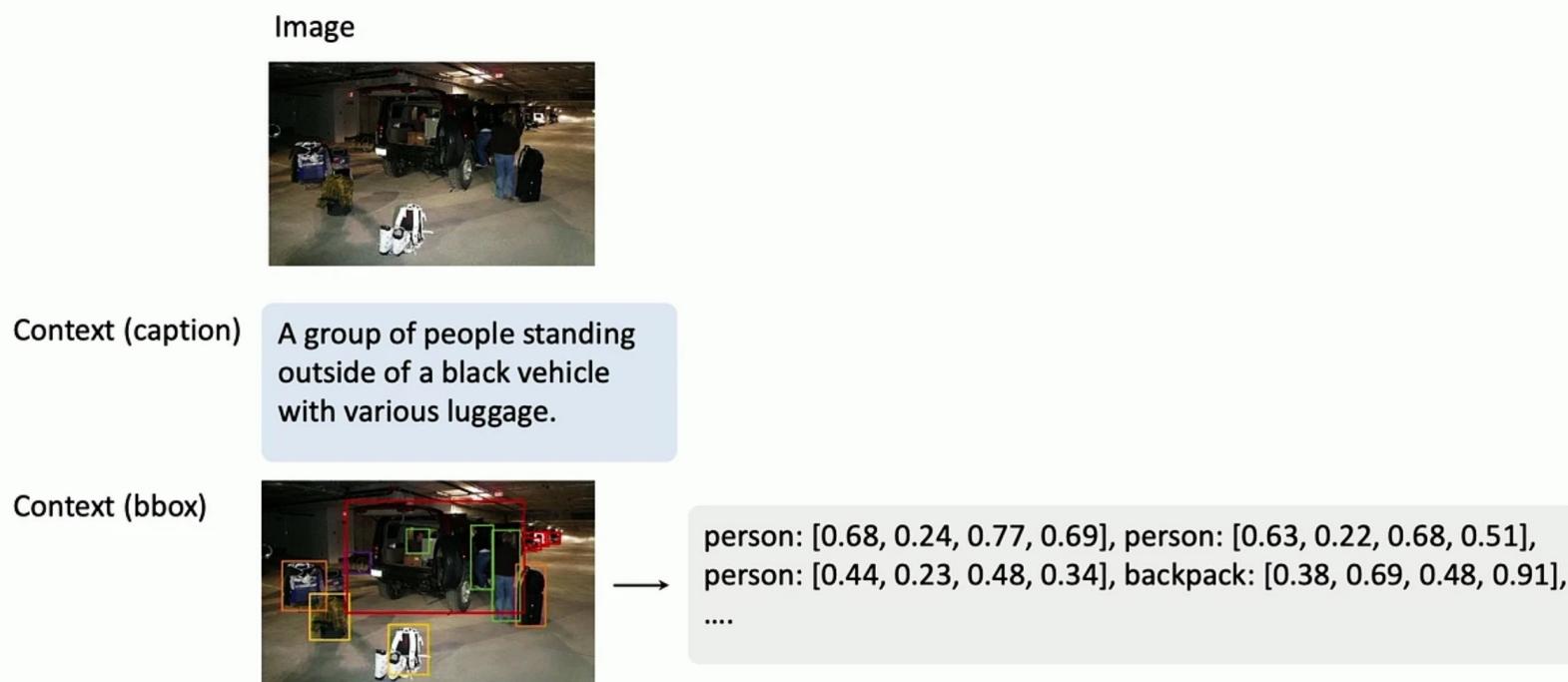
Visual Instruction Tuning in LLaVA

- In LLaVA, instruction-tuning is tailored to handle both text and visual inputs, teaching the model how to interpret and respond to complex multimodal instructions.
- However, we don't have multi-modal instruction tuning datasets.
 - LLaVA uses image-caption pairs and question-answer sets to simulate real-world queries about images.



