Recent Developments in Vision-Language Transformers

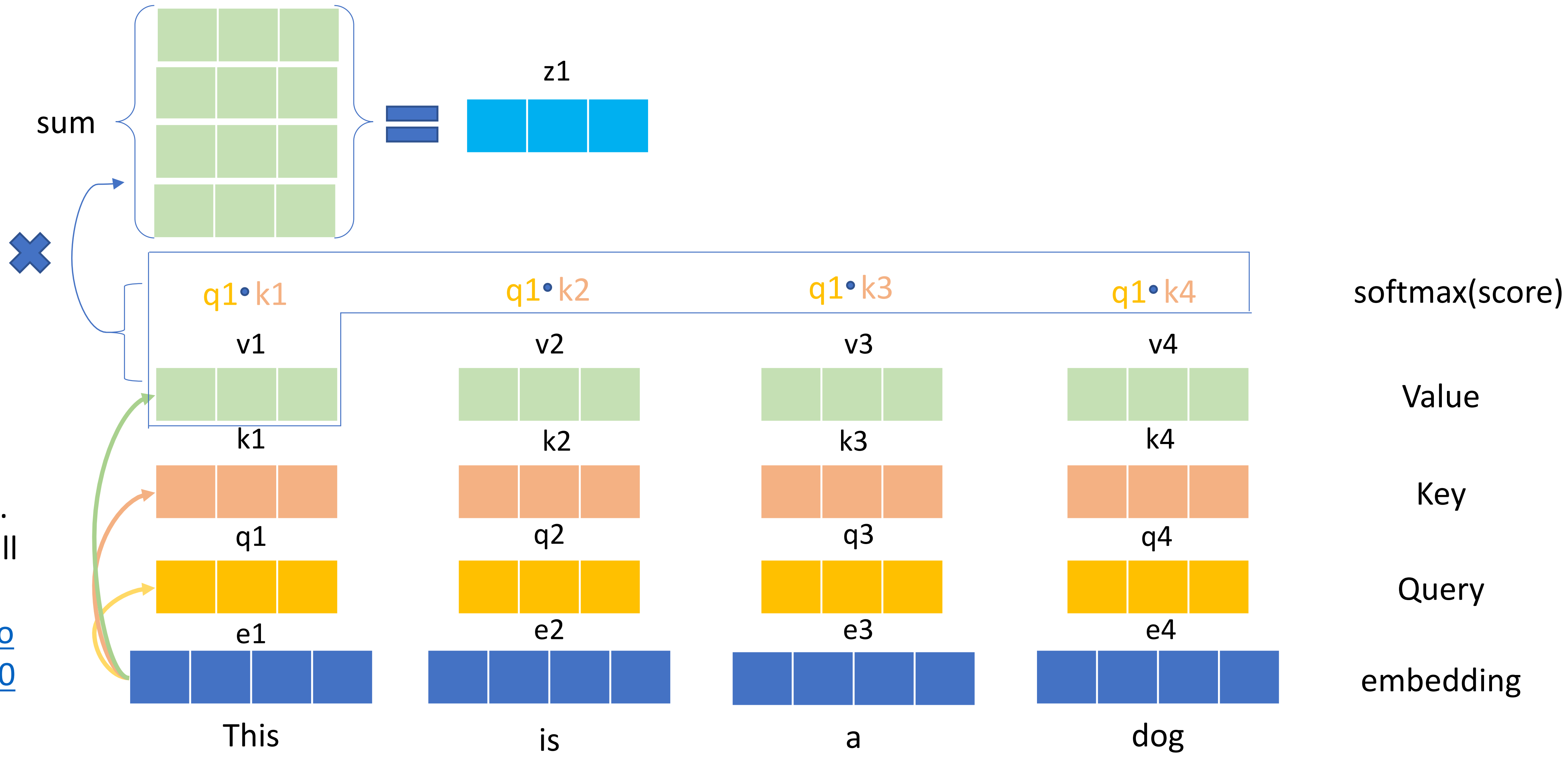
VisualBERT, PixelBERT, VILT and ALBEF

RICE UNIVERSITY

Today

- Review: Attention is All you Need!
- Review: BERT
- VisualBERT
- PixelBERT
- ViLT
- ALBEF

Attention is All you Need



Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

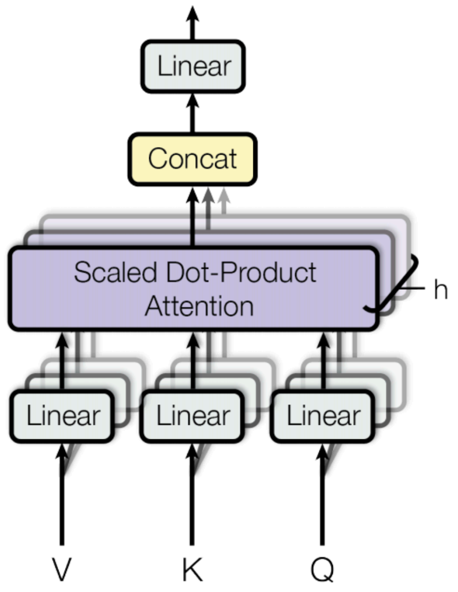
Multi-head Attention: Do not settle for just one set of attention weights.

Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Multi-Head Attention

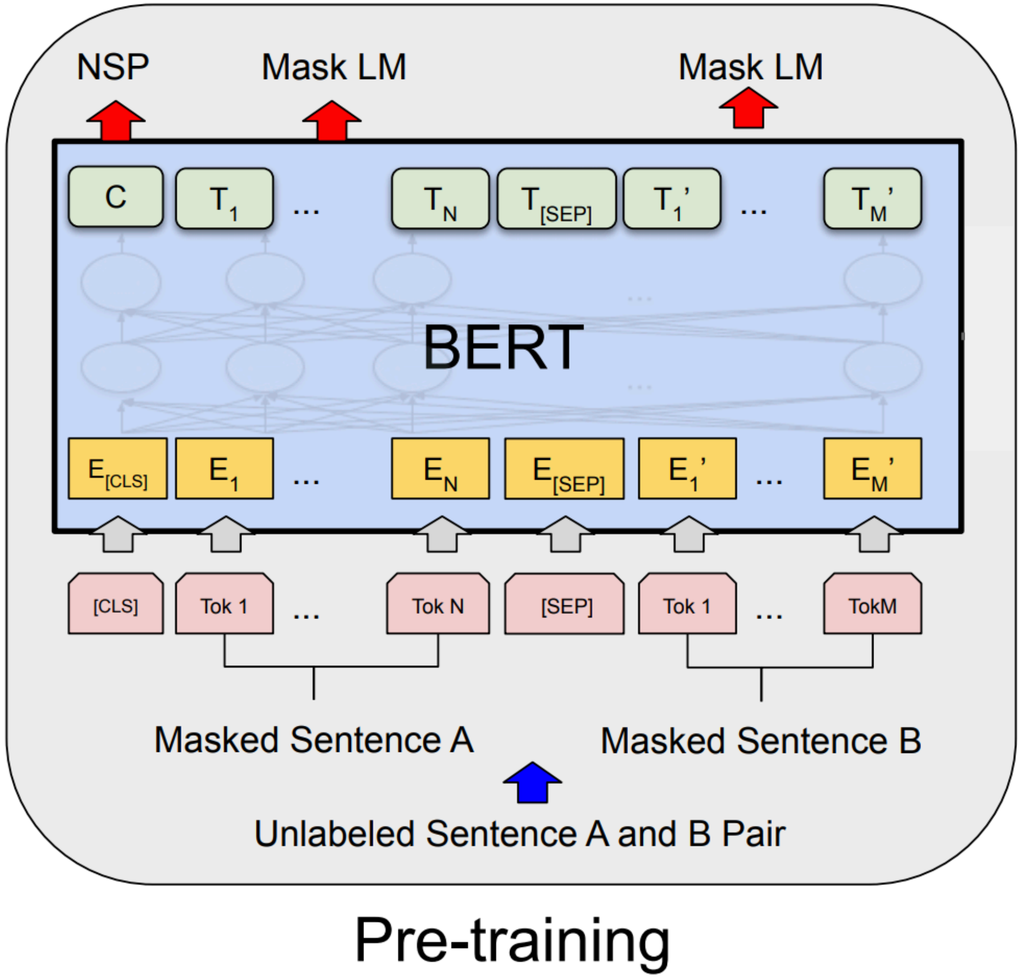


The BERT Encoder Model

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding . https://arxiv.org/abs/1810.04805

