

AIAA 5047
Responsible AI
2025 Fall

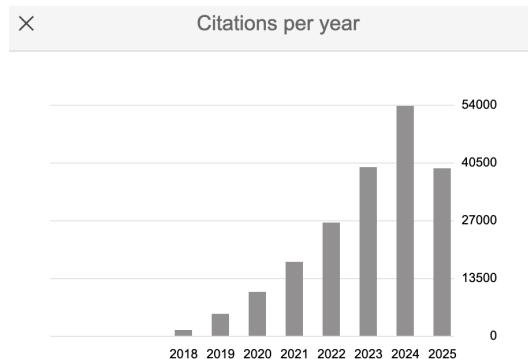
Sihong Xie, AI Thrust, Information Hub

Lecture 2

W2 201, 9-11:50 AM F

Introduction to Transformer

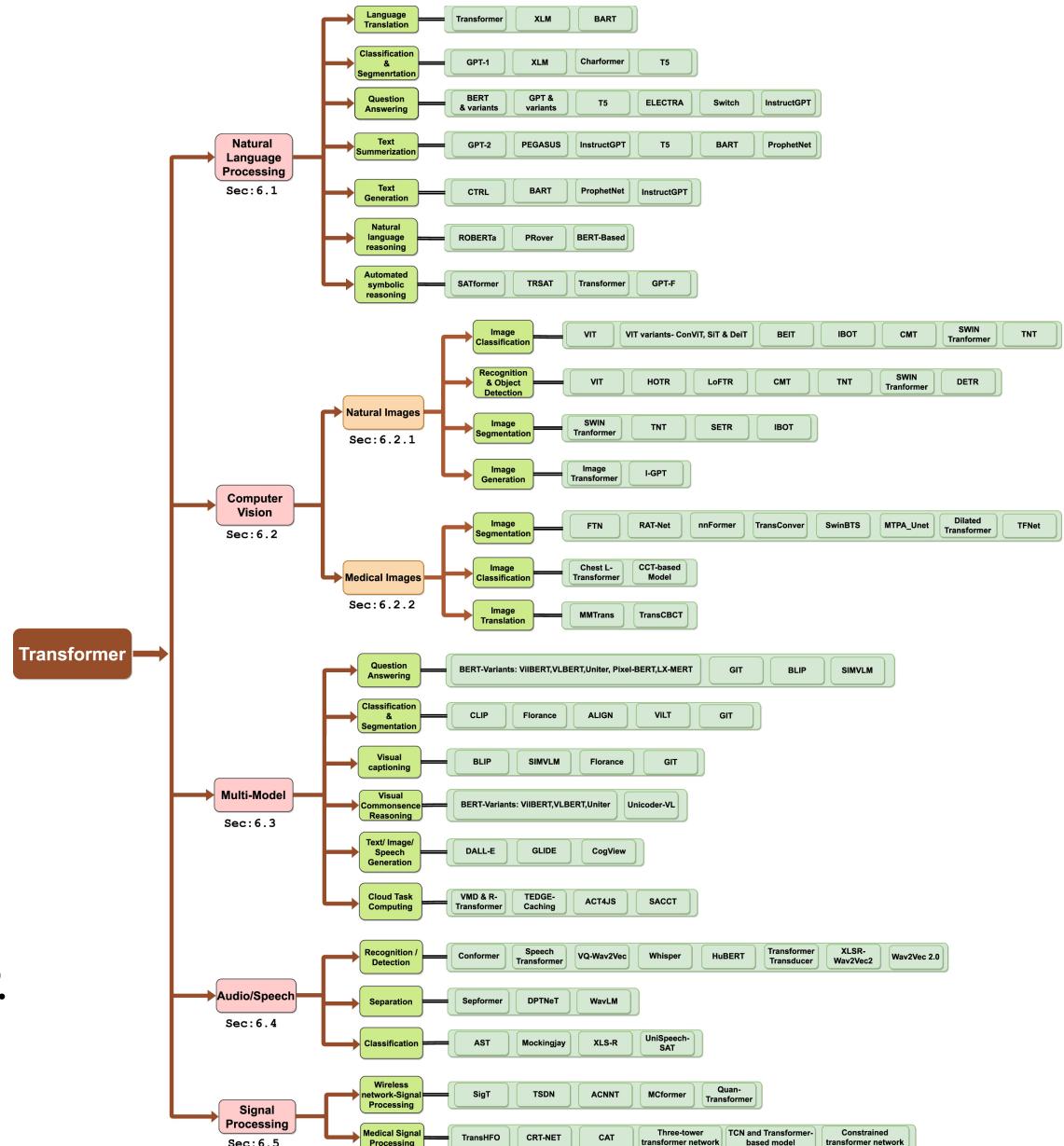
- Started from the 2017 paper "Attention is all you need".



- Extended to computer vision and robotics and data science.
- The current popular GPT-like applications are based on Transformer.

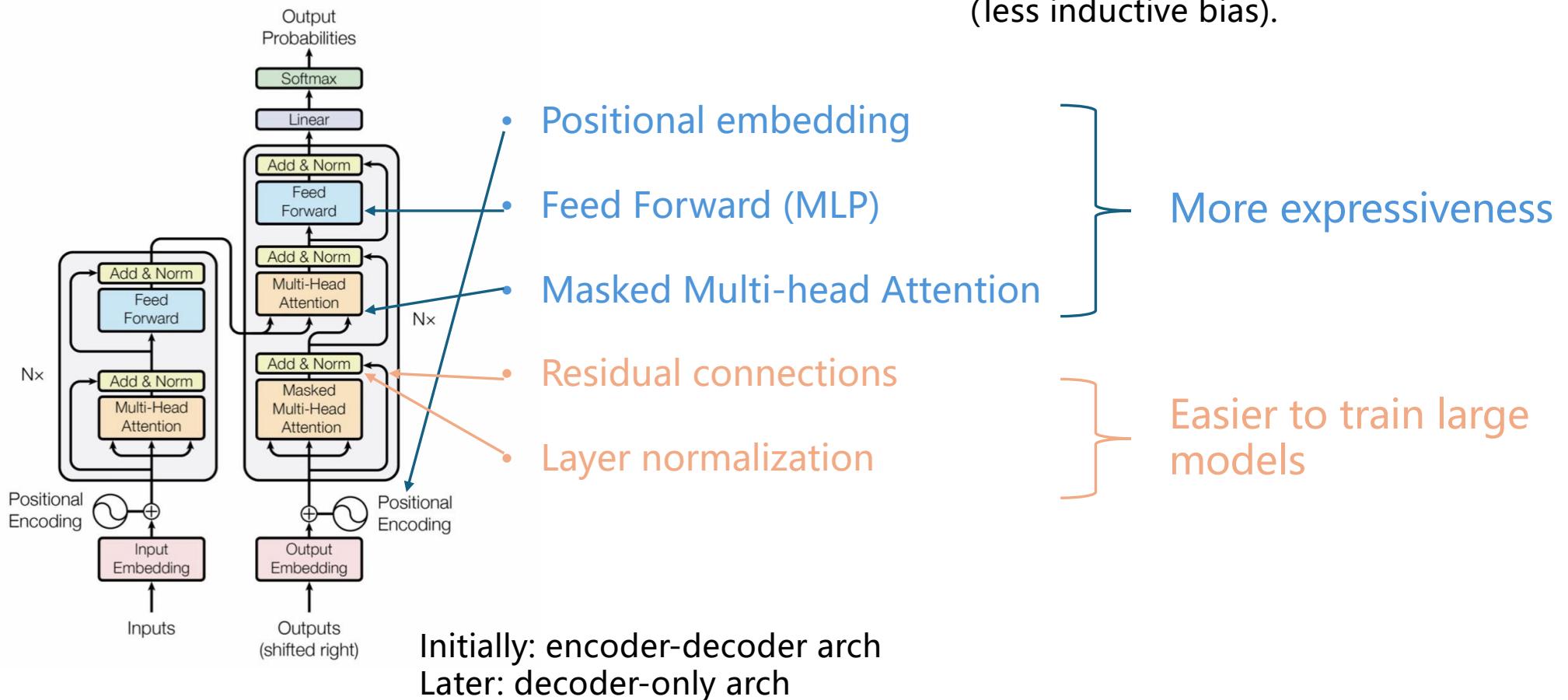
Image Source:

A comprehensive survey on applications of transformers for deep learning tasks.
Saidul Islam, etc. 2024



Transformer

- A breakdown of the transformer

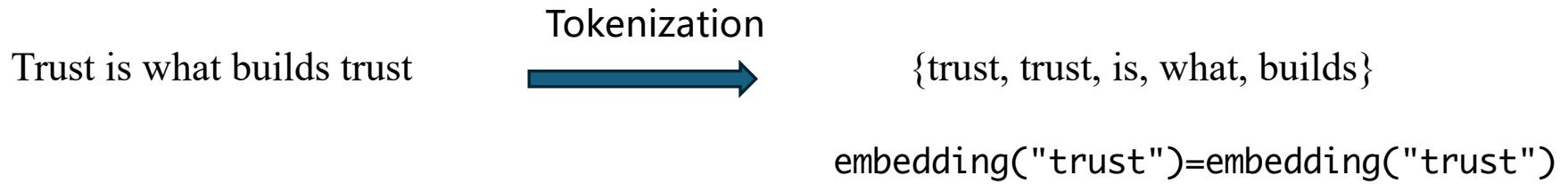


As a foundation model, it needs

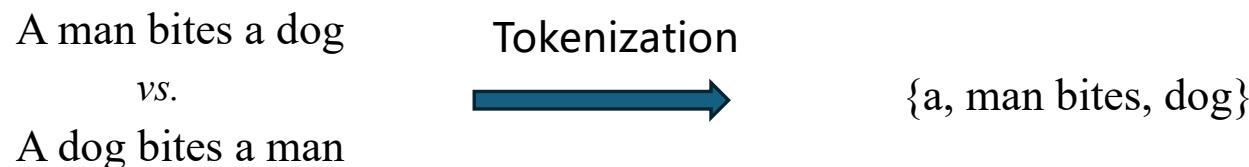
- To be a **large model** to model general and diverse linguistic patterns (transformer-based to unification).
- Consume **large data** to see all patterns (less inductive bias).