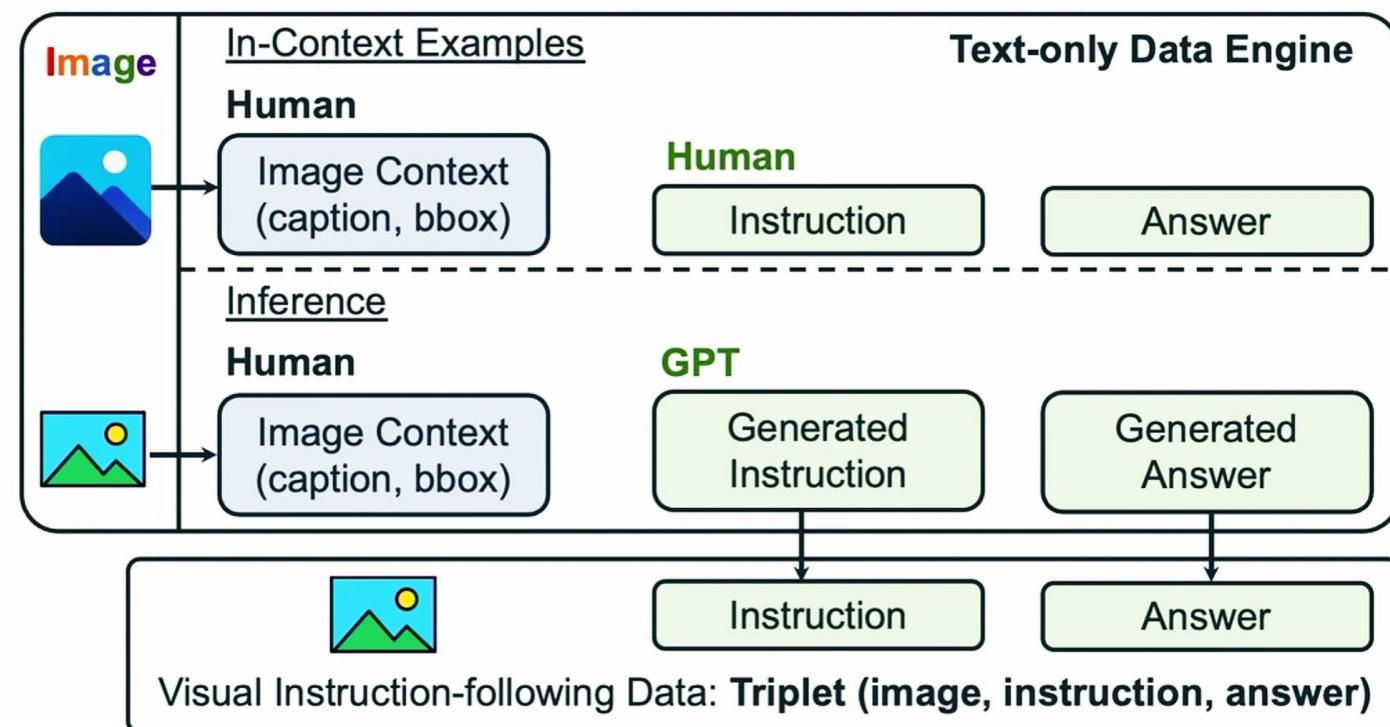
GPT-Assisted Visual Instruction Data Creation

- LLVA uses image-caption pairs and question-answer sets to simulate real-world queries about images.
 - Given an image and its corresponding caption, a natural extension is to generate a list of questions. GPT-4 is prompted to create a set of questions and responses.
 - To generate richer data that encodes visual content using symbolic representations:
 - Captions: Provide descriptions of the visual scene from multiple perspectives.
 - Bounding Boxes: Mark locations of objects in the image, providing both object identification and spatial context.