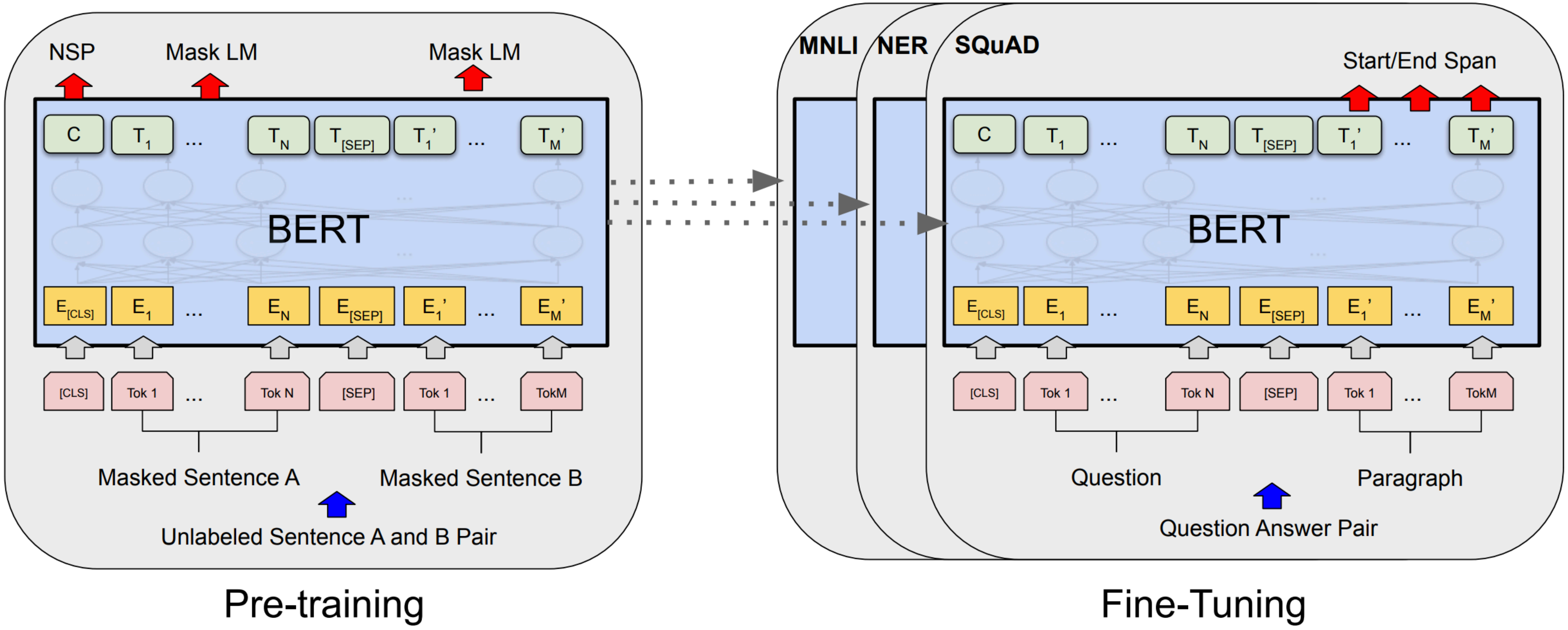
Important things to know

- No decoder
- Masking Language Modeling (**MLM**): Train the model to fill-in-the-blank by masking some of the input tokens and trying to recover the full sentence.
- The input is not one sentence but two sentences separated by a [SEP] token.
- Next Sentence Prediction (**NSP**): Also try to predict whether these two input sentences are consecutive or not.



The BERT Encoder Model

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding . https://arxiv.org/abs/1810.04805

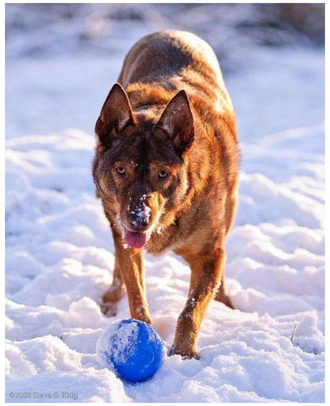


Datasets

MSCOCO:

a umbrella stuck into sand at a beach with boats and hills in the background.
a beach umbrella with a backpack underneath it is on the beach.
an umbrella on a beach with a backpack and bag under it.
a green umbrella sitting on top of a sandy beach.
an umbrella provides shade on a beach in front of the water.

SBU:



My dog playing with her favorite ball in the snow.

Visual Genome:

frisbee flying above tree line
green frisbee with white cloud
sky is gray and cloudy
…

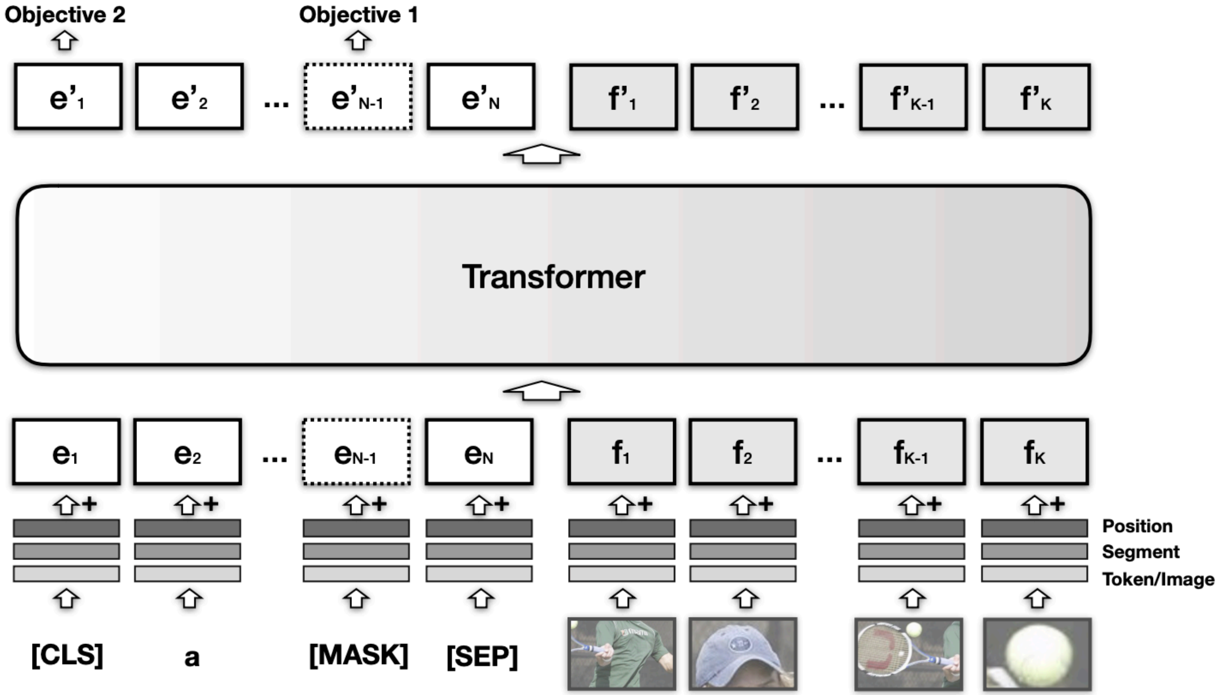
VisualBERT

Li, Liunian Harold, et al. Visualbert: A simple and performant baseline for vision and language. https://arxiv.org/pdf/1908.03557.pdf

Important things to know

- Input: paired image + two sentences
- Faster-RCNN image features
- Objective 1: Masking Language Modeling (**MLM**)
- Objective 2: Sentence Image Prediction: predict whether both sentences are describing the input image.

