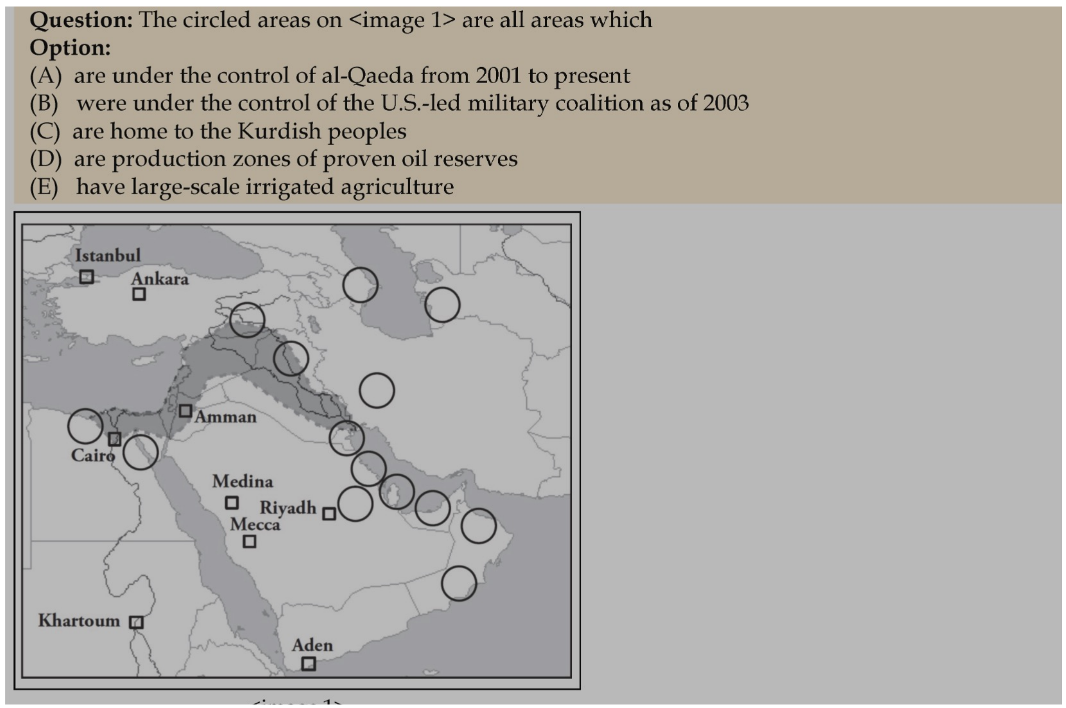
# Vision Language Model

# AGI = Artificial General Intelligence

- AGI should be able to take any forms of input and produce any forms of output.
- Forms: Audio, Video, Image, Text.
- We focus on Image for this lecture, Audio and Video will be deferred to the next lecture.

# So, what is VLM?

- VLM takes input that contains both text and images, and output texts (also can potentially output image, but that's not the main focus)



# VLM intuition:

- Standard Text-only transformer: Take an input text (like “How to feed a pig efficiently? … ”), transfer it into a sequence of tokens.
  - [182142, 5123, 99817, 52321, 477, 325, …]
- Transformer “accepts” input like a sequence of tokens.
- VLM input:
  - Here is an image <|image_1|>, tell me what is in this image.
- VLM encoder transfers the special token <|image_1|> to a sequence of tokens, which is acceptable by transformer as input.