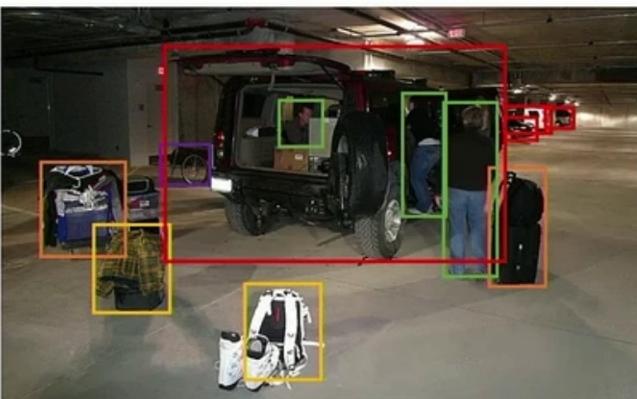
GPT-Assisted Visual Instruction Data Creation

- Human Annotations:
 - Humans provide the initial In-Context Examples in the form of captions and bounding boxes along with instructions and answers.
- GPT-based Inference:
 - GPT-4 is prompted with the initial human-generated context to generate new instructions and answers.
- The generated data forms a triplet:
 - Image
 - Instruction based on context
 - Answer to the posed instruction



GPT-Assisted Visual Instruction Data Creation

- A total of 158K triplet samples were generated:
 - 58K conversations
 - 23K detailed descriptions
 - 77K complex reasoning



Response type 1: conversation

Question: What type of vehicle is featured in the image?

Answer: The image features a black sport utility vehicle (SUV).

Question: Where is the vehicle parked?

Answer: The vehicle is parked in an underground parking area, likely in a public garage.

Question: What are the people in the image doing?

Answer: The people are trying to fit all of their luggage into the SUV, likely preparing for a trip.

Response type 2: detailed description

The image is an underground parking area with a black sport utility vehicle (SUV) parked. There are three people in the scene, with one person standing closer to the left side of the vehicle, another person in the middle, and the third person on the right side. They are all working together to pack their luggage into the SUV for a trip. (omitted)

Response type 3: complex reasoning

Question: What challenges do these people face?

Answer: In the image, a group of people is standing outside a black SUV in a parking area, surrounded by various pieces of luggage, including suitcases and backpacks. They are facing the challenge of fitting all their luggage into the black SUV. There are multiple suitcases and backpacks to be packed, which suggests that the group has a significant amount of belongings to accommodate. They might have to strategize and arrange the luggage efficiently to ensure that everything fits properly into the vehicle. Additionally, they need to consider the comfort of the passengers and visibility while driving, so the placement of the luggage must not obstruct the driver's view or make the passengers uncomfortable during the trip.