A person hits a ball with a tennis racket

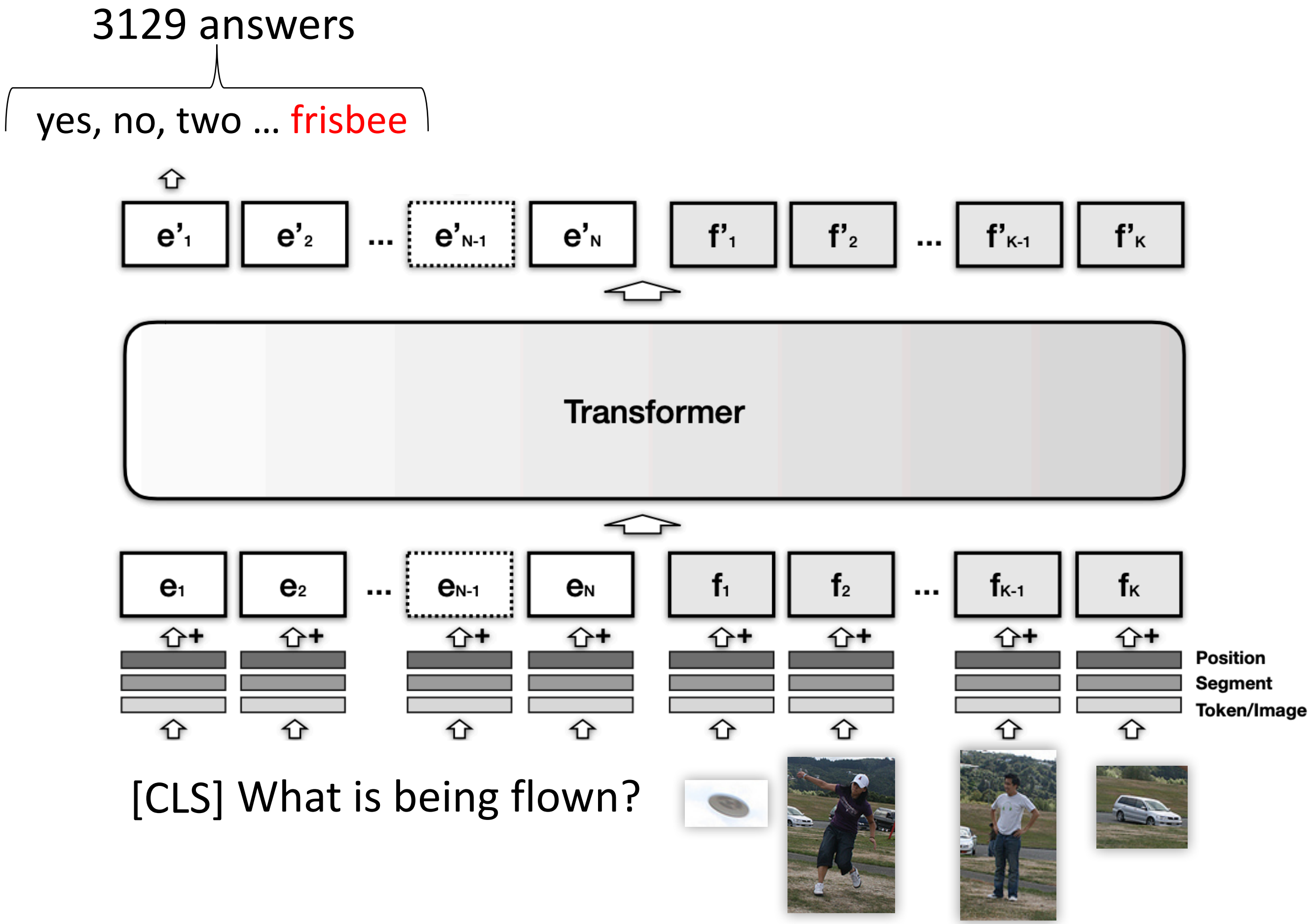


VisualBERT

Li, Liunian Harold, et al. Visualbert: A simple and performant baseline for vision and language. https://arxiv.org/pdf/1908.03557.pdf

Downstream tasks

- Visual Question Answering (VQA)
- Visual Commonsense Reasoning (VCR)
- Natural Language for Visual Reasoning (NLVR)
- Visual Grounding (VG)

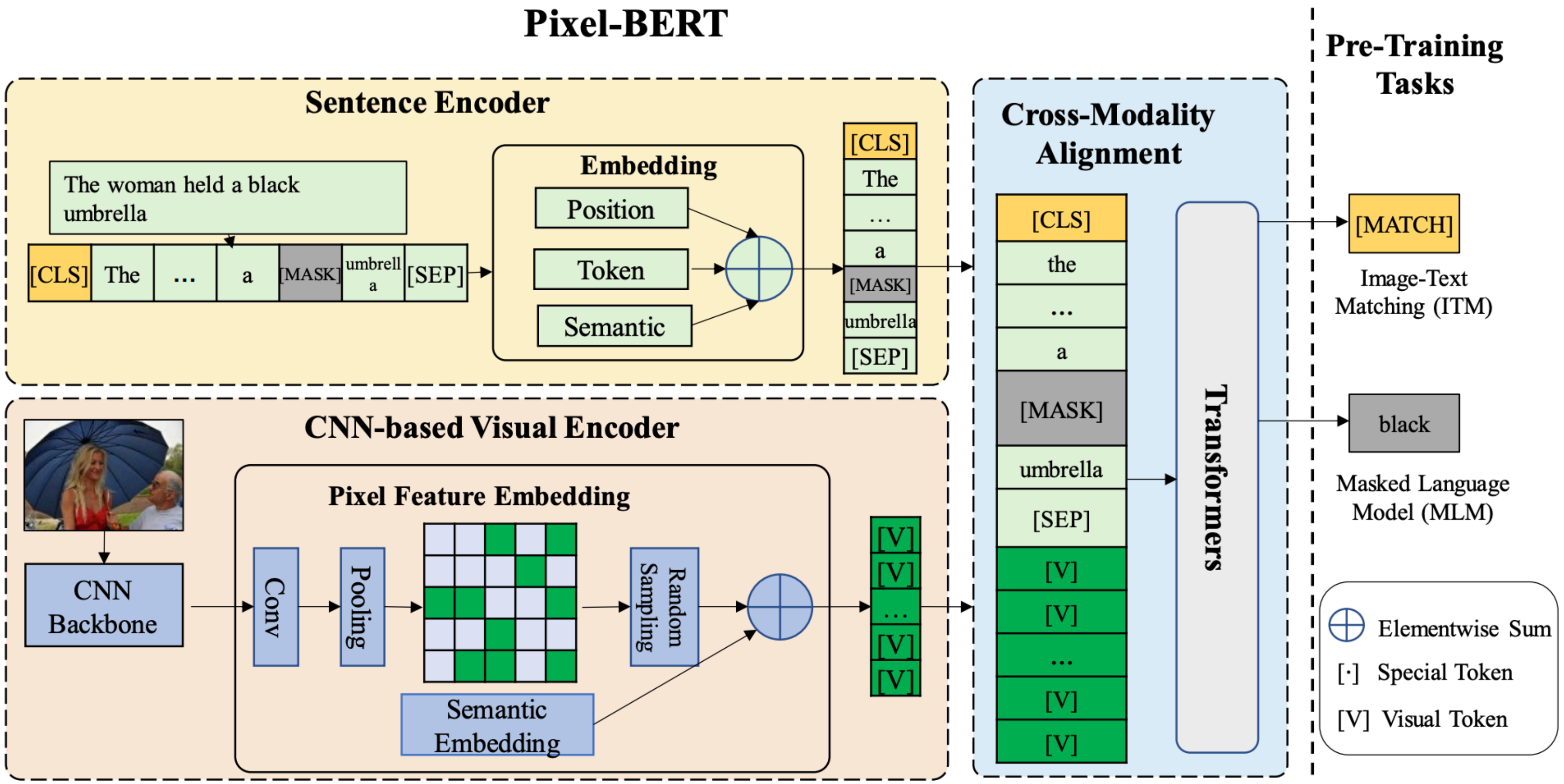


PixelBERT

Huang, Zhicheng, et al. Pixel-bert: Aligning image pixels with text by deep multi-modal transformers. https://arxiv.org/pdf/2004.00849.pdf