# VLM Encoder

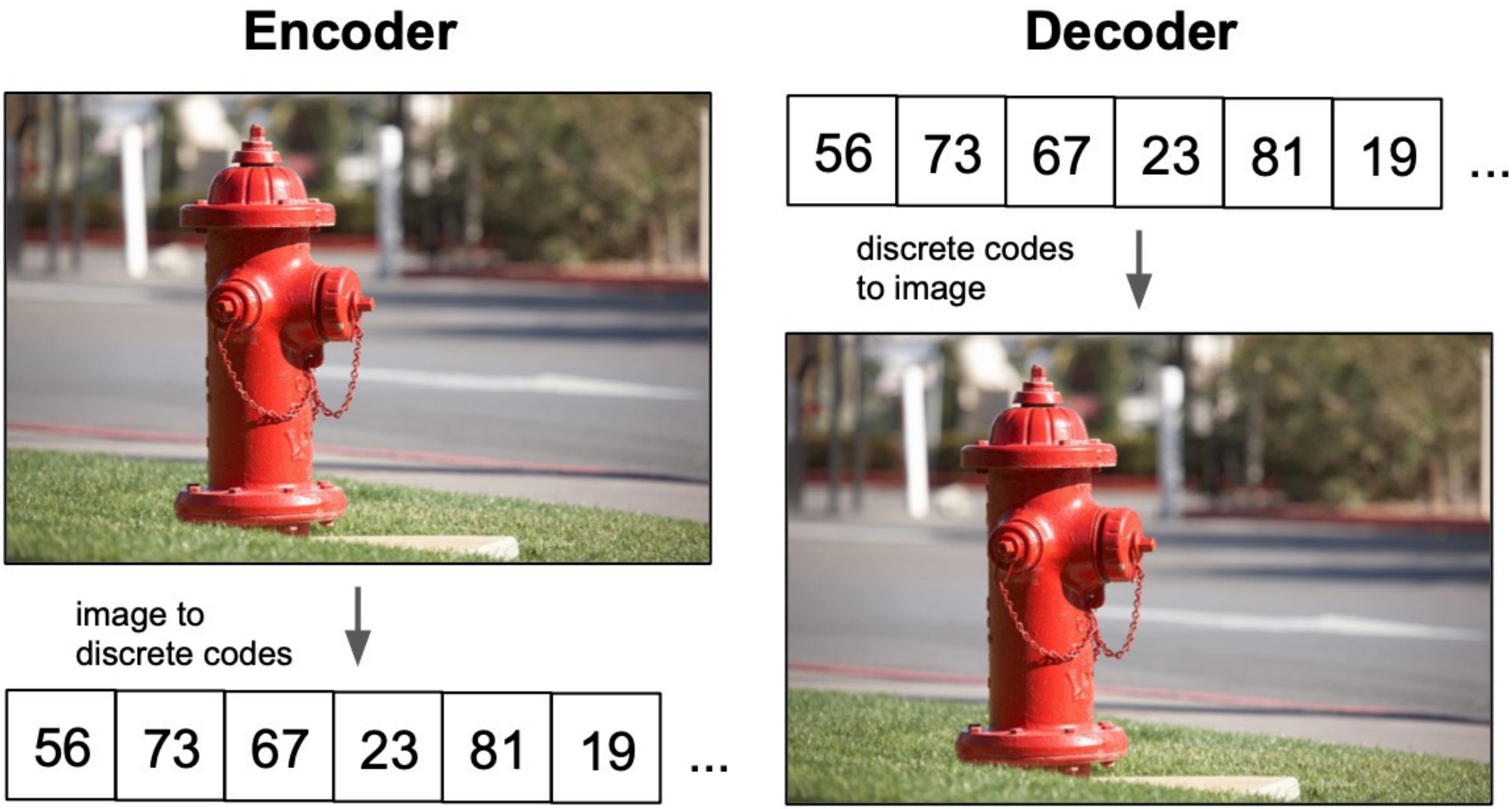
- Roughly speaking, there are two types of VLM encoders.
  - CLIP based VLM encoder (Used in GPT-V)
  - VQ-VAE based VLM encoder (Used in Gemini)

# VQ-VAE based VLM encoder

- VQ-VAE encodes a image into a sequence of tokens (integers)
- VQ-VAE-Encoder(image) = a list of integers, each integer is in [image_vocab_size] (something like 8096).
- How do we ensure that those integers are good and maintaining all the information of the original image?

# VQ-VAE based VLM encoder

- AE: Auto-Encoder.
- Roughly speaking, we want to train two models:
  - VQ-VAE-Encoder
  - VQ-VAE-Decoder
- Such that for any image,
  - VQ-VAE-Decoder(VQ-VAE-Encoder(image)) = image.
- So, we can recover the original image from the list of integers output by VQ-VAE-Encoder.


Encoder
image to
discrete codes
56 73 67 23 81 19 ...
Decoder
56 73 67 23 81 19 ...
discrete codes
to image

# Training a VQ-VAE

- Quantizing the output of standard VAE.
- VAE takes an input image, and output (a sequence of) vectors.
  - VAE(image) = vector1, vector2, …, vectork. Those vectors can take any value.
- We want to map each vector to  a vector from a finite set (e1, e2, …, en).
  - Training objective:
  - Maintain a set of vectors e1, e2, …, en.
  - For each vector vectori, map it to the argmin of $\left|\left|e_j - vector_i\right|\right|^2$ (for all j in [n]), let’s call the argmin R(i)
  - Add to the loss function: sum of $\left|\left|e_{R(i)} - vector_i\right|\right|^2$ for all i.
  - Argmin is not differentiable, but we just treat the gradient as 0.