Transformer

- Positional embedding
 - To give a positional identification to a token.
 - The position of a token is important in deriving linguistic features.



- But they are different:
 - **Syntactically:** the first "trust" is subject, while the second is a object.
 - **Semantically:** the first "trust" the thing that **creates something**, while the second is the thing **being created**.
- Another example



Transformer

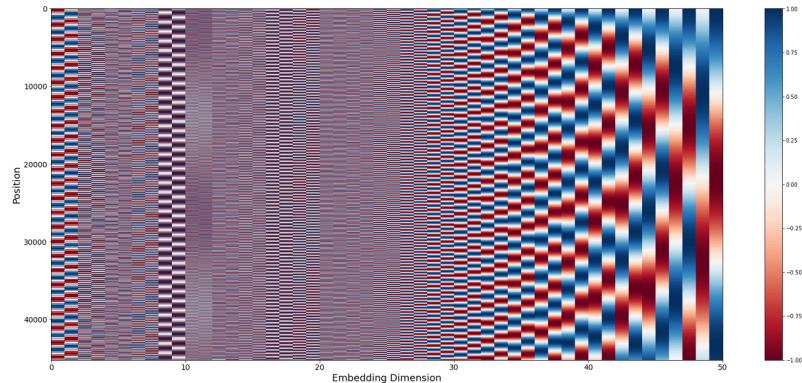
- Positional embedding

Trust is what builds trust

0 1 2 3 4

embedding("trust") + position_vector(0)
!=
embedding("trust") + position_vector(4)

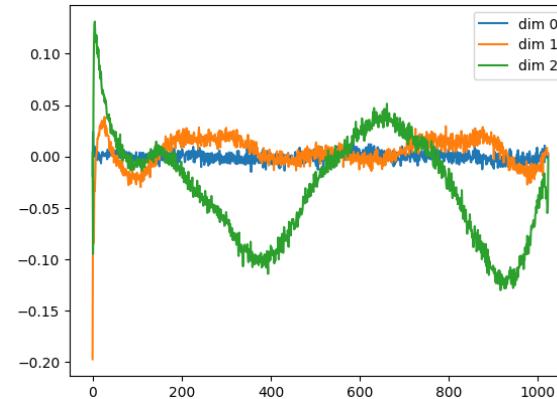
- Positional encoding (fixed)



$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

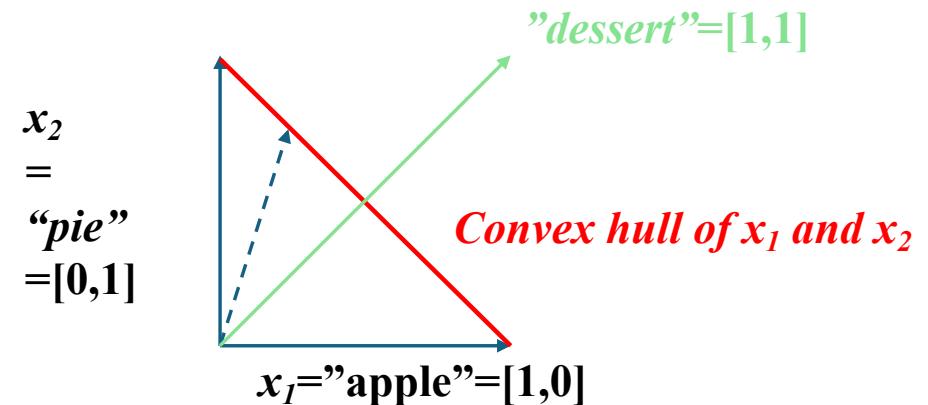
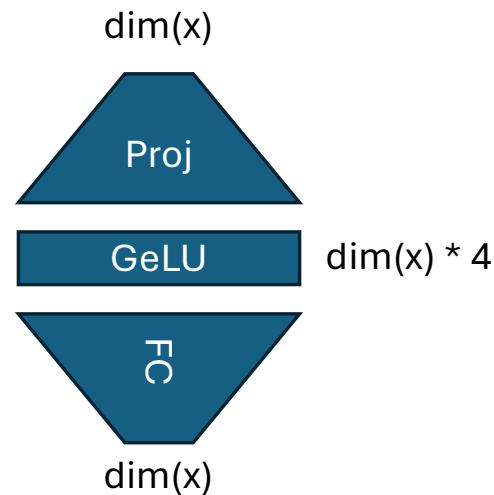
Learnable positional embedding



wpe = nn.Embedding(config.block_size, config.n_embd)

Transformer

- MLP Feed Forward Layer
 - MLP first maps x to higher dimensional space, then takes the GeLU nonlinear mapping, and finally projects back to $\text{dim}(x)$.
- Why it is necessary?
 - The self-attention layer can only find linear combination of value vectors.
 - What if some non-linearity is needed.



Transformer

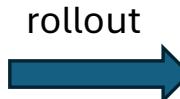
- The central of Transformer: attention mechanism
 - To address the gradient propagation issues of RNN/LSTM.

"I grew up in **France**, where I spent many years surrounded by my family and friends, so I speak fluent **French**."



- Why it is hard to learn the dependency?
 - If the model makes a wrong prediction "**English**" at the end, it is possible that the training data have more frequent "**fluent English**" occurrences (short-range dep).
 - The model forward passes are more influenced by the recent word such as "fluent", while the far-away word "France" has its semantic vector modified by all the other words in the middle and the many multiplication of the weight matrix of RNN.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

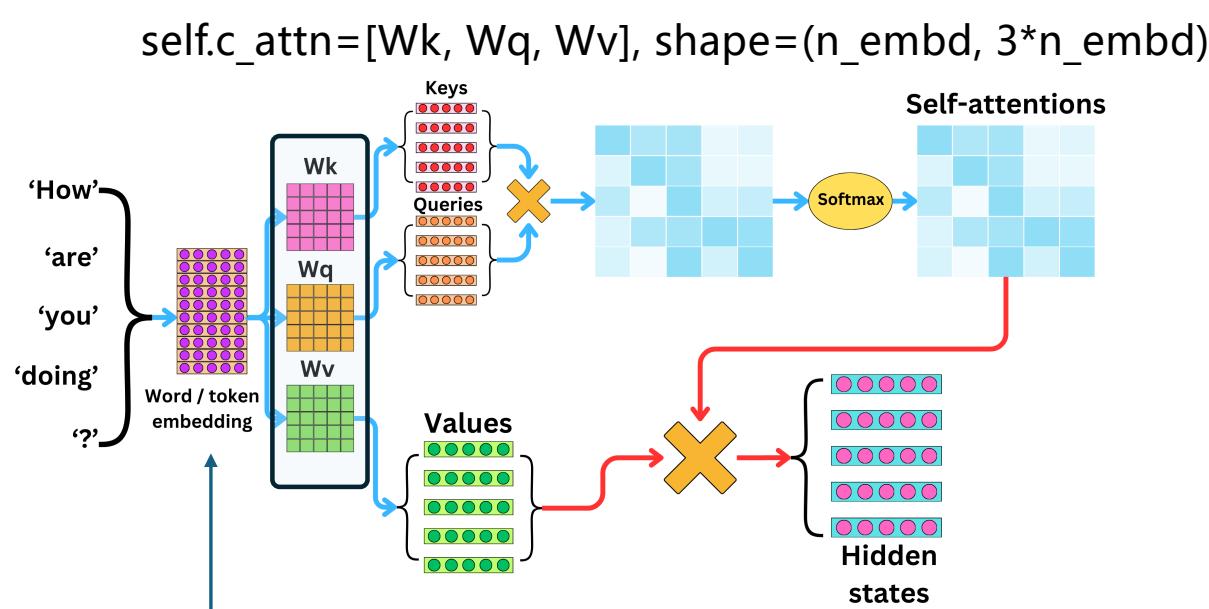


$$y_t = W_{hy}h_t$$

$$h_t = \tanh(W_{hh}\tanh(W_{hh}h_{t-2} + w_{xh}x_{t-2}) + w_{xh}x_{t-1})$$

Transformer

```
# calculate query, key, values for all heads in batch and move head forward to be the batch dim
q, k, v = self.c_attn(x).split(self.n_embd, dim=2)
k = k.view(B, T, self.n_head, C // self.n_head).transpose(1, 2) # (B, nh, T, hs)
q = q.view(B, T, self.n_head, C // self.n_head).transpose(1, 2) # (B, nh, T, hs)
v = v.view(B, T, self.n_head, C // self.n_head).transpose(1, 2) # (B, nh, T, hs)
```



x.shape=[Batch_size, Token_len, n_embd]

C = nh * hs=n_embd
k.shape[B, T, nh * hs]
view ↓
k.shape[B, T, nh, hs]
transpose ↓
k.shape[B, nh, T, hs]

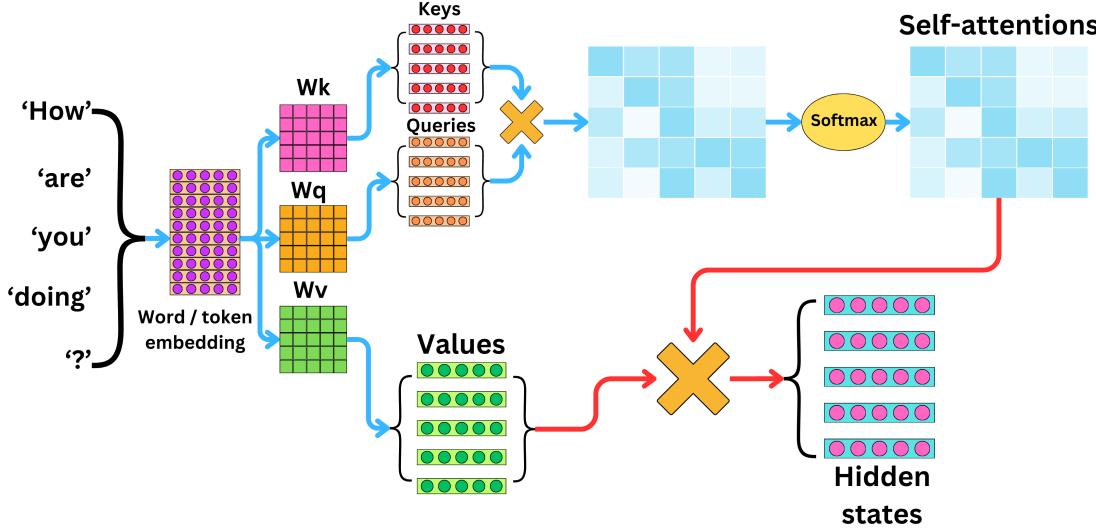
Transformer

```

att = (q @ k.transpose(-2, -1)) * (1.0 / math.sqrt(k.size(-1)))
att = att.masked_fill(self.bias[:, :, :T, :T] == 0, float('-inf'))
att = F.softmax(att, dim=-1)
    
```

To prevent the inner products from inflating the scales.