Important things to know

- Input: paired image + sentence
- CNN image features
- Masking Language Modeling (**MLM**)
- Image Text Matching (**ITM**): predict if the input image and text are matched

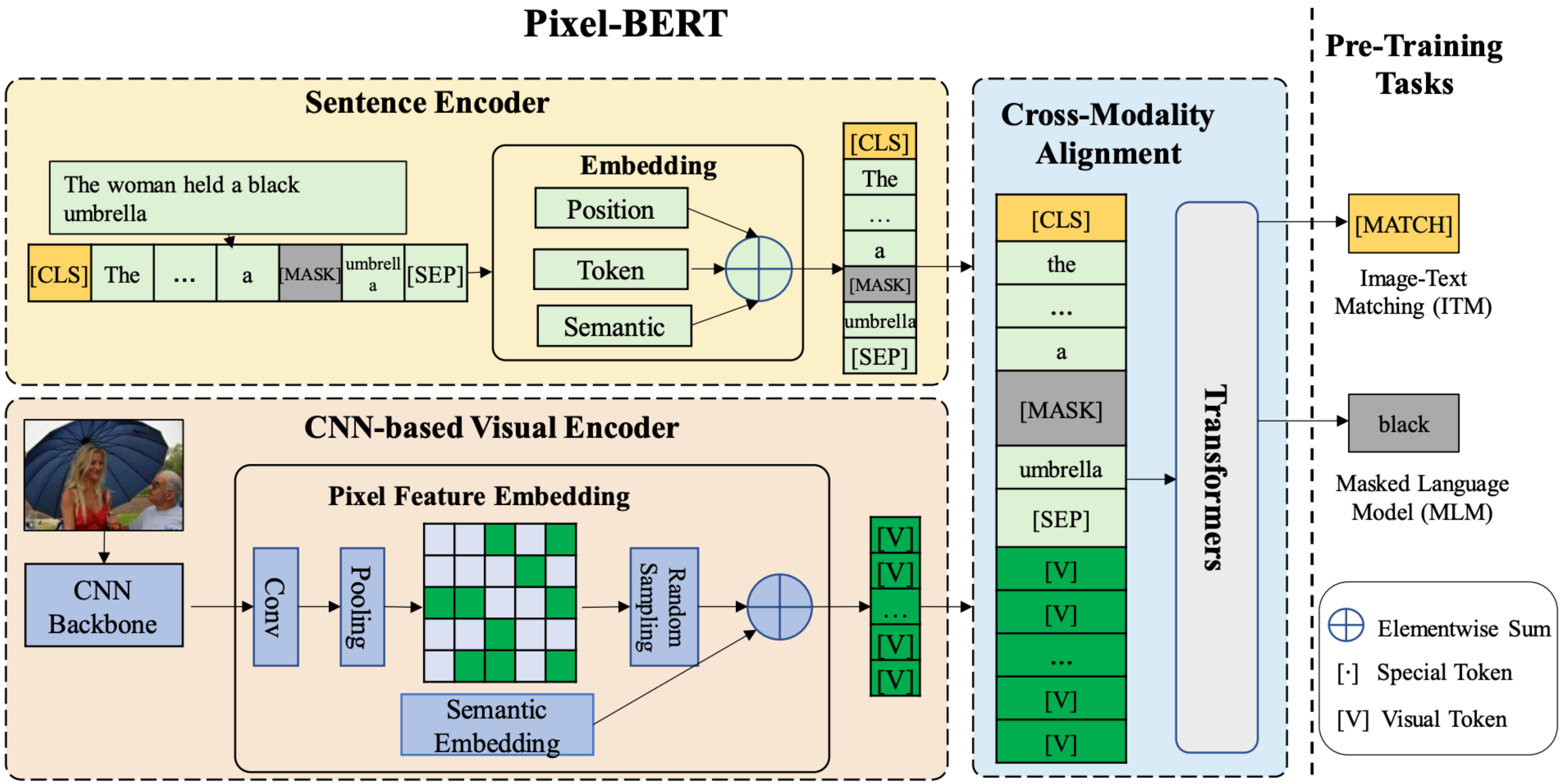


PixelBERT

Huang, Zhicheng, et al. Pixel-bert: Aligning image pixels with text by deep multi-modal transformers. https://arxiv.org/pdf/2004.00849.pdf

Downstream tasks

- Visual Question Answering (VQA)
- Natural Language for Visual Reasoning (NLVR)
- Image Retrieval & Text Retrieval (IR & TR)



PixelBERT

Huang, Zhicheng, et al. Pixel-bert: Aligning image pixels with text by deep multi-modal transformers. https://arxiv.org/pdf/2004.00849.pdf

Downstream tasks

- Visual Question Answering (VQA)
- Natural Language for Visual Reasoning (NLVR)
- Image Retrieval & Text Retrieval (IR & TR)