LLaVA-Instruct-158K

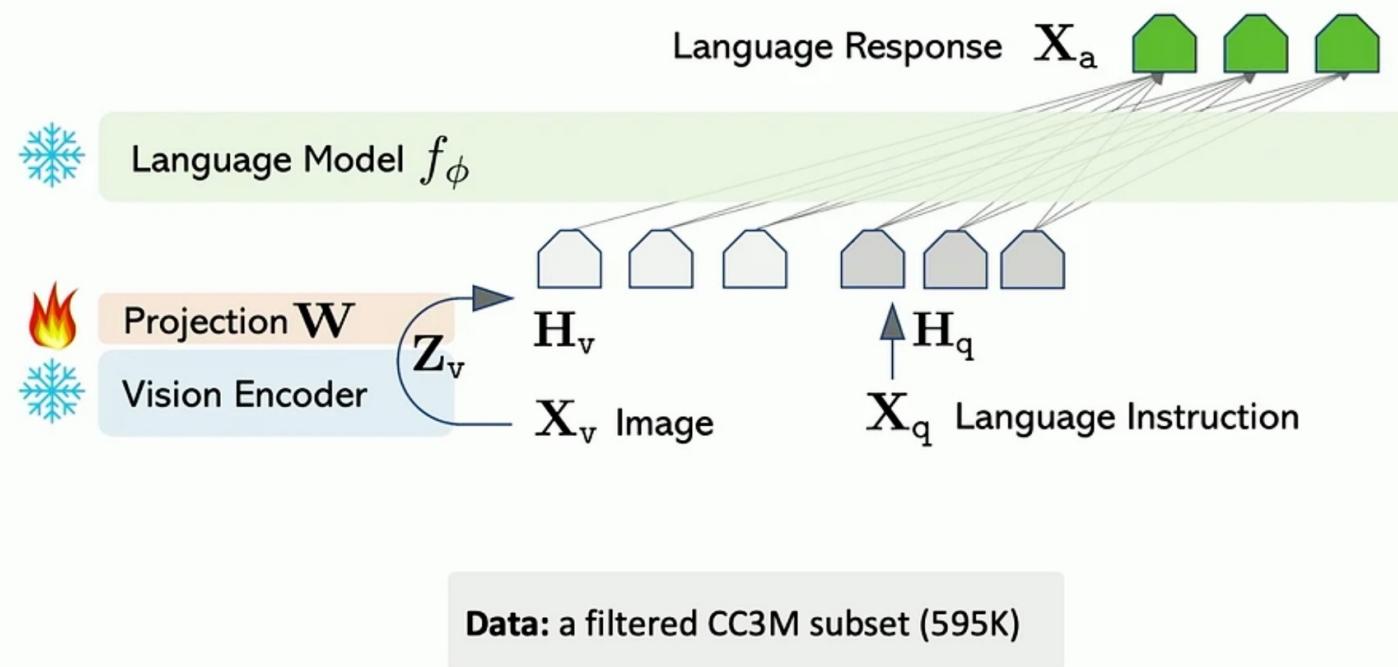
Conversation: 58K

Detailed description: 23K

Complex reasoning: 77K

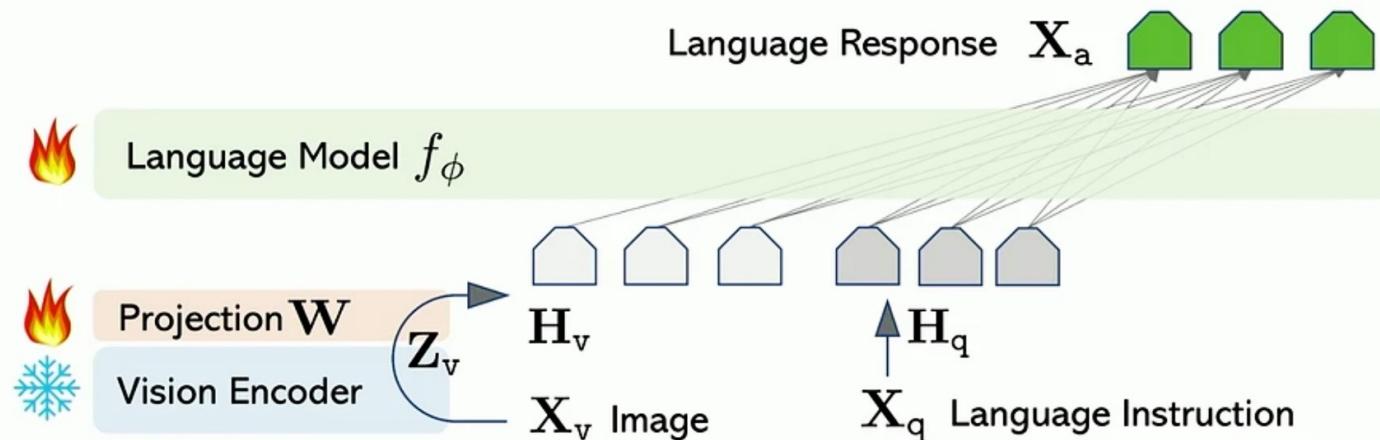
LLaVA Stage 1: Pre-training for feature alignment

- Both the visual encoder and LLM weights are frozen, while LLaVA aligns the image features H_v with the pretrained LLM word embedding using trainable parameters W (the projection matrix).
 - This stage can be understood as training a compatible visual tokenizer for the frozen LLM.



