ViLT Survey

ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision

Written by Kim EunHo

ABSTRACT

From now

VLP(Vision-and-Language Pre-trained) has improved performance on various joint vision-and-language downstream tasks.

Current VLP models rely heavily on the image feature extraction process, and most include Region Supervision (ex. object detection) and Convolutional Architecture.

ViLT replace this process to Patch Projection.

ViLT is up to tens of times faster than previous VLP Models, and show similar or better downstream task performance.

I N T R O D U C T I O N

What has changed?

Convolution networks have become a essential for visual embedding step by AlexNet.

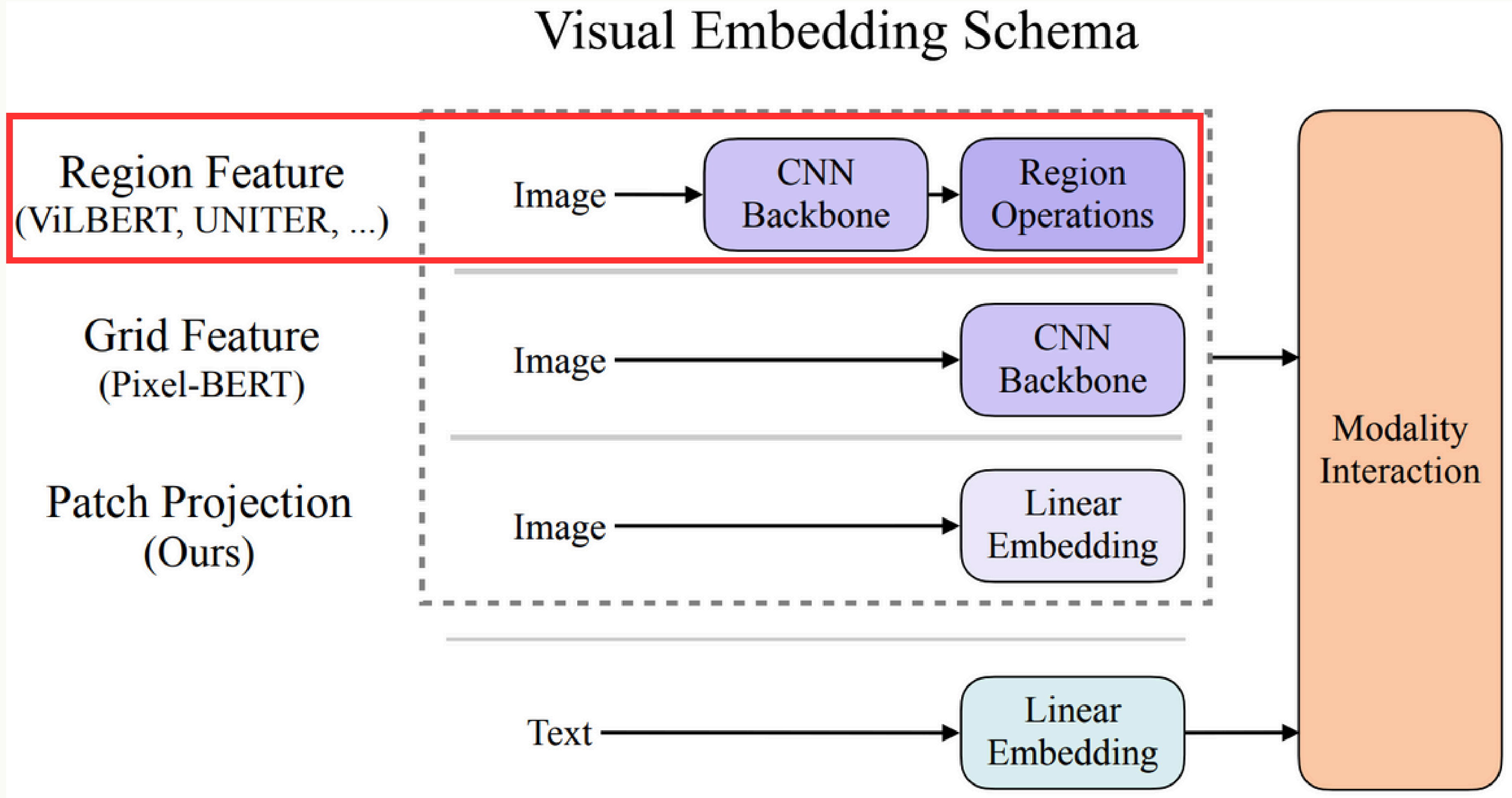
Lot of research to efficiently use visual embedder, but it still have structurally slow extraction process.