0.85, 0.12, 0.02, …

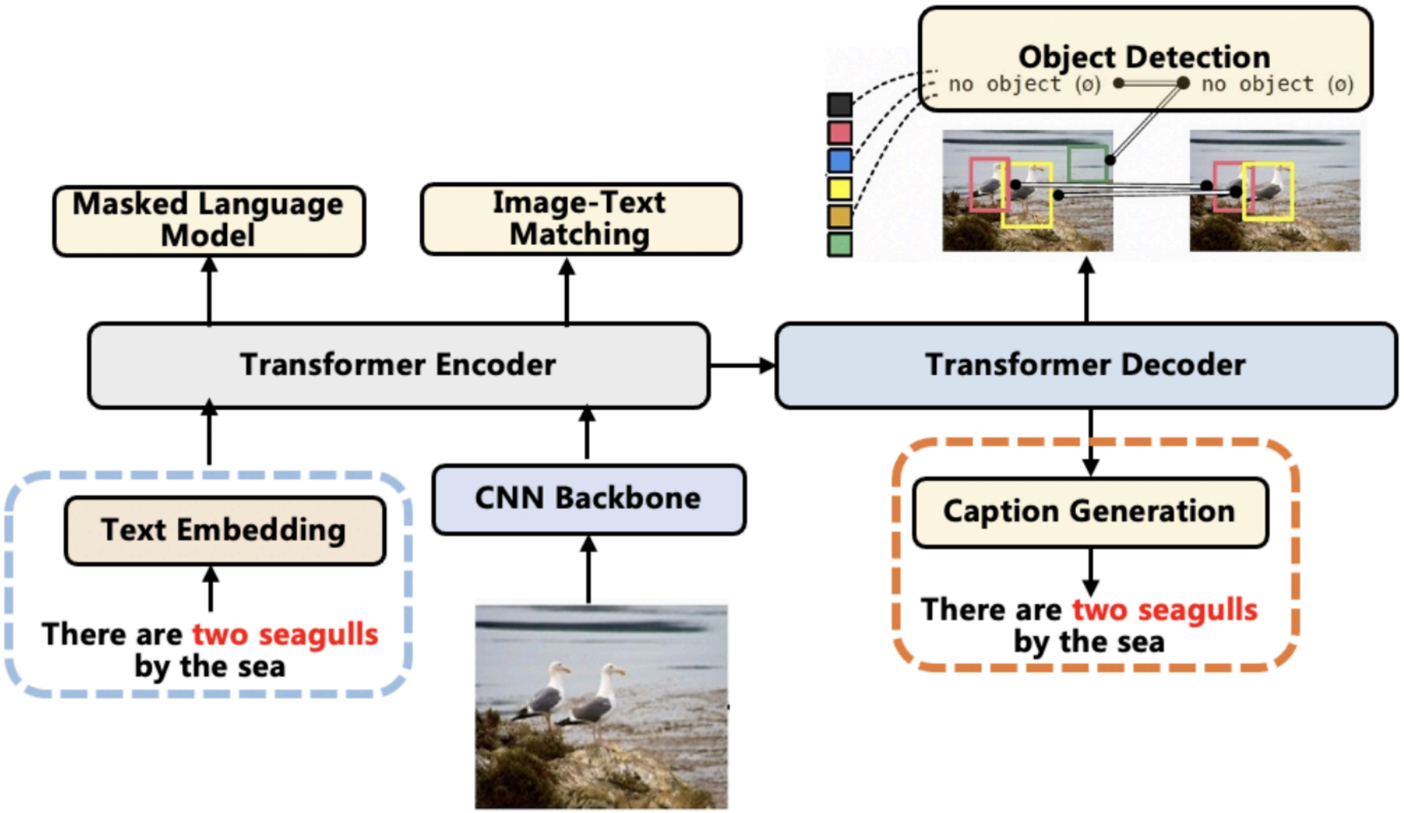
PixelBERT

Positive:
a fluffy brown cat is laying in a bathroom sink.

Negative:
an image of a cat looking out the window
a man stands in front of an elephant
a white building that is a medical clinic.
…

Reading for you: E2E-VLP

Xu, Haiyang, et al. E2E-VLP: End-to-End Vision-Language Pre-training Enhanced by Visual Learning. https://arxiv.org/abs/2106.01804

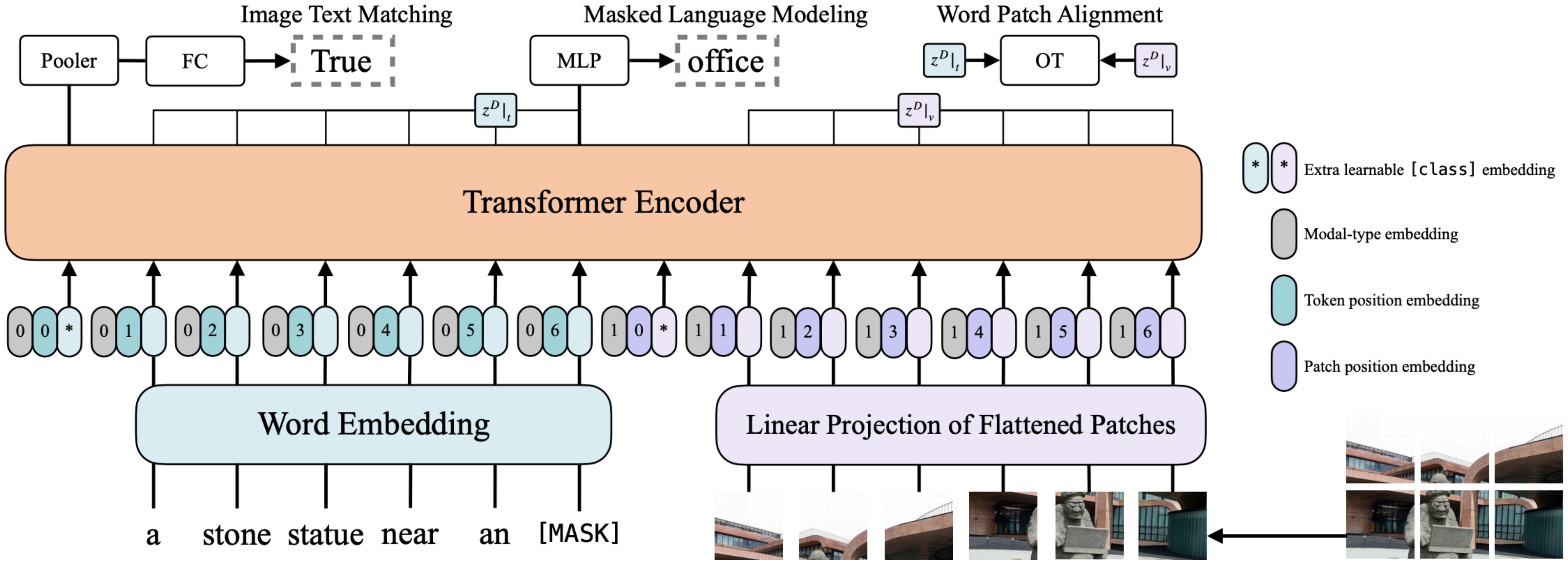


ViLT

Kim, Wonjae, et al. ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision
https://arxiv.org/pdf/2102.03334.pdf

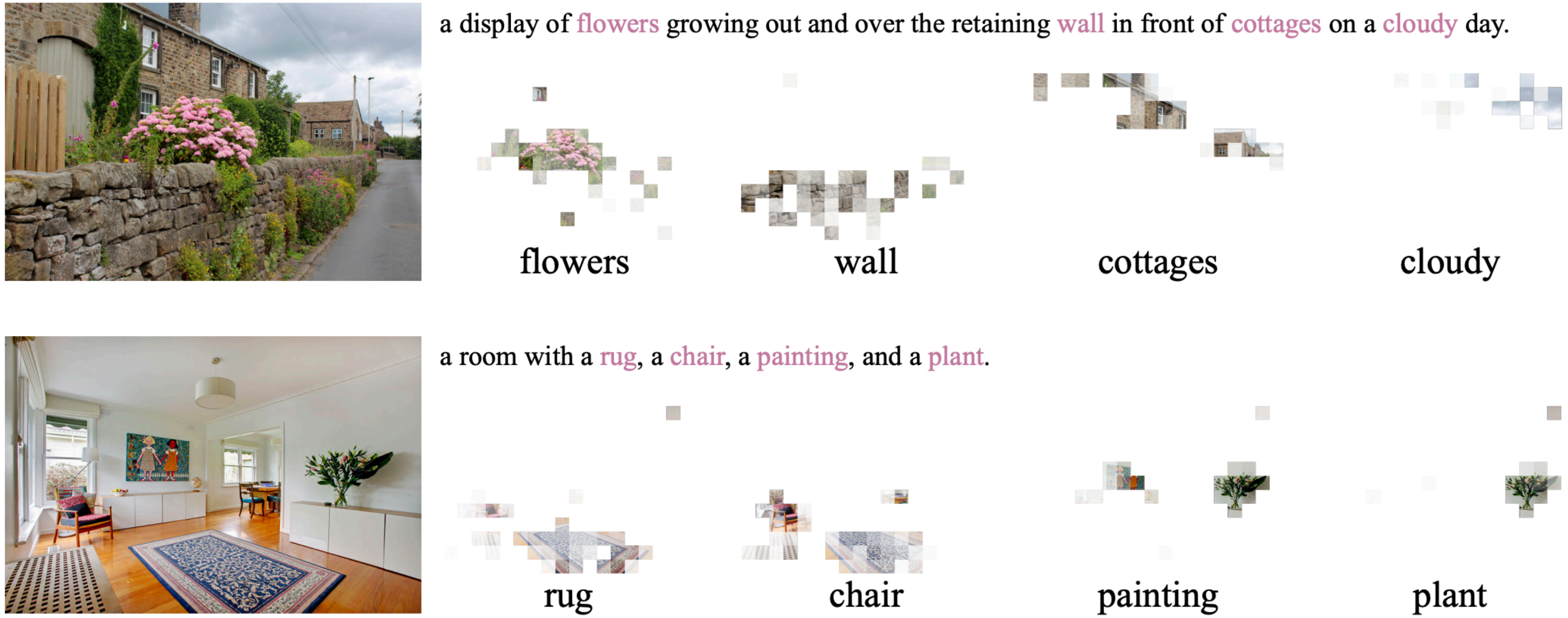
Important things to know

- Input: paired image + sentence
- Image inputs are flattened patches
- Masking Language Modeling (**MLM**)
- Image Text Matching (**ITM**)
- Word Patch Alignment (**WPA**): predict if the patch and the token are matched