$q.shape = [B, nh, T, hs]$
 $k.transpose(-2, -1).shape = [B, nh, hs, T]$
 $(q @ k).shape = [B, nh, T, T]$



```

>>> import torch
>>> a=torch.tensor([1,2,3,float('-inf'), float('-inf'), float('-inf')])
>>> F.softmax(a)
tensor([0.0900, 0.2447, 0.6652, 0.0000, 0.0000, 0.0000])
    
```

$\text{self.bias}[:, :, :T, :T]$ has the upper triangle = 0

Causal Attention: don't look into the future

Attention weight between the tokens corresponding to "Life" and "short"

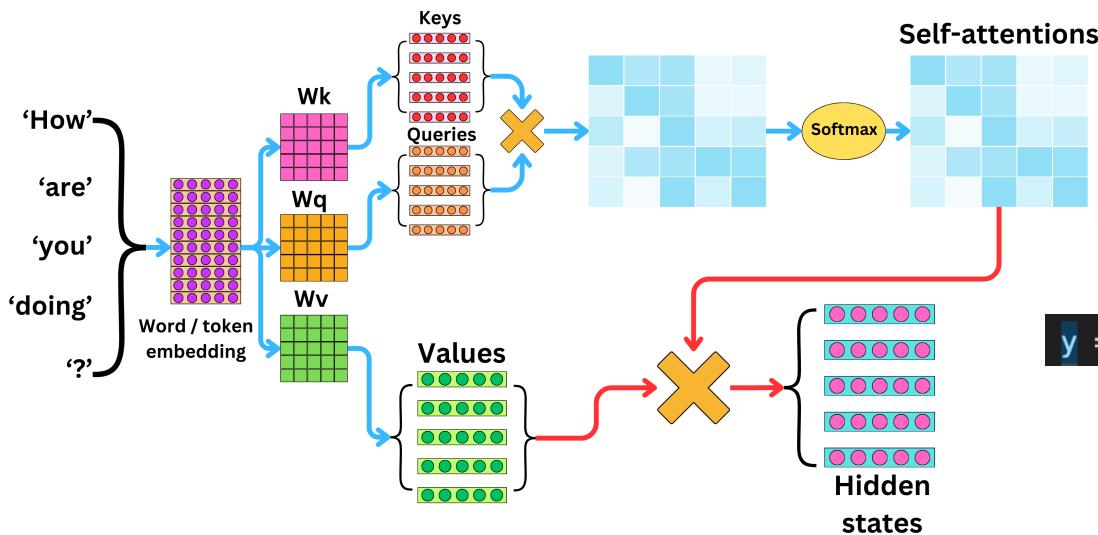
	Life	is	short	eat	desert	first
Life	0.17	0.13	0.18	0.16	0.15	0.18
is	0.03	0.68	0.02	0.08	0.14	0.02
short	0.19	0.06	0.25	0.14	0.11	0.23
eat	0.15	0.21	0.14	0.16	0.17	0.14
desert	0.13	0.27	0.11	0.16	0.18	0.12
first	0.19	0.02	0.31	0.11	0.07	0.27

In the row for corresponding to "Life", mask out all words that come after "Life"

	Life	is	short	eat	desert	first
Life	0.17	0.13	0.18	0.16	0.15	0.18
is	0.03	0.68	0.02	0.08	0.14	0.02
short	0.19	0.06	0.25	0.14	0.11	0.23
eat	0.15	0.21	0.14	0.16	0.17	0.14
desert	0.13	0.27	0.11	0.16	0.18	0.12
first	0.19	0.02	0.31	0.11	0.07	0.27

Attention weights calculated in the previous "Self-Attention" section

Transformer



$\text{att.shape} = [\text{B}, \text{nh}, \text{T}, \text{T}]$

$\text{v.shape} = [\text{B}, \text{nh}, \text{T}, \text{hs}]$

$(\text{att} @ \text{v}).\text{shape} = [\text{B}, \text{nh}, \text{T}, \text{hs}]$

$y = \text{att} @ \text{v} \ # (\text{B}, \text{nh}, \text{T}, \text{T}) \times (\text{B}, \text{nh}, \text{T}, \text{hs}) \rightarrow (\text{B}, \text{nh}, \text{T}, \text{hs})$

```
y = y.transpose(1, 2).contiguous().view(B, T, C) # re-assemble all head outputs side by side
```

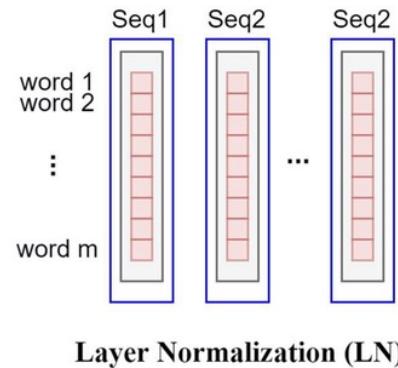


Transformer

- Residual connections
 - $x = \text{MSA}(x) + x$ and $x = \text{MLP}(x) + x$
- Intuitions:
 - At any layer, if a mapping is unnecessary for a token, just let that mapping to be zero and push the input to the next layer.
 - Mapping from [9, 11, 19, 21] to [10, 12, 18, 20] is hard, but from [9, 11, 19, 21] to the residual [-1, -1, 1, 1] is much easier.
- Why it is necessary?
 - Allow information to directly forward to the next level without MSA or MLP (which are just *learned* to be the zero mapping).
 - Allow gradients (of loss function of supervision) to directly backprop to the earlier layers, without too many multiplications and eventual gradient explosion or vanishing.

Transformer

- Layer normalization



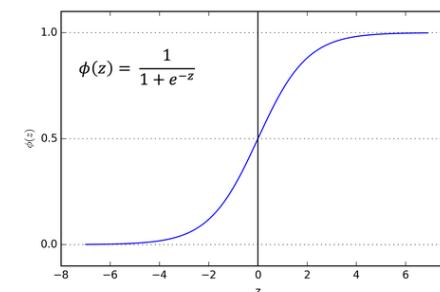
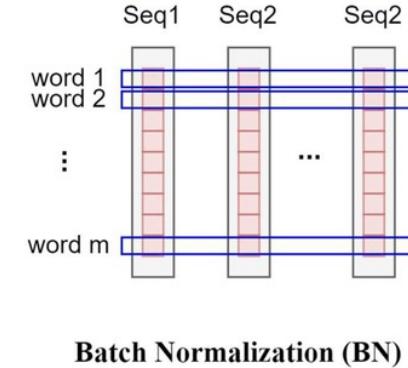
- Intuitions:

- For all neurons at the same layer, the “scale” of their activations should remain about the same as the input.
- After going through many layers, the scale of activations should remain about the same.

- Why it is necessary?

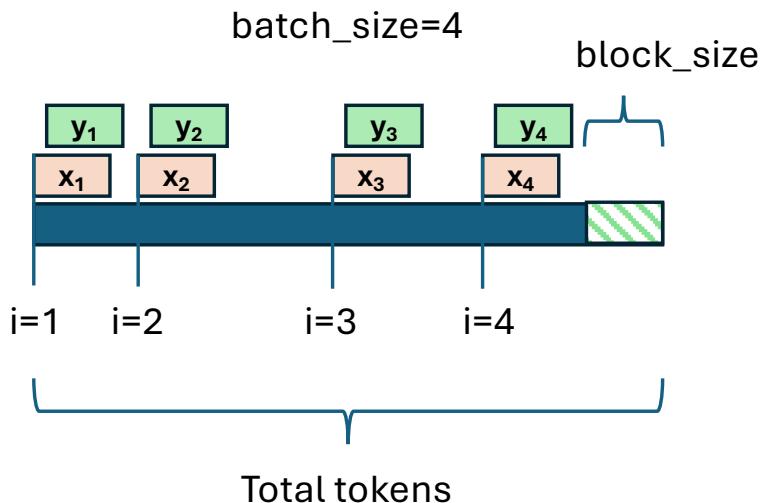
- If several layers enlarge or shrink the activation scales dramatically, there is the so-called “saturation of softmax” and consequently the “vanishing gradient” issue.

```
>>> a=torch.tensor([1.,2.,3.])
>>> b=torch.tensor([10.,20.,30.])
>>> softmax(a)
tensor([0.0900, 0.2447, 0.6652])
>>> softmax(b)
tensor([2.0611e-09, 4.5398e-05, 9.9995e-01])
```



Transformer

- Training
 - Prepare training and validation data
 - Load raw texts.
 - Use an existing encoder to turn text token into indices.
 - Save to binary files.
 - Sample a batch: randomly