# Using VQ-VAE in VLM

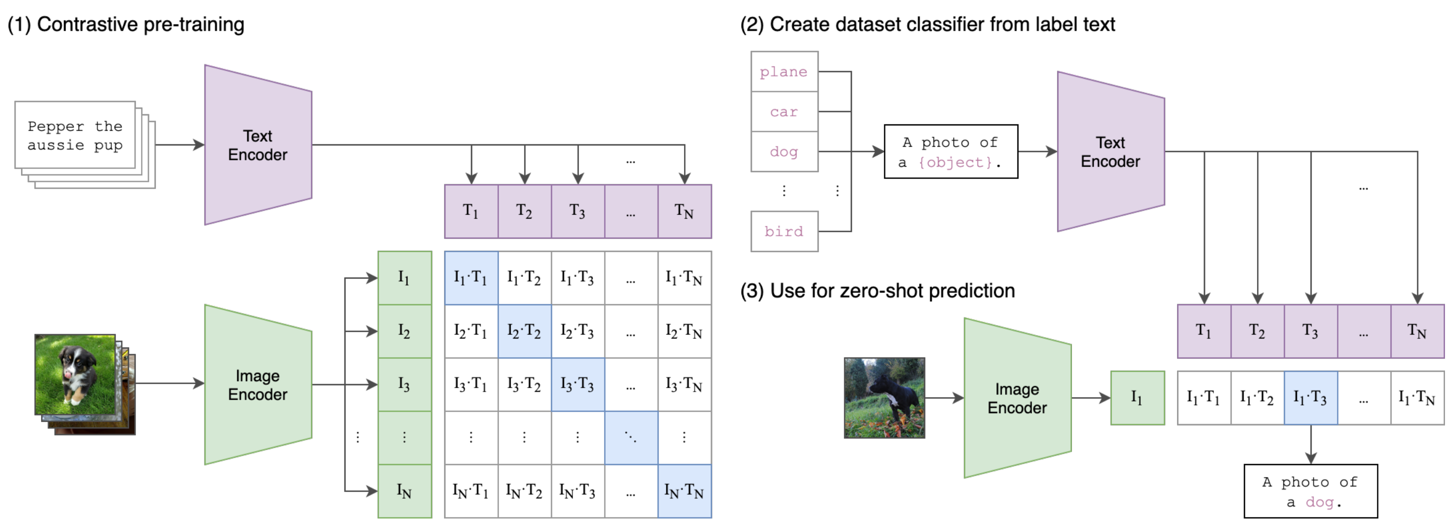
- Input to the transformer:
  - [text_token1, text_token2, …., text_tokenk, image_token1, image_token2, …, image_tokenm, text_token{k+1}, ….]
- Transformer embedding layer -> We change this to
  - Transformer Embedding Layer (Text) (WTE_T, [text_vocab_size] -> R^{emb})
  - Transformer Embedding Layer (Image) (WTE_I, [image_vocab_size] -> R^{emb})
- Then we apply WTE_T to text_tokens, and apply WTE_I to image_tokens
- Training objective: Next token prediction (loss on both image and text tokens).

# CLIP-based VLM encoder

- CLIP: Maps an image into a sequence of vectors (not tokens), where each vector is in R^{clip_dimension} (usually clip_dimension = 1024).
- Then, input to the transformer looks like
- [text_token1, text_token2, …., text_tokenk, image_vector1, image_vector2, …, image_vectorm, text_token{k+1}, ….]
- Transformer embedding layer -> We change this to
  - Transformer Embedding Layer (Text) (WTE_T, [text_vocab_size] -> R^{emb})
  - Transformer Embedding Layer (Image) (WTE_I, R^{clip_dimension} -> R^{emb})
- Training objective: Next token prediction, no loss on the image_vectors.

# Training a CLIP

- How do we make sure the sequence of vectors are good? (preserving the information of the original images?)



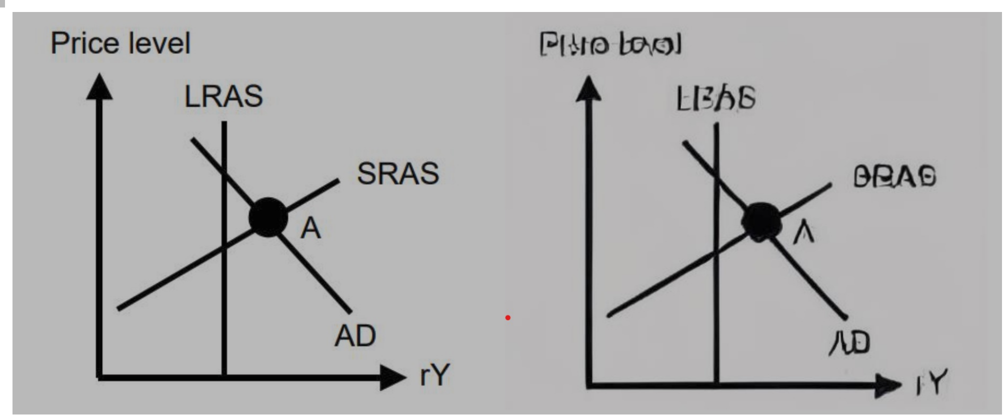
# CLIP Training Objective

- Given an image I, we can define R(I) as some augmentation of the image I (such as crop, resize, jittering, etc.)
- We also have the corresponding text label of the image I, we call it T(I)
- We want to make sure that the output of the CLIP (sequence of vectors) satisfies that
  - CLIP(R(I)) is close to Transformer(T(I)), for some Text-Only Transformer model.
  - CLIP(R(I)) is far from Transformer(T(I’)), for all other texts.
  - Maximize:
    - Exp(< CLIP(R(I)) , Transformer(T(I))>)/ E[Exp(< CLIP(R(I)) , Transformer(T(I’))>)]