ViLT

- inexact proximal point method for optimal transports (IPOT)

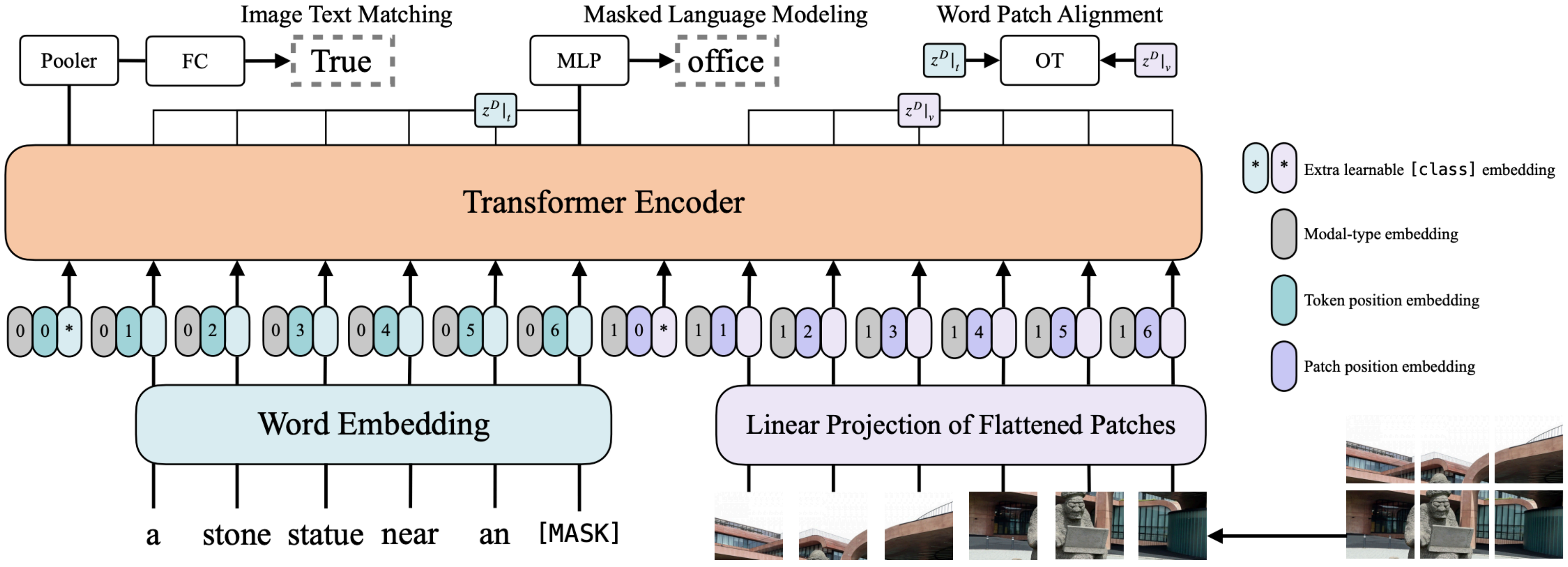


Xie, Yujia, et al. "A fast proximal point method for computing exact wasserstein distance." https://arxiv.org/abs/1802.04307

ViLT

Kim, Wonjae, et al. ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision
https://arxiv.org/pdf/2102.03334.pdf

Downstream tasks



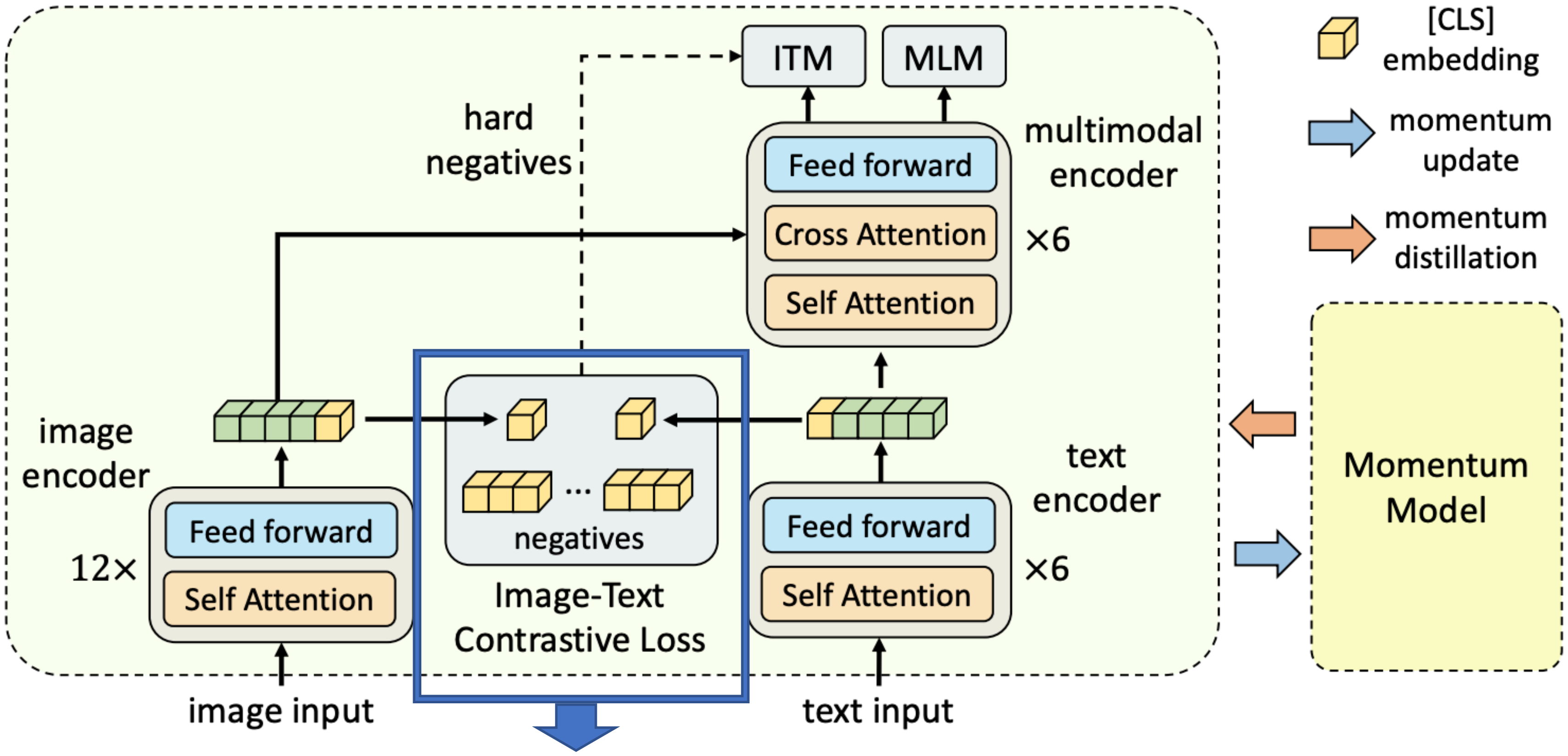
- Visual Question Answering (VQA)
- Natural Language for Visual Reasoning (NLVR)
- Image Retrieval & Text Retrieval (IR & TR)

ALBEF

Li, Junnan, et al. Align before Fuse: Vision and Language Representation Learning with Momentum Distillation https://arxiv.org/pdf/2107.07651.pdf

Important things to know

- Input: paired image + sentence
- Use ViT to encode images
- Masking Language Modeling (**MLM**)
- Image Text Matching (**ITM**)
- Image-Text Contrastive loss: make positive image-text pairs similar to each other