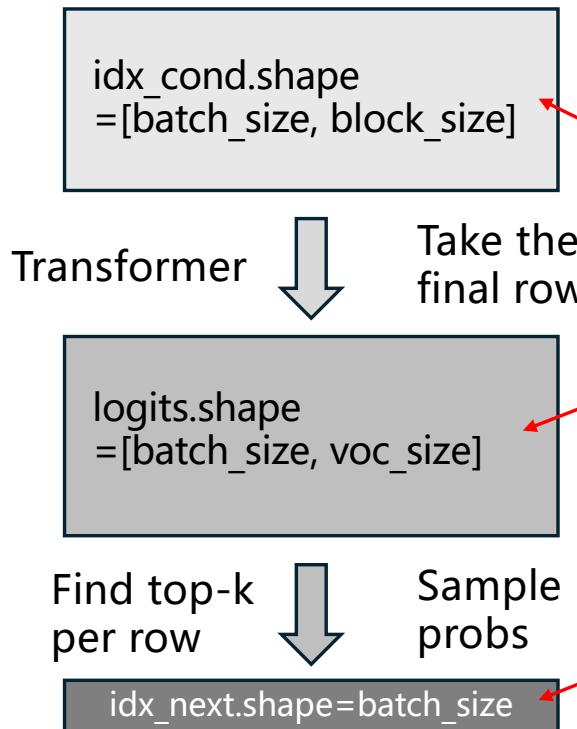
```
with open(input_file_path, 'r', encoding='utf-8') as f:  
    data = f.read()  
n = len(data)  
train_data = data[:int(n*0.9)]  
val_data = data[int(n*0.9):]  
  
# encode with tiktoken gpt2 bpe  
enc = tiktoken.get_encoding("gpt2")  
train_ids = enc.encode Ordinary(train_data)  
val_ids = enc.encode Ordinary(val_data)
```

Load part (not all) of the file to memory.

```
# poor man's data loader  
data_dir = os.path.join('data', dataset)  
def get_batch(split):  
    # We recreate np.memmap every batch to avoid a memory leak, as per  
    # https://stackoverflow.com/questions/45132940/numpy-memmap-memory-usage-want-to-iterate-once/61472122#61472122  
    if split == 'train':  
        data = np.memmap(os.path.join(data_dir, 'train.bin'), dtype=np.uint16, mode='r')  
    else:  
        data = np.memmap(os.path.join(data_dir, 'val.bin'), dtype=np.uint16, mode='r')  
    ix = torch.randint(len(data) - block_size, (batch_size,))  
    x = torch.stack([torch.from_numpy((data[i:i+block_size]).astype(np.int64)) for i in ix])  
    y = torch.stack([torch.from_numpy((data[i+1:i+1+block_size]).astype(np.int64)) for i in ix])  
    if device_type == 'cuda':  
        # pin arrays x,y, which allows us to move them to GPU asynchronously (non_blocking=True)  
        x, y = x.pin_memory().to(device, non_blocking=True), y.pin_memory().to(device, non_blocking=True)  
    else:  
        x, y = x.to(device), y.to(device)  
    return x, y
```

Transformer

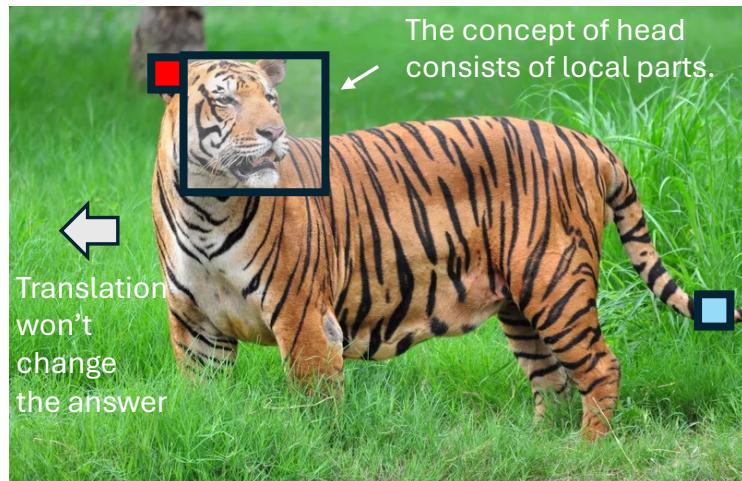
- Inference



```
@torch.no_grad()
def generate(self, idx, max_new_tokens, temperature=1.0, top_k=None):
    """
    Take a conditioning sequence of indices idx (LongTensor of shape (b,t)) and complete
    the sequence max_new_tokens times, feeding the predictions back into the model each time.
    Most likely you'll want to make sure to be in model.eval() mode of operation for this.
    """
    for _ in range(max_new_tokens):
        # if the sequence context is growing too long we must crop it at block_size
        idx_cond = idx if idx.size(1) <= self.config.block_size else idx[:, -self.config.block_size:]
        # forward the model to get the logits for the index in the sequence
        logits, _ = self(idx_cond)
        # pluck the logits at the final step and scale by desired temperature
        logits = logits[:, -1, :] / temperature
        # optionally crop the logits to only the top k options
        if top_k is not None:
            v, _ = torch.topk(logits, min(top_k, logits.size(-1)))
            logits[logits < v[:, [-1]]] = -float('Inf')
        # apply softmax to convert logits to (normalized) probabilities
        probs = F.softmax(logits, dim=-1)
        # sample from the distribution
        idx_next = torch.multinomial(probs, num_samples=1)
        # append sampled index to the running sequence and continue
        idx = torch.cat((idx, idx_next), dim=1)
    return idx
```

ViT: vision Transformer

- Motivations
 - CNN has difficulty in modeling long-range dependencies on images well.



Can we give up CNN architecture?
Can the success of transformer in
NLP be transferred to computer vision?

There is a risk of needing more data
as the inductive bias, locality and
translational invariance, will be dropped.

■ ■ Convolutional kernel (window)

Query: “Does the tiger have both head and tail?”

To answer this, need to gather information from both head and tail areas.

The horizontal distance between head and tail = 800 pixels

Window size = 40 x 40 pixels

With a stride length=1 pixel and without pooling, it takes

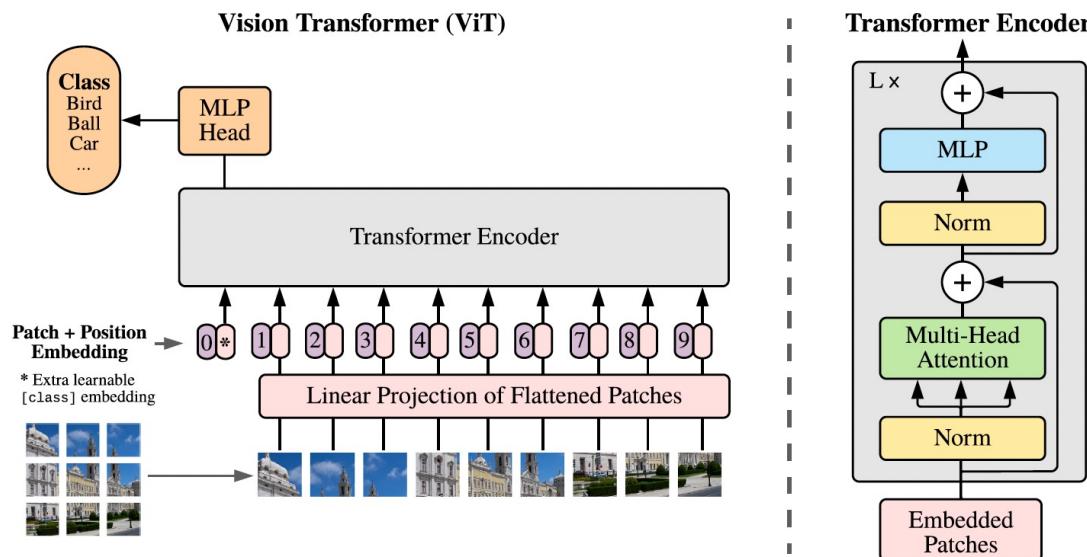
$(800-1)/(40-1)=20$ layers of convolution to connect the head and tail.

(how to calculate this: imagine a 1D array of length 800,
each 1D kernel reaches to the farthest neuron in its receptive field
to extend its receptive field of size (40-1)).

Deeper networks are harder to optimize smoothly,
due to gradient explosion or vanishing. Many techniques,
such as normalization, residual connection and dropout, are invented
to make the CNN deeper, but the problem is not solved.

ViT: vision Transformer

- Take the idea of Transformer to computer vision
 - Not the first work using attention mechanism on images^[1,2].
 - Not the first work using a purely transformer architecture^[3].
 - The first to train pure transformer on large-scale image datasets.



- Make a $H \times W \times C$ image into $N \times (P \times P) \times C$ patches
- Position embedding is important for image understanding (man in car vs. man by car)
- Since the first layer, a patch can reach to any other patch through Multi-Head Self-Attention, thereby capturing long-range dependencies.
- Image classification with a [CLS] token.

[1] Xiaolong Wang, etc.. Non-local neural networks. CVPR, 2018.

[2] Nicolas Carion, etc.. End-to-end object detection with transformers. ECCV, 2020.

[3] Prajit Ramachandran, etc. Stand-alone self-attention in vision models. NeurIPS, 2019.

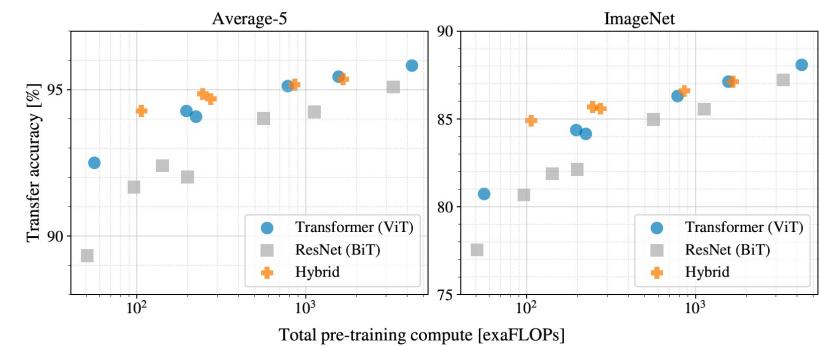
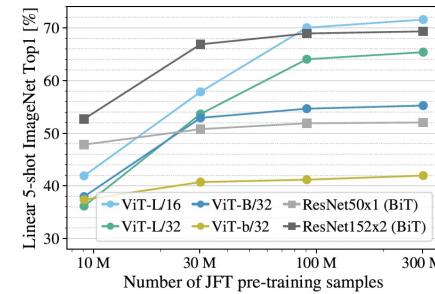
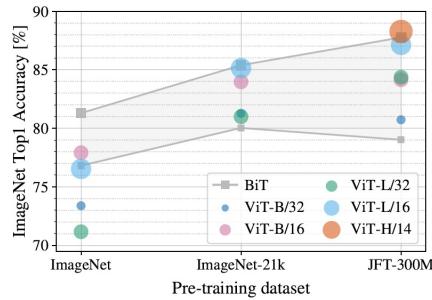
ViT: vision Transformer