# CLIP versus VQ-VAE

- VQ-VAE:
  - Can be used to generate images (loss on image tokens), support arbitrary resolution/aspect-ratio (different images will be mapped to different length of image tokens).
- CLIP:
  - It is used by OpenAI.
  - Vectors encode more information than discrete tokens, so CLIP preserves more details of the original image.

# VQVAE decoding:



# VLM: Training Data

- I don’t know what Gemini or GPT-V used exactly, here’s the common source of VLM training data.
- But the standard ones are:
  - Image Caption pairs data (such as Google Images, Bing Images)
  - Interlacing Image and Text data (standard websites, arxiv papers etc)
  - Textbook Exercises
  - ChartQA/TableQA (synthetically generated)
  - Document layout understanding/screenshot understanding (mostly human labeled)
  - OCR training data (such as PDF images to markdown)
  - Etc.

# VLM: Common Benchmarks

| Benchmark Name | Category |
|---|---|
| MMMU (Val) | Multi-discipline college-level problems |
| TextVQA (val) | Text reading on natural images |
| DocVQA (test) | Document understanding |
| ChartQA (test) | Chart understanding |
| InfographicVQA (test) | Infographic understanding |
| MathVista (testmini) | Mathematical reasoning |
| AI2D (test) | Science diagrams |
| | |
| V-star | Visual detail understanding |
| OCRBench | comprehensive OCR evaluation benchmark |

# VLM: Common Benchmarks

**Question:** <*image 1*> The region bounded by the graph as shown above. Choose an integral expression that can be used to find the area of R.

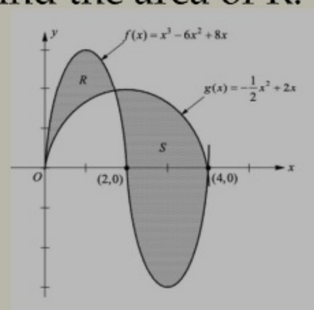
**Options:**

**(A)** $\int_0^{1.5}[f(x) - g(x)]dx$

(B) $\int_0^{1.5}[g(x) - f(x)]dx$

(C) $\int_0^{2}[f(x) - g(x)]dx$

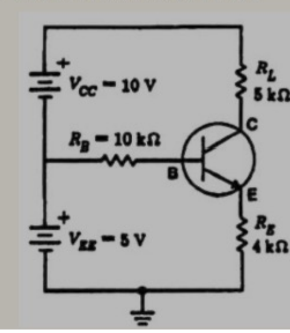
(D) $\int_0^{2}[g(x) - x(x)]dx$




**Question:** Find the VCE for the circuit shown in <*image 1*>. Neglect VBE

**Answer:** 3.75

**Explanation**: …IE = [(VEE) / (RE)] = [(5 V) / (4 k-ohm)] = 1.25 mA; VCE = VCC - IERL = 10 V - (1.25 mA) 5 k-ohm; VCE = 10 V - 6.25 V = 3.75 V



# VLM: Training Tricks

- Training VLM usually requires two phases: Pretraining phase and an Instruction Finetuning phase.
- In the pretraining phase, we take data that are “interlacing of images and texts”.
  - Bob is a very magical student <|image_1|> (image showing Bob’s grade transcript), Bob usually plays football 24/7 everyday and Bob still got A+ in all of his classes, except getting an A++++ in Elementary SpaceShip Maintaince.
- In the finetuning phase, we take data that are more QA-like.
  - <|user|> Here is an image of Bob’s transcript <|image_1|>, what score did Bob get in the class named “Advanced Chicken Cooking?” <|end|><|assistant|>Bob got A+ in this class.<|end|>

# VLM: Training Tricks

- Resolution Matters A Lot, we need the VLM encoder to support high resolution images.
  - Standard VLM uses resolution 336x336 for input images.
  - GPT-V uses 1K x 1K resolution for input images.
- Native Aspect Ratio:
  - Models like VIT only takes square images, but some image has bad aspect ratio (like an image of a formula).
  - Sora uses NaVIT (patching + 2D positional encoding).