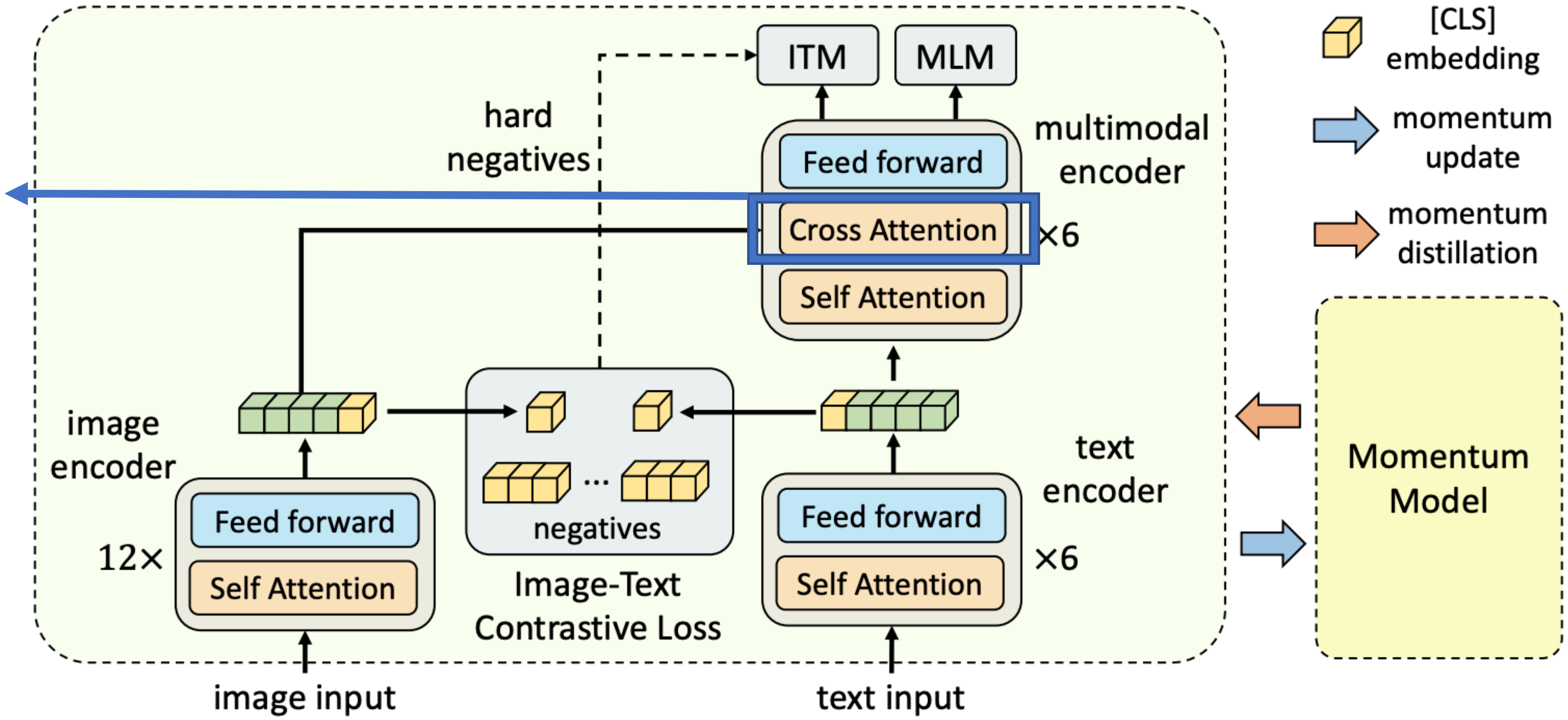


CLIP

ALBEF

Li, Junnan, et al. Align before Fuse: Vision and Language Representation Learning with Momentum Distillation https://arxiv.org/pdf/2107.07651.pdf

Query: text vector
Key, Value: image vectors

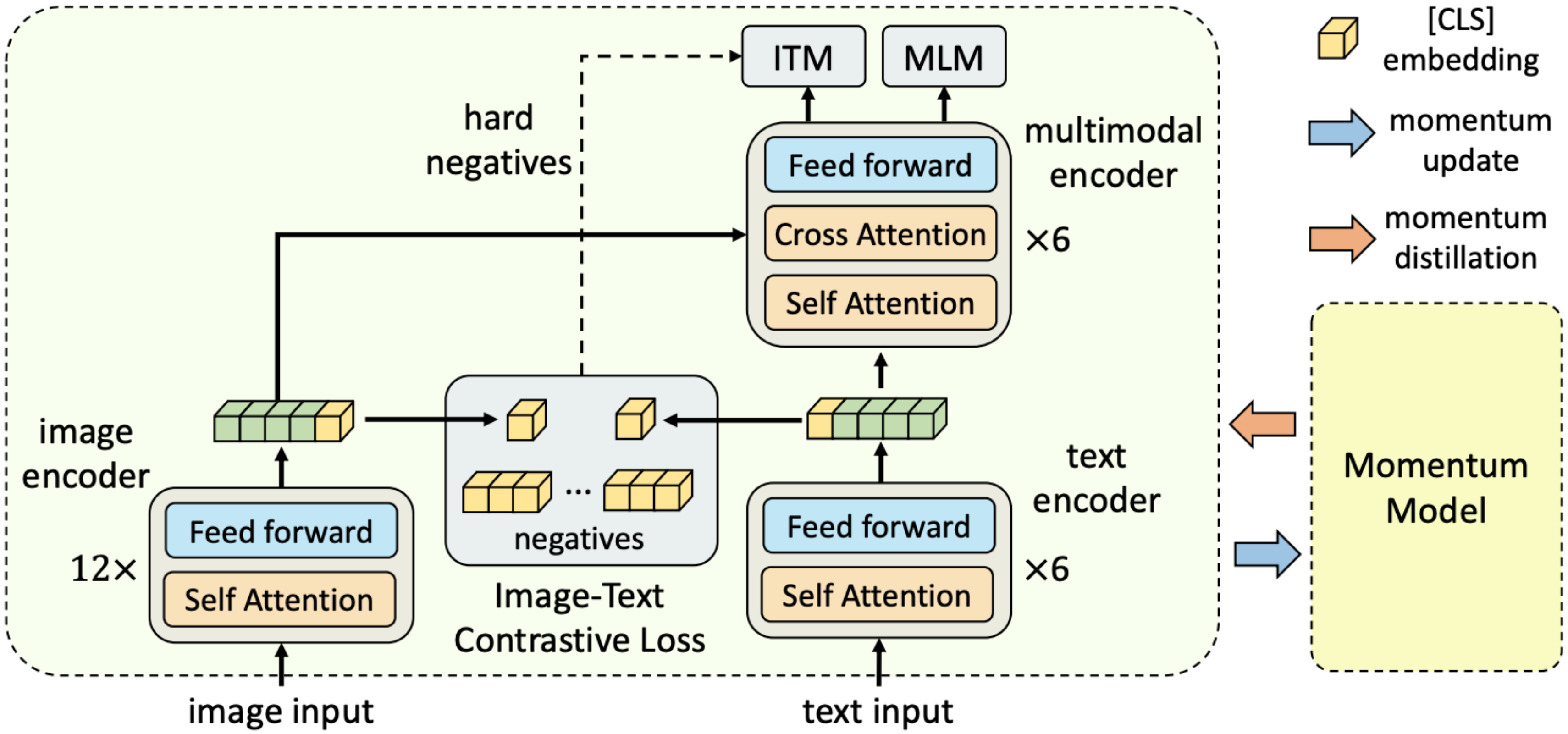


ALBEF

Li, Junnan, et al. Align before Fuse: Vision and Language Representation Learning with Momentum Distillation https://arxiv.org/pdf/2107.07651.pdf

Downstream tasks

- Visual Question Answering (VQA)
- Natural Language for Visual Reasoning (NLVR)
- Image Retrieval & Text Retrieval (IR & TR)
- Visual Entailment (VE)
- Visual Grounding (VG)