Test data	Bigger model	More data	ResNet		
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	88.55 ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	90.72 ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	99.50 ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	—
CIFAR-100	94.55 ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	—
Oxford-IIIT Pets	97.56 ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	—
Oxford Flowers-102	99.68 ± 0.02	99.74 ± 0.00	99.61 ± 0.02	99.63 ± 0.03	—
VTAB (19 tasks)	77.63 ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	—
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Less computation

Scaling law in CV:

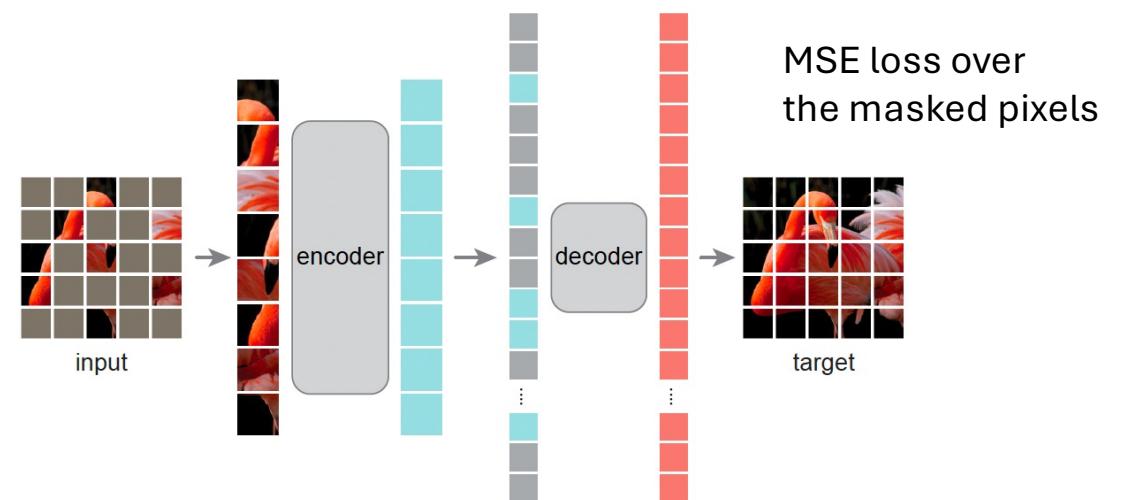
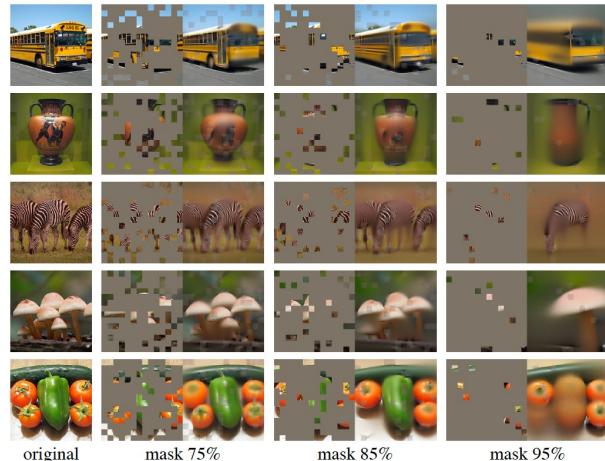
With enough data,
inductive bias can
be dropped.



With a medium-size image dataset (ImageNet), not as good as ResNet.
With a much larger dataset (Google JFM-300M), beats SOTA

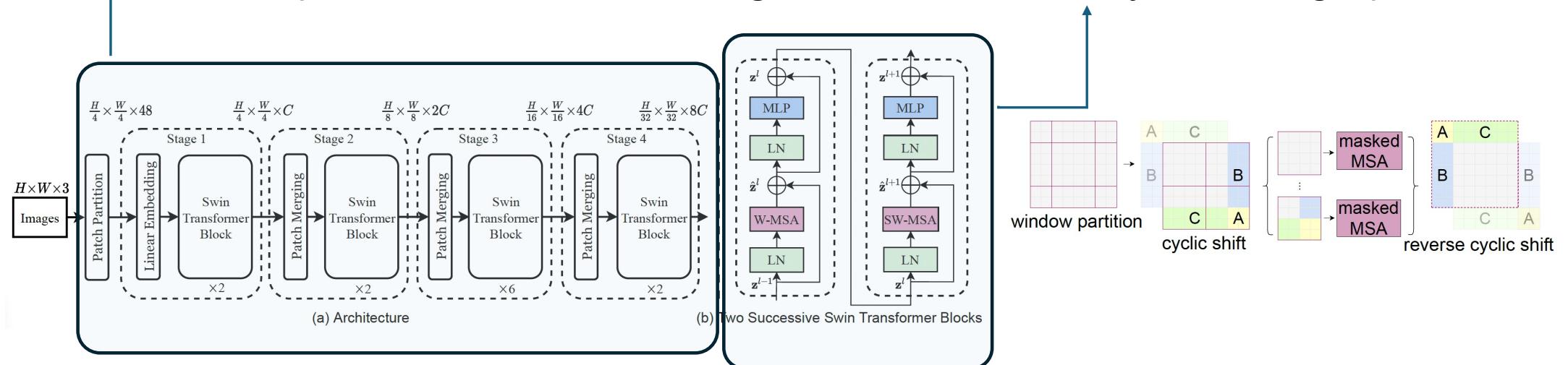
ViT variants: MAE self-supervised pre-training

- Motivation
 - Can self-supervision works for pre-training ViT?
 - Autoregressive or masked autoencoding are successful for GPT and BERT, resp.
 - Image vs. Languages
 - Each word has higher information density than a missing image patch.
 - Decoder of image needs to predict many pixels and thus more critical.
- Solution
 - Very high % of masked patches: increase information density and reduce redundancy.



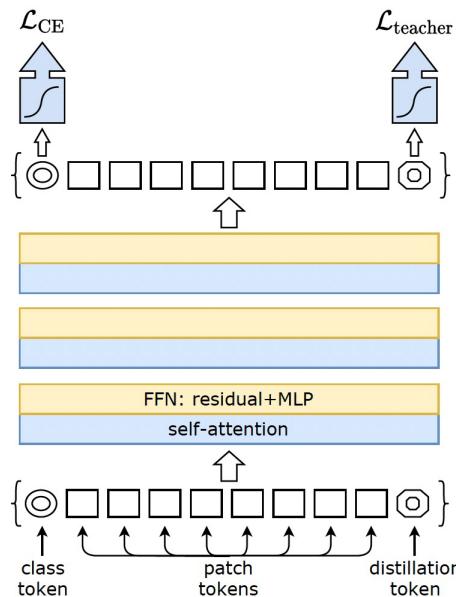
ViT variants: Swin Transformer

- Motivations
 - ViT uses a fixed patch size, and cannot be adapted to different object size.
 - What if the boundary of a patch cuts on an object?
- Solution
 - Let the patch size varies across layers: as small as 2x2 and as large as 32x32.
 - Can detect small and large objects
 - Can make object segmentation.
 - Let the patches shift around the image: able to cover one object in a single patch



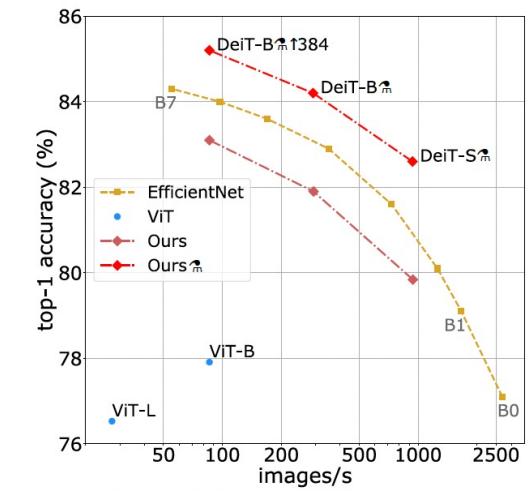
ViT variants: DeiT

- Motivations
 - ViT uses 300m private labeled image for training and its model size is large.
 - Can we use only ImageNet-1k (publicly available) and cheaper GPU?
- Solution
 - Same ViT architecture except additional distillation token and loss function.
 - To mimic a teacher model which is a CNN model (maybe the inductive bias therein).



$$\mathcal{L}_{global} = (1 - \lambda)\mathcal{L}_{CE}(\psi(Z_s), y) + \lambda\tau^2\text{KL}(\psi(Z_s/\tau), \psi(Z_t/\tau)).$$

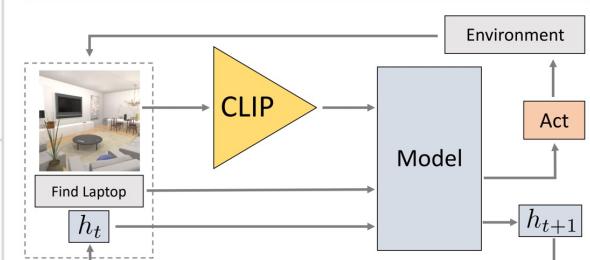
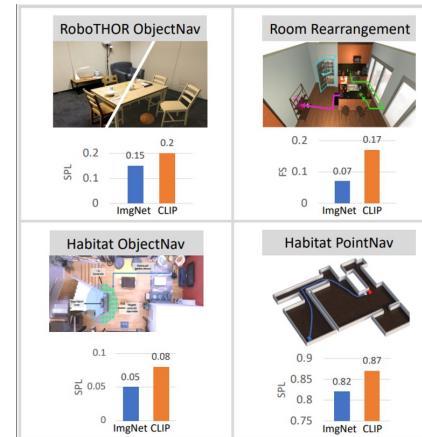
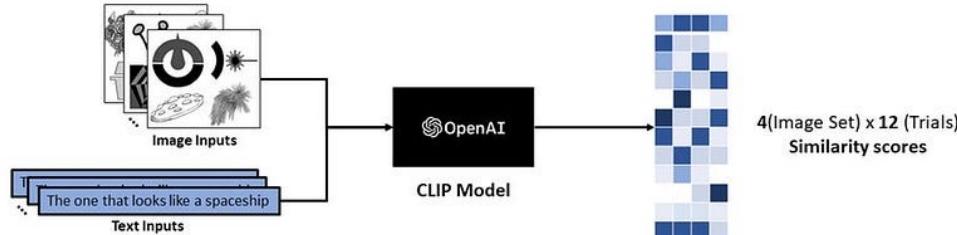
DeiT: method ↓	supervision		ImageNet top-1 (%)			
	label	teacher	Ti 224	S 224	B 224	B↑384
no distillation	✓	✗	72.2	79.8	81.8	83.1
usual distillation	✗	soft	72.2	79.8	81.8	83.2
hard distillation	✗	hard	74.3	80.9	83.0	84.0
class embedding	✓	hard	73.9	80.9	83.0	84.2
distil. embedding	✓	hard	74.6	81.1	83.1	84.4
DeiT \ddagger : class+distil.	✓	hard	74.5	81.2	83.4	84.5



CLIP (Contrastive Language-Image Pre-training)

- Motivation

- ViT has one label per image. Utilize weaker supervision: (image, text) pairs; no segmentation mask or bounding box, no image class labels.
- Understanding the correspondence between image and text modalities.
- Allow open-world classification/recognition...
- Applications: multi-modal QA, robotics (VLM and VLA).