LLaVA Stage 2: Fine-tuning End-to-End

- The vision encoder is always frozen, but the pre-trained projection layer and LLM are updated with the language-image instruction-following data.
 - A visual chatbot is developed by fine-tuning on the visual instruction-tuning data.
 - LLaVA is also evaluated on the ScienceQA benchmark, where the questions are provided in the form of natural language or images.

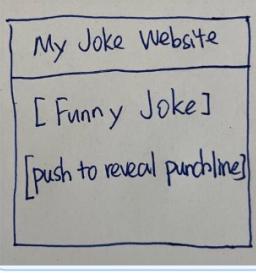


Two Tasks

- Visual Chat: LLaVA-Instruct-158K for open-ended user-oriented visual tasks.
- Science QA: Multimodal reasoning dataset for the science domain.

LLaVA Results: Recognizing visual content and generating code

User



Turn this mock-up into an interactive website using html/js, replace the joke with a real joke.

LLaVA

```
<!DOCTYPE html>
<html>
<head>
<title>My Joke Website</title>
<style>
body { background-color: white; font-family: Arial, sans-serif; }
.container { display: flex; flex-direction: column; align-items: center; justify-content: center; height: 100vh; }
h1 { font-size: 36px; margin-bottom: 20px; }
p { font-size: 20px; }
button { background-color: blue; color: white; border: none; padding: 10px 20px; font-size: 18px; cursor: pointer; }
</style>
</head>
<body>
<div class="container">
<h1>My Joke Website</h1>
<p id="punchline">Welcome to my joke website. Push the button to reveal the punchline!</p>
<button onclick="revealPunchline()">Push me!</button>
</div>
<script>
function revealPunchline() {
  document.getElementById("punchline").innerHTML = "The punchline is: 'Why was the math book sad? Because it had too many problems.'";
}
</script>
</body>
</html>
```

Place the generated HTML code into an HTML renderer to render the web page.

My Joke Website

Welcome to my joke website. Push the button to reveal the punchline!

Push me!

Rendered website

My Joke Website

The punchline is: 'Why was the math book sad? Because it had too many problems.'

Push me!

Rendered website
(clicked the button)

User



What are the meals that I can cook with these?

LLaVA

With the variety of food items stored in the refrigerator, you can prepare several meals. For example, you can create a fruit salad using the strawberries, blueberries, and carrots. Additionally, you can make a delicious smoothie using the fruits and vegetables, which would be a healthy and nutritious snack or breakfast option. The milk and eggs can be used to create various dishes, such as a milk-based smoothie, a baked casserole, or scrambled eggs. The juice can also be used to make a fresh fruit and juice mix or as a refreshing homemade popsicle. The possibilities are vast, and the meals you can create will depend on your preferences and dietary needs.

I would like to do the fruit salad. Show me the recipe.

LLaVA

Certainly! Here's a simple fruit salad recipe using the items in your refrigerator:

Ingredients:

- 2 cups fresh strawberries, sliced
- 1 cup fresh blueberries
- 1 large carrot, grated
- Juice of half a lemon
- A handful of chopped fresh parsley or mint
- A few ice cubes
- Optional: honey or maple syrup (adjust to taste)

Instructions:

1. In a large bowl, combine the sliced strawberries, blueberries, and carrot. Gently toss them together to mix.
2. Drizzle the juice of half a lemon over the fruit and vegetables to prevent them from oxidizing and add a refreshing citrus flavor.
3. Sprinkle the chopped fresh parsley or mint over the fruit and gently toss once more.
4. If desired, add a drizzle of honey or maple syrup for a touch of sweetness. Adjust the amount according to your taste preferences.
5. Cover the bowl and refrigerate the fruit salad for at least 30 minutes or up to a few hours. This allows the flavors to meld and creates a more refreshing salad.
6. Serve the fruit salad in cups or bowls, garnished with a sprinkle of fresh parsley or mint, if desired.