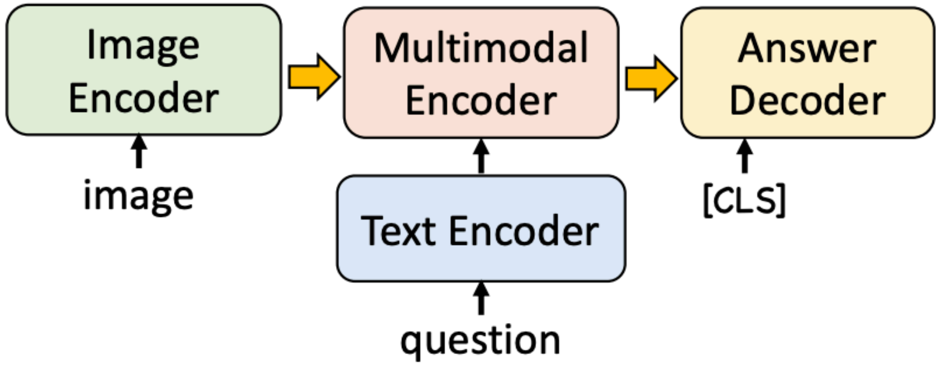


ALBEF

Li, Junnan, et al. Align before Fuse: Vision and Language Representation Learning with Momentum Distillation https://arxiv.org/pdf/2107.07651.pdf

Downstream tasks

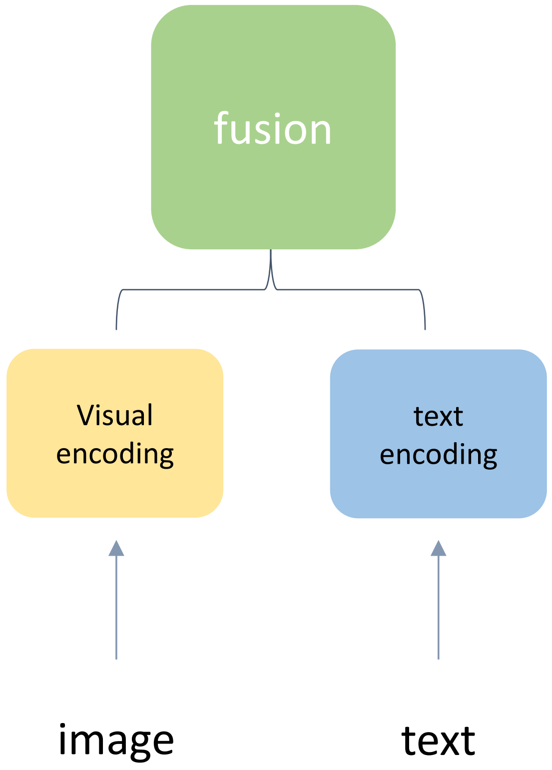
- Visual Question Answering (VQA)
- Natural Language for Visual Reasoning (NLVR)
- Image Retrieval & Text Retrieval (IR & TR)
- Visual Entailment (VE)
- Visual Grounding (VG)



(a) VQA

Summary

- Image encoding:
  - CNN
  - Object detectors
  - Patches
  - Visual Transformer
  - etc.
- Text encoding:
  - BERT
  - Word embedding
  - etc.



Download

- VisualBERT: https://github.com/uclanlp/visualbert
- VILT: https://github.com/dandelin/vilt
- ALBEF: https://github.com/salesforce/ALBEF

Code - ALBEF

```python
from functools import partial
from models.vit import VisionTransformer
from models.xbert import BertConfig, BertModel
from models.tokenization_bert import BertTokenizer

import torch
from torch import nn
from torchvision import transforms

import json

class VL_Transformer_ITM(nn.Module):
    def __init__(self,
                 text_encoder = None,
                 config_bert = ''
                 ):
        super().__init__()

        bert_config = BertConfig.from_json_file(config_bert)

        self.visual_encoder = VisionTransformer(
            img_size=384, patch_size=16, embed_dim=768, depth=12, num_heads=12,
            mlp_ratio=4, qkv_bias=True, norm_layer=partial(nn.LayerNorm, eps=1e-6))

        self.text_encoder = BertModel.from_pretrained(text_encoder, config=bert_config, add_pooling_layer=False)

        self.itm_head = nn.Linear(768, 2)


    def forward(self, image, text):
        image_embeds = self.visual_encoder(image)

        image_atts = torch.ones(image_embeds.size()[:-1],dtype=torch.long).to(image.device)

        output = self.text_encoder(text.input_ids,
                                   attention_mask = text.attention_mask,
                                   encoder_hidden_states = image_embeds,
                                   encoder_attention_mask = image_atts,
                                   return_dict = True,
                                  )

        vl_embeddings = output.last_hidden_state[:,0,:]
        vl_output = self.itm_head(vl_embeddings)
        return vl_output
```

Code - ALBEF

```
import torch
import yaml
from models.vit import interpolate_pos_embed
from models.tokenization_bert import BertTokenizer

model_path = 'ALBEF.pth'
bert_config_path = 'configs/config_bert.json'

checkpoint = torch.load(model_path, map_location='cpu')
state_dict = checkpoint['model']

#### Model ####
print("Creating model")
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = VL_Transformer_ITM(text_encoder='bert-base-uncased', config_bert=bert_config_path)

# reshape positional embedding to accomodate for image resolution change
pos_embed_reshaped = interpolate_pos_embed(state_dict['visual_encoder.pos_embed'],model.visual_encoder)
state_dict['visual_encoder.pos_embed'] = pos_embed_reshaped

for key in list(state_dict.keys()):
    if 'bert' in key:
        encoder_key = key.replace('bert.','')
        state_dict[encoder_key] = state_dict[key]
        del state_dict[key]
msg = model.load_state_dict(state_dict,strict=False)

print('load checkpoint from %s'%model_path)

print(msg)
```

Code - ALBEF

```
from PIL import Image

import cv2
import numpy as np
from torchvision import transforms
from skimage import transform as skimage_transform
from scipy.ndimage import filters
from matplotlib import pyplot as plt

normalize = transforms.Normalize((0.48145466, 0.4578275, 0.40821073), (0.26862954, 0.26130258, 0.27577711))

transform = transforms.Compose([
    transforms.Resize((384,384),interpolation=Image.BICUBIC),
    transforms.ToTensor(),
    normalize,
])

image_path = 'examples/image0.jpg'
image_pil = Image.open(image_path).convert('RGB')
image = transform(image_pil).unsqueeze(0)

caption = 'the woman is working on her computer at the desk'
text = pre_caption(caption)
text_input = tokenizer(text, return_tensors="pt")
```

```
import re

def pre_caption(caption,max_words=50):
    caption = re.sub(
        r"([,.'!?\"()*#:;~])",
        '',
        caption.lower(),
    ).replace('-', ' ').replace('/', ' ')

    caption = re.sub(
        r"\s{2,}",
        ' ',
        caption,
    )
    caption = caption.rstrip('\n')
    caption = caption.strip(' ')

    #truncate caption
    caption_words = caption.split(' ')
    if len(caption_words)>max_words:
        caption = ' '.join(caption_words[:max_words])
    return caption
```

Code - ALBEF

```
image_embeds = model.visual_encoder(image)
print('Visual Transformer output shape is: ', image_embeds.shape)

image_atts = torch.ones(image_embeds.size()[:-1],dtype=torch.long).to(image.device)
output = model.text_encoder(text_input.input_ids,
                            attention_mask = text_input.attention_mask,
                            encoder_hidden_states = image_embeds,
                            encoder_attention_mask = image_atts,
                            return_dict = True,
                            )
vl_embeddings = output.last_hidden_state[:,0,:]
print('Visual Language Transformer output shape is: ', vl_embeddings.shape)
vl_output = model.itm_head(vl_embeddings)
print(vl_output)
```

```
Visual Transformer output shape is:  torch.Size([1, 577, 768])
Visual Language Transformer output shape is:  torch.Size([1, 768])
tensor([[-3.1364,  3.2983]], grad_fn=<AddmmBackward>)
```

Questions?