- Intuitions:

- If we can pair an image with the correct describing texts, and a piece of text with the image that the text is describing, then the model may learn to relate the contents in the image to the keywords in the text.

Source: <https://blog.lancedb.com/multi-modal-ai-made-easy-with-lancedb-clip-5aaf8801c939/>
Simple but Effective: CLIP Embeddings for Embodied AI

CLIP

ResNet-50 and -101

```

if isinstance(vision_layers, (tuple, list)):
    vision_heads = vision_width * 32 // 64
    self.visual = ModifiedResNet(
        layers=vision_layers,
        output_dim=embed_dim,
        heads=vision_heads,
        input_resolution=image_resolution,
        width=vision_width
)
else: ViT-B/32, ViT-B/16, ViT-L/14
    vision_heads = vision_width // 64
    self.visual = VisionTransformer(
        input_resolution=image_resolution,
        patch_size=vision_patch_size,
        width=vision_width,
        layers=vision_layers,
        heads=vision_heads,
        output_dim=embed_dim
)

```

```

# cosine similarity as logits
logit_scale = self.logit_scale.exp()
logits_per_image = logit_scale * image_features @ text_features.t()
logits_per_text = logits_per_image.t()

# shape = [global_batch_size, global_batch_size]
return logits_per_image, logits_per_text

```

```

def encode_text(self, text):
    x = self.token_embedding(text).type(self.dtype) # [batch_size, n_ctx, d_model]

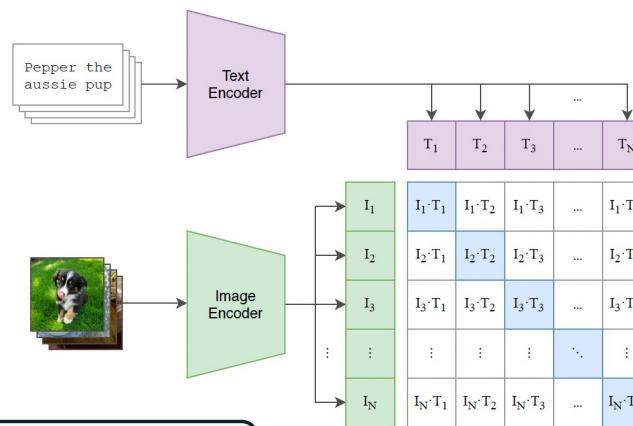
    x = x + self.positional_embedding.type(self.dtype)
    x = x.permute(1, 0, 2) # NLD -> LND
    x = self.transformer(x)
    x = x.permute(1, 0, 2) # LND -> NLD
    x = self.ln_final(x).type(self.dtype)

    # x.shape = [batch_size, n_ctx, transformer.width]
    # take features from the eot embedding (eot_token is the highest number in each sequence)
    x = x[torch.arange(x.shape[0]), text.argmax(dim=-1)] @ self.text_projection

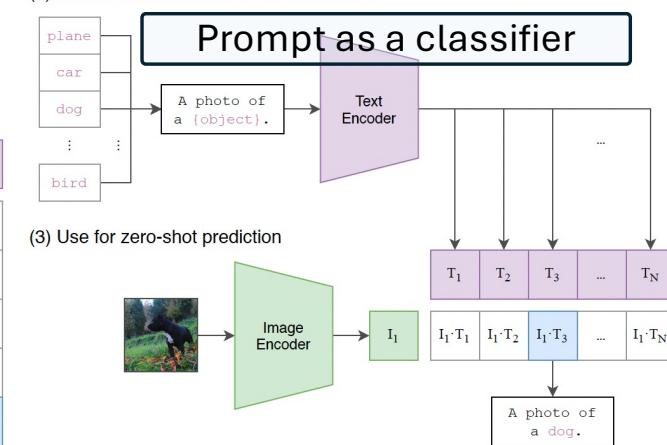
    return x

```

(1) Contrastive pre-training



(2) Create dataset classifier from label text



$$L_{i2t} = -\log(\exp(sim(i, t_i)/\tau) / \sum_j \exp(sim(i, t_j)/\tau))$$

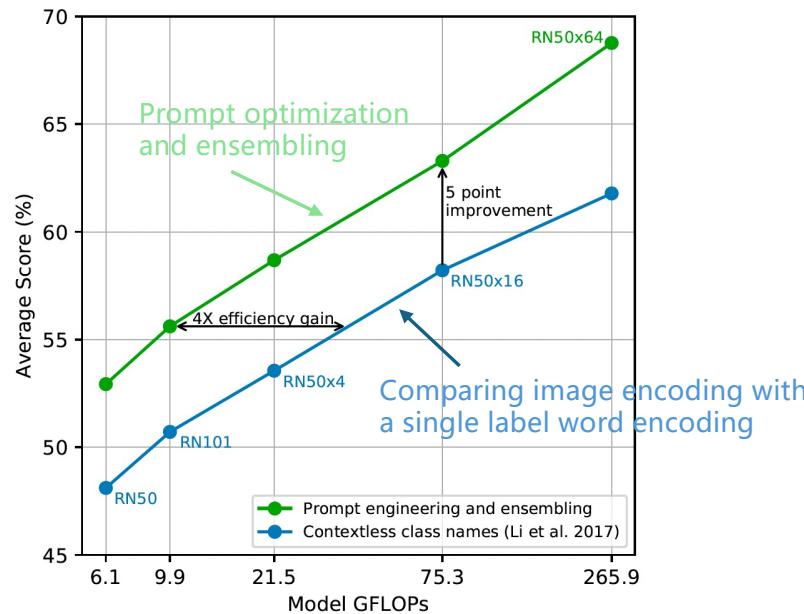
$$L_{InfoNCE} = (L_{i2t} + L_{t2i}) / 2$$

CLIP: zero-shot performance

Image classification datasets not used in pre-training

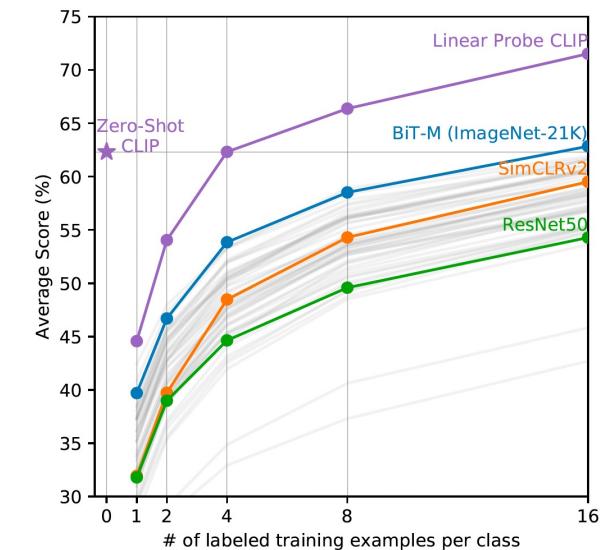
	aYahoo	ImageNet	SUN
Visual N-Grams	72.4	11.5	23.0
CLIP	98.4	76.2	58.5

The importance of prompt optimization: create prompts like "*A photo of a big {label}*" and "*A photo of a small {label}*" and ensemble the results of 80 such prompts.

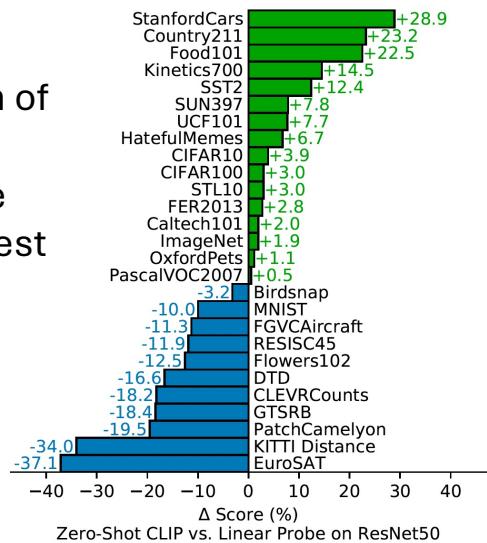


A similar work appears simultaneously with larger training set:
Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. ICML 2021

On average, zero-shot CLIP is comparable to ResNet50 trained on 21k labeled images.