This fruit salad recipe is perfect for a healthy snack or side dish, and with the variety of ingredients available in your refrigerator, it should be easy to put together quickly.

User

LLaVA Results: Strong Visual Reasoning Capability

Visual input example, Extreme Ironing:



Source: <https://www.barnorama.com/wp-content/uploads/2016/12/03-Confusing-Pictures.jpg>

User
LLaVA

What is unusual about this image?

The unusual aspect of this image is a man ironing clothes on the back of a minivan or van. This is not a typical place to perform this activity, as one would usually iron clothes in a more stationary and safe location, such as a home, using a regular ironing board. The scene depicted in the image is peculiar as it involves a makeshift ironing setup on a vehicle, which can be both unsafe and unconventional. Additionally, it is not clear how the man is able to maintain balance and stability while ironing clothes in such an unstable environment.

[Start a new conversation, and clear the history]

User
LLaVA

What's happening in the scene?

The image depicts a man standing on top of a yellow SUV in a busy city street. He is holding a portable ladder, seemingly ironing clothes while standing on the vehicle. Around the scene, there are other cars, a traffic light, and a fire hydrant. Additionally, there are two other people in the vicinity, one close to the man on the car and another further away. The man on the ladder appears to be attracting attention due to his unconventional choice of ironing his clothes on top of a moving car. The city street around him is bustling with activity, adding to the unique nature of the scene.

User
GPT-4 [32]

What is unusual about this image?

The unusual thing about this image is that a man is ironing clothes on an ironing board attached to the roof of a moving taxi.

User
BLIP-2

Can you explain this meme in detail?