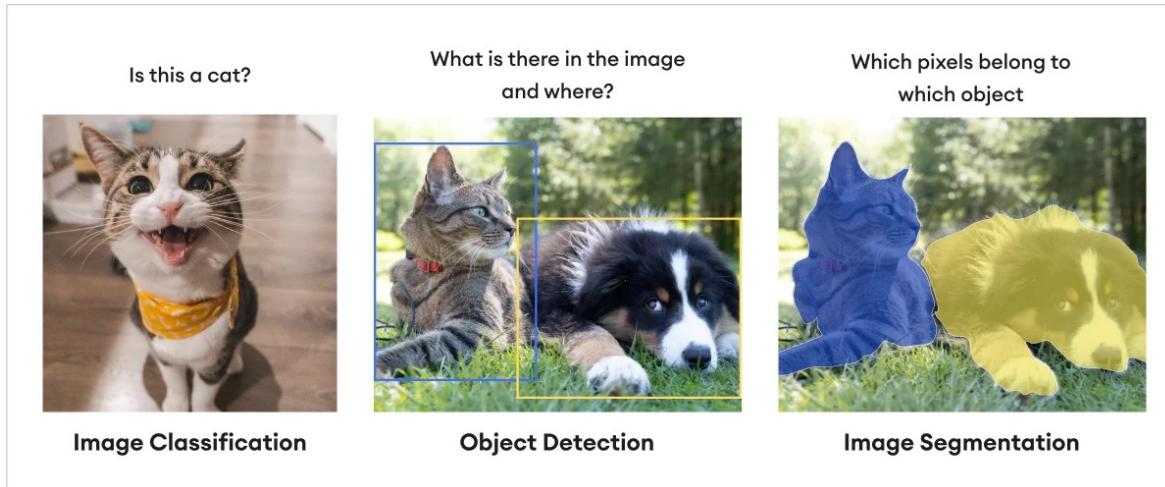


A breakdown of the average performance over the 27 test datasets.



SAM: Segment Anything Model

- Motivation
 - What is segmentation?



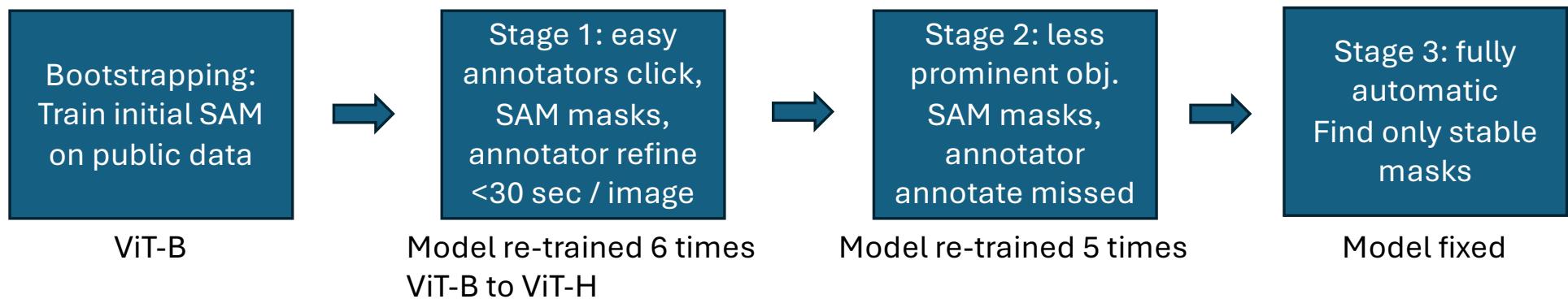
Segmentation is harder than image classification or object detection:

- More details on the pixel level, so that each pixel requires a label.
- Ambiguity: what's the boundary between the cat's fur and the fuzzy grass?
- More computational costly: a model has to predict on each pixel.
- Providing masks is very hard and time-consuming: fine details, non-regular shapes, ambiguity.

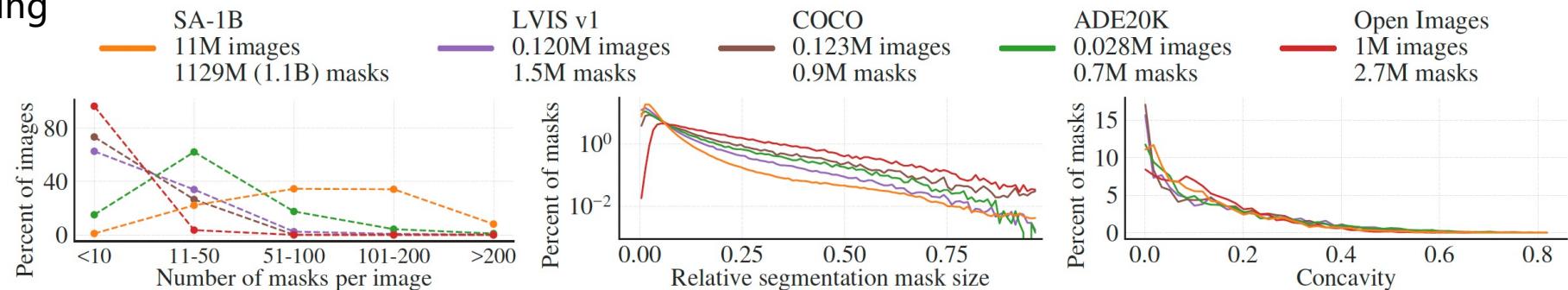
- There is sharp conflict between “more data” and “labeling cost”.
- Need a foundation model for image segmentation so that it can do zero-shot segmentation (rather than collecting many more masked images for each new class), or be quickly fine-tuned to special tasks (with only a small number of labeled data).

SAM: Segment Anything Model

- First question: how to construct a large-scale segmentation mask dataset?
 - Before SAM, the largest one is OpenImage. SAM has 11x more images and 400x more masks.
 - Better to have an automatic tool to assist the labeling.

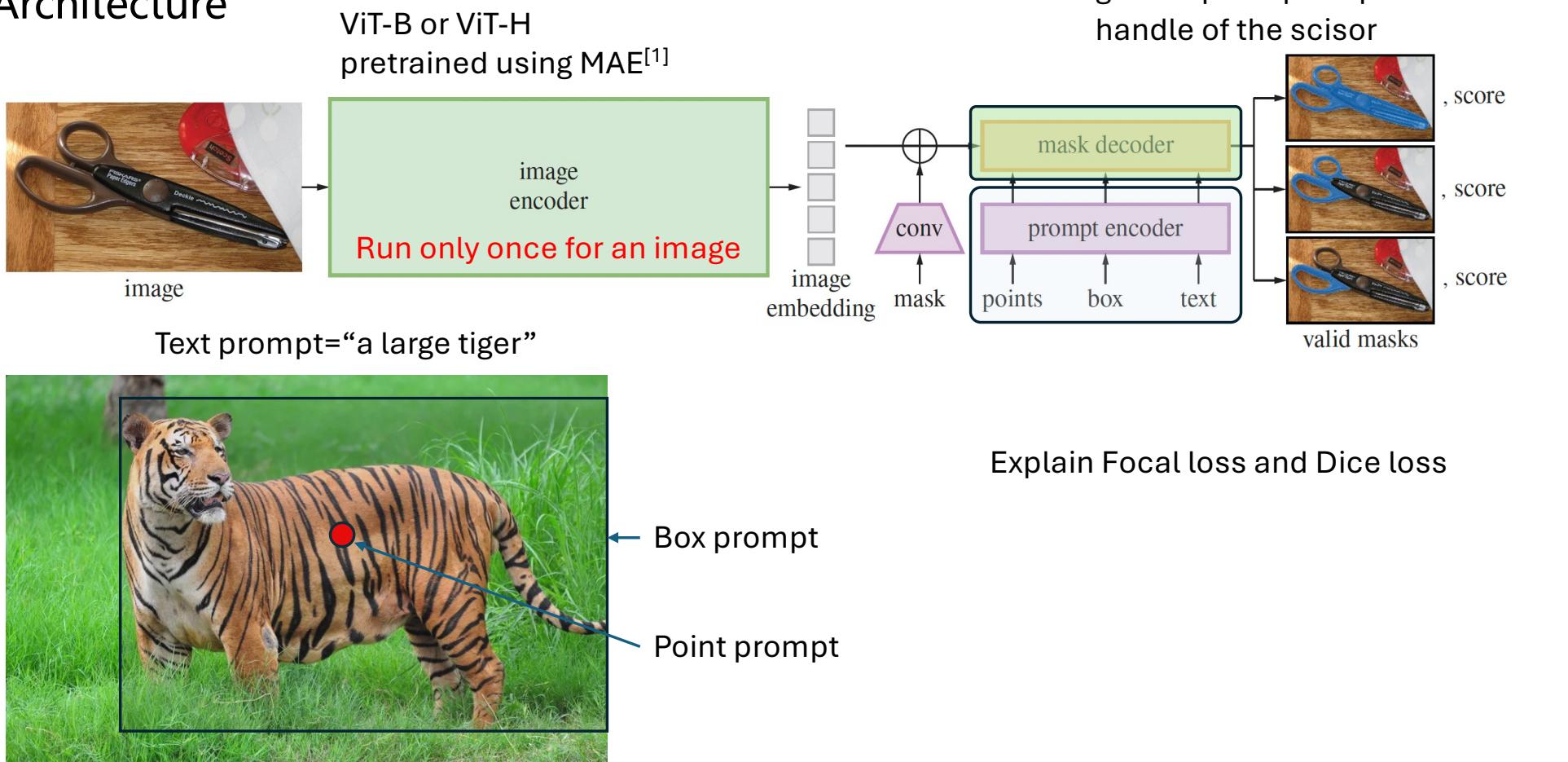


The resulting
SA-1B
dataset



SAM: Segment Anything Model

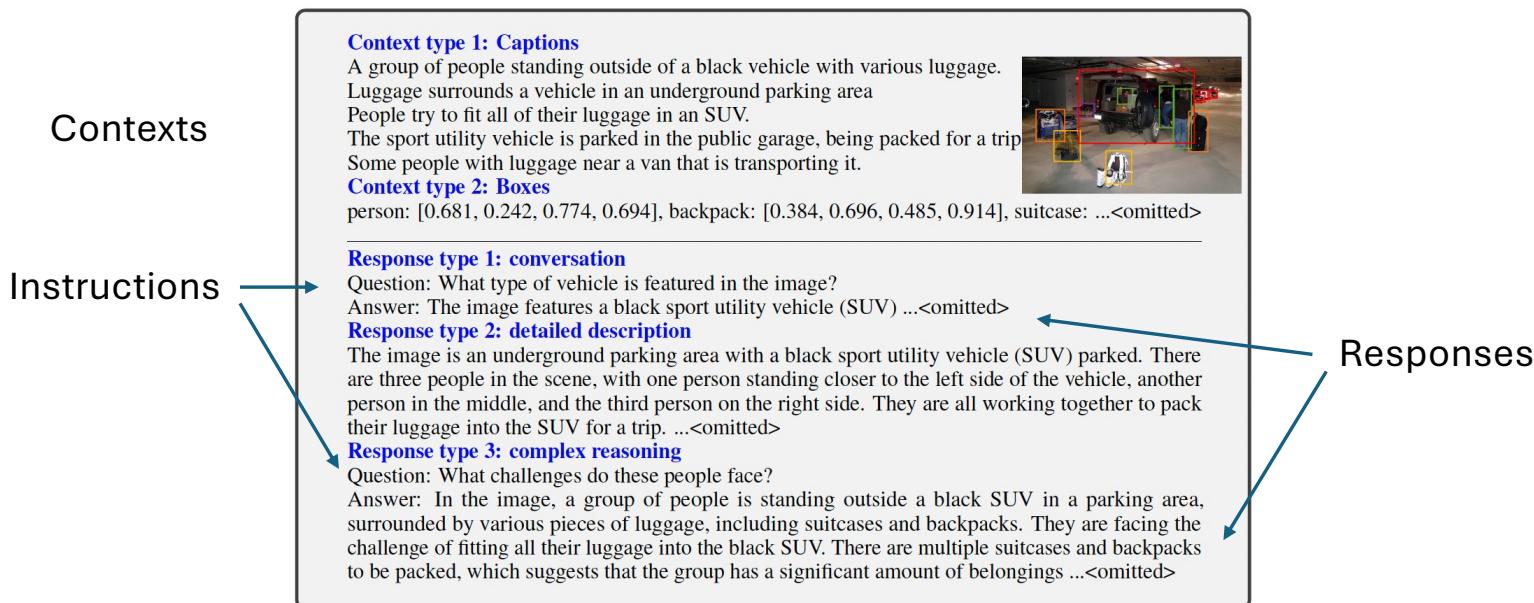
- Architecture



[1] Kaiming He, etc.. Masked autoencoders are scalable vision learners. CVPR, 2022.

LLaVA: going beyond matching

- Motivation
 - Multi-modal models are constrained to single tasks, such as image recognition, object detection, etc. [one for one]
 - On the other hand, LLM can do many things such as translation, summarization, reasoning, etc. [one for all]
 - How can we make multi-modal models to follow user instructions to conduct diverse kinds of tasks, including **generative** ones?



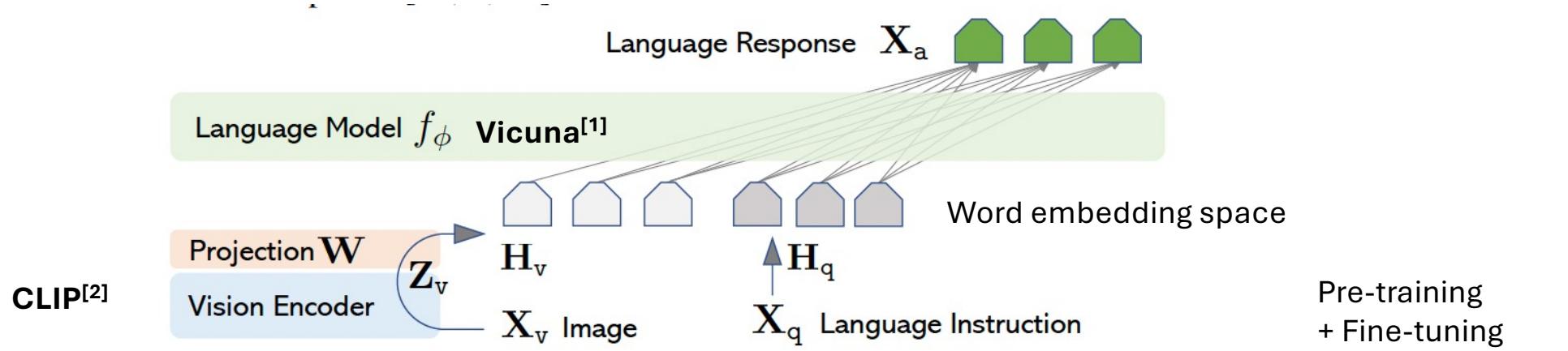
LLaVA

- Solution

- Data engineering: construct the first large image-text conversational dataset. Image (from COCO) is translated into a **symbolic representation** {caption, bounding boxes} for GPT-4 to understand the image and generate three types of questions (QA, description, complex reasoning).



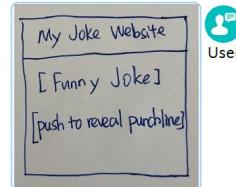
- Compose a larger model using **light-weighted existing** pre-trained visual and text encoders.



LLaVA: going beyond matching

- (Amazing) Results

Coding



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

```
<!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: 20px; margin-bottom: 20px; }
p { font-size: 16px; }
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!

My Joke Website

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

Push me!

Rendered website

Rendered website
(clicked the button)

Perception and planning



User

What are the meals that I can cook with these?



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.

User



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)

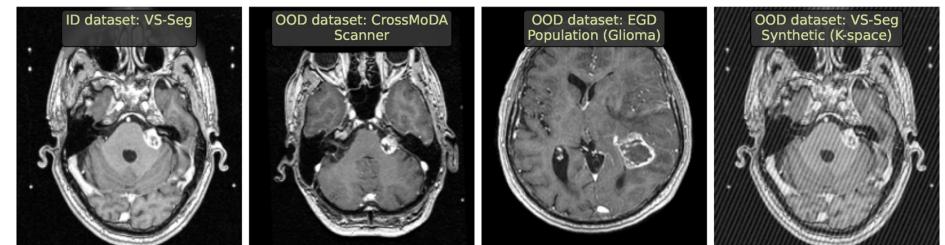
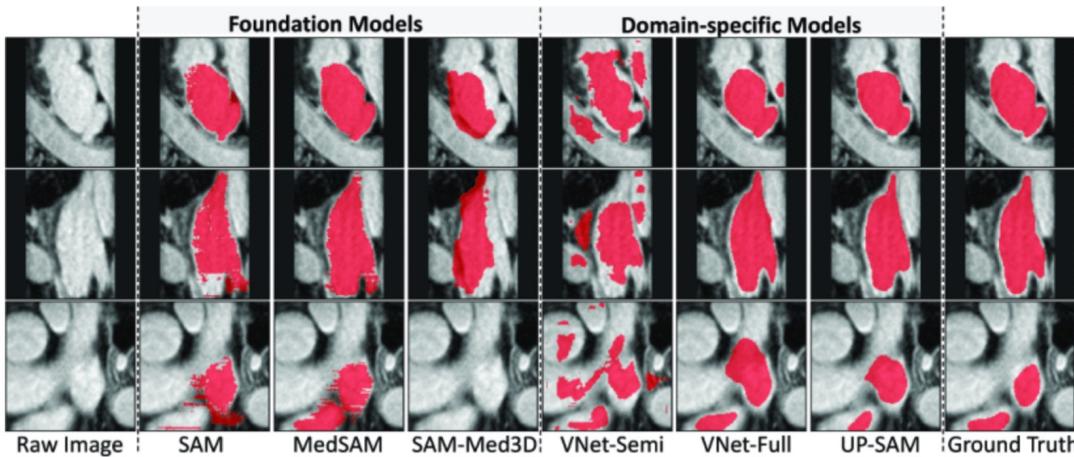
Instructions:

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.

Responsibility issues

- Uncertainty is not estimated well.
 - How can a model know it does not know?
- Out-of-distribution data



UP-SAM: Uncertainty-Informed Adaptation of Segment Anything Model for Semi-Supervised Medical Image Segmentation. 2024 BIBM
Out-of-distribution detection in medical image analysis: A survey. 2024

Responsibility issues

- When querying images of “doctor”(left)
 - White male
 - White female
 - Hispanic male
 - Hispanic female
 - Black male
 - Black female
- When querying images of “ceo” (right)

{'corpus_id': 3, 'score': 0.23189577460289001}



{'corpus_id': 1, 'score': 0.22440794110298157}



{'corpus_id': 2, 'score': 0.21883784234523773}



{'corpus_id': 5, 'score': 0.2124222218990326}



{'corpus_id': 4, 'score': 0.19799582660198212}



{'corpus_id': 0, 'score': 0.19406268000602722}



{'corpus_id': 3, 'score': 0.22499889135360718}



{'corpus_id': 1, 'score': 0.20429760217666626}



{'corpus_id': 4, 'score': 0.2039734125137329}



{'corpus_id': 2, 'score': 0.19803905487060547}



{'corpus_id': 0, 'score': 0.19433677196502686}



{'corpus_id': 5, 'score': 0.15472416579723358}



Responsibility issues



Describe this image in detail.



The image shows a **table** surrounded by two wooden **chairs**. There's a **coffee cup** on the table's surface near the edge.

Source: Retrieve-then-compare mitigates visual hallucination in multi-modal large language models
<https://www.alphxiv.org/overview/2505.19474v1>

LLaVA1.5-7B+CCA:

The image depicts a busy airport scene with a large group of people standing in line, waiting to **check in for their flights**. There are numerous suitcases and handbags scattered throughout the scene, indicating that the passengers are preparing for their journeys.
In total, there are at least 14 people visible in the image, with some standing closer to the foreground and others further back. The suitcases and handbags are spread across the scene, **with at least 11 suitcases and 6 handbags visible**. The suitcases come in various sizes and are placed in different positions, while the handbags are held by some of the passengers or placed on the ground.
The atmosphere suggests that the passengers are eager to board their flights and continue their travels.

LLaVA1.5-7B+MCA:

The image depicts a busy airport scene with a large group of people standing around and **waiting for their luggage**. There are numerous suitcases and backpacks scattered throughout the area, indicating that the passengers are in the process of **collecting their belongings**.
In the crowd, a man wearing a blue shirt stands out, possibly waiting for his luggage as well. The people are standing in various positions, some closer to the foreground and others further back, creating a sense of depth in the scene. The overall atmosphere suggests that the passengers are in the process of disembarking from their flights and preparing to continue their journeys.



User: Please describe this image in detail.