a man is sitting on the back of a yellow cab

Strong Visual Reasoning Capability

Example 1: Extreme Ironing

LLaVA Results: Strong Visual Reasoning Capability



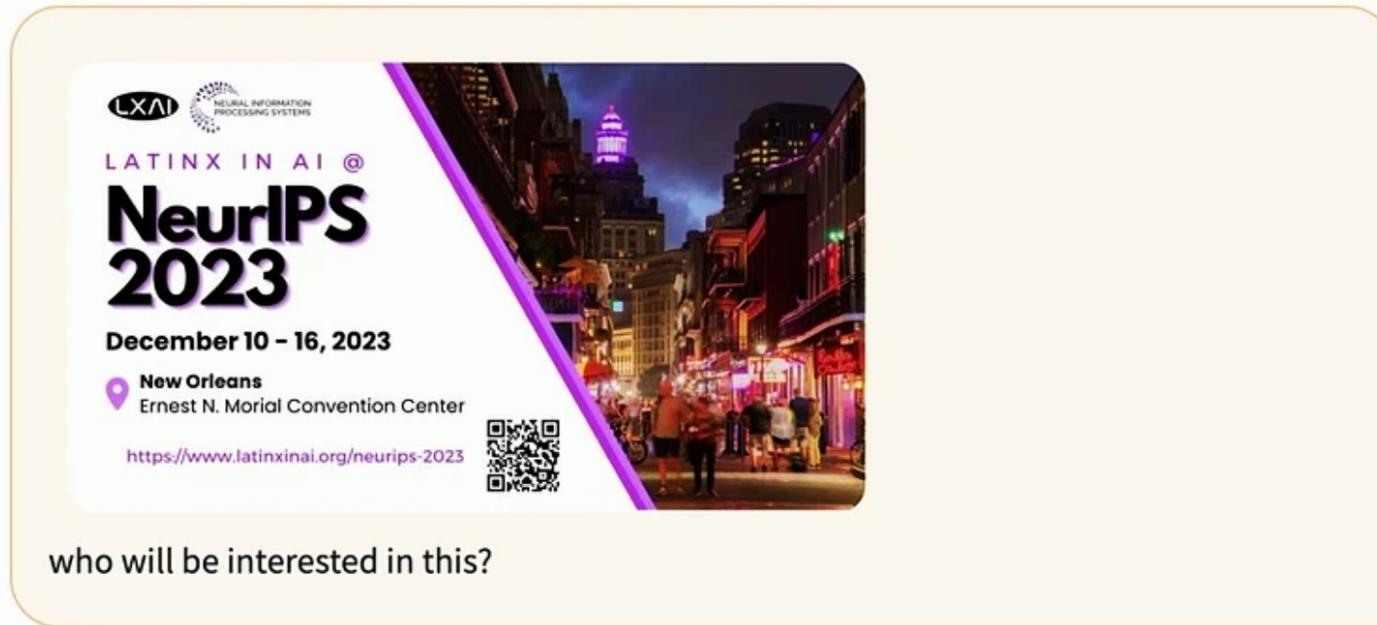
Strong Visual Reasoning Capability

Example 2: Parodied Mona Lisa

What may be the purpose of this painting?

The purpose of this painting could be to create an artistic representation of the famous painting, "The Mona Lisa," by Leonardo da Vinci. The artist has painted a dog in the same pose as the Mona Lisa, wearing a mask, and standing in front of a backdrop that resembles the original painting. This artistic interpretation may serve as a creative and humorous take on the iconic painting, drawing attention to the dog's unique expression and the contrast between the dog's appearance and the original subject. The painting could also be a tribute to the original work or a commentary on the cultural significance of the Mona Lisa.

LLaVA Results: Strong Emergent OCR Capability



Strong Emerging OCR Capability

Example: NeurIPS 2023

who will be interested in this?

People interested in **artificial intelligence, machine learning**, and related fields will be interested in this event. The image features a poster advertising the **NeurIPS 2023** conference, which is a leading conference in the field of artificial intelligence. Attendees at this conference will likely include researchers, academics, industry professionals, and enthusiasts who are passionate about advancements in AI and its applications.

LLaVA: Summary

- LLaVA integrates large language models (LLMs) with visual encoders to develop multimodal capabilities and generate contextual dialogue.
- Architecture:
 - Visual Encoder: extracts feature from images, typically using pre-trained CLIP.
 - Language Model: utilizes a powerful pretrained LLM (e.g., LLaMA) to interpret and generate textual content.
 - Projection Matrix: a learnable projection matrix that aligns visual features with text embeddings.
- High-Quality Data Generation:
 - Uses GPT-4 to create rich, diverse instruction-following datasets with 158K language-image samples.
- Two-Stage Training Process:
 - Stage 1: Pretraining for Feature Alignment
 - Stage 2: Fine-tuning End-to-End

Vision and Language Transformer Models Summary

Model	Core Focus	Architecture	Training Method	Applications
BERT and GPT	Natural Language Processing	Transformer Encoder (BERT) and Decoder (GPT)	Pretraining on large text corpora (masked tokens for BERT, autoregressive for GPT)	Text-based tasks like sentiment analysis, text generation, etc.
ViT (Vision Transformer)	Computer Vision	Transformer Encoder for image patches	Supervised training on image classification	Computer vision tasks: image classification, object recognition, etc.
CLIP	Unified Vision-Language Representation	Dual encoders (ViT for images, Transformer for text)	Contrastive learning on 400M image-text pairs	Zero-shot classification, Text-to-Image Retrieval, etc.
ViLBERT	Unified Vision and Language	Dual stream Transformer with Co-Attention	Masked multi-modal learning & alignment tasks	Visual Question Answering, Image-Text Retrieval, etc.
DALL-E	Tex-to-Image Generation	Discrete VAE with autoregressive transformer	Two-stages: VAE pretraining, autoregressive learning	Image generation from text
LLaVA	Multimodal Dialogue Generation	CLIP-based visual encoder + Language Decoder with projection	Visual instruction tuning data generated with GPT-4 assistance	Visual dialogue and reasoning