ViLT replace this process to linear embedding in a single unifed manner.
(Region Feature, Grid Feature [X] → Patch Projection [O])

For the first time in VLP Models, apply whole-word masking and image augmentation to improve downstream performance.

VISUAL EMBEDDING SCHEMA

Region Feature



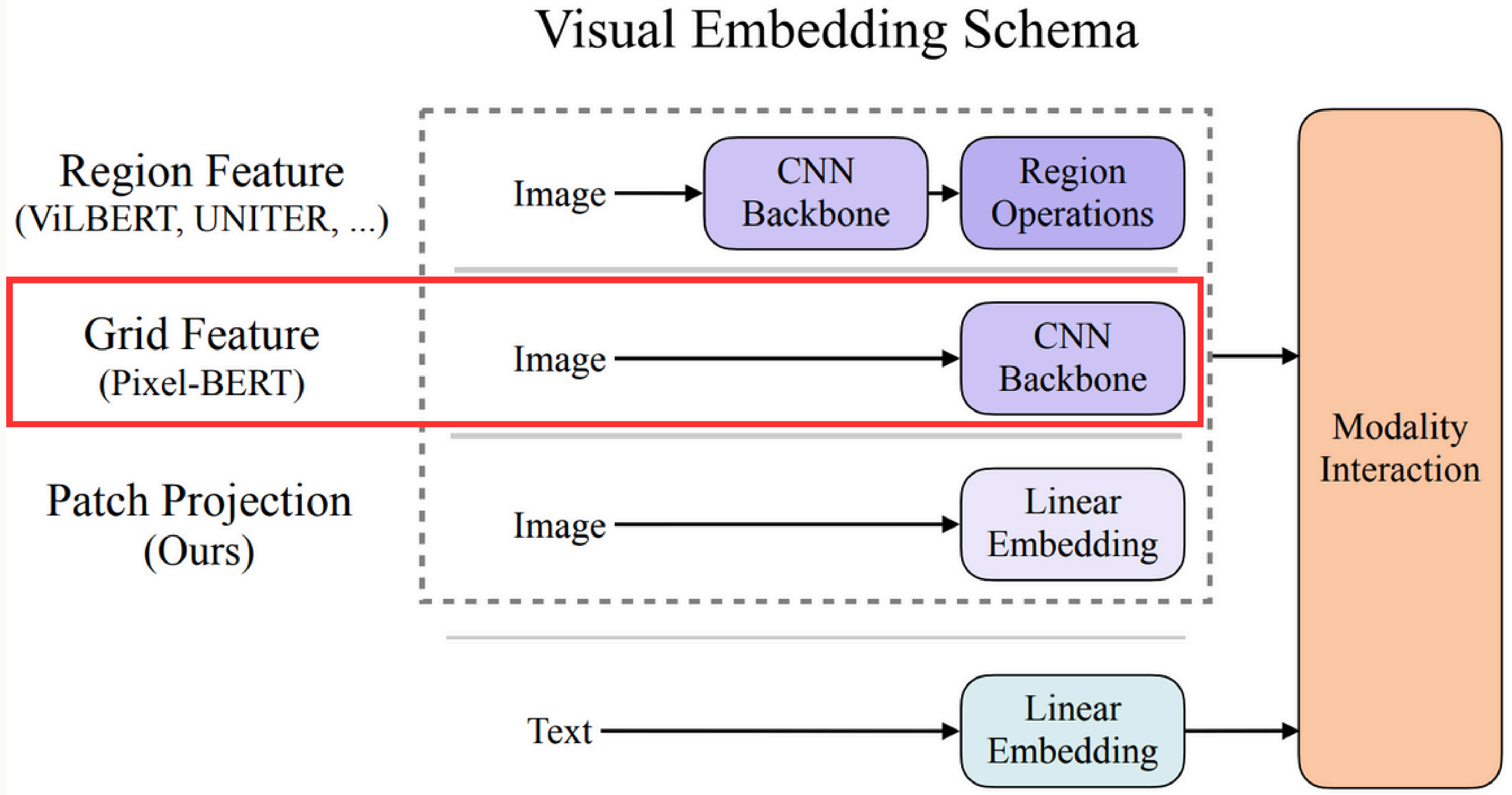
Visualization each Model

CNN Backbone, Region Operations(object detection required)

Extract features from specific regions of an image.

VISUAL EMBEDDING SCHEMA

Grid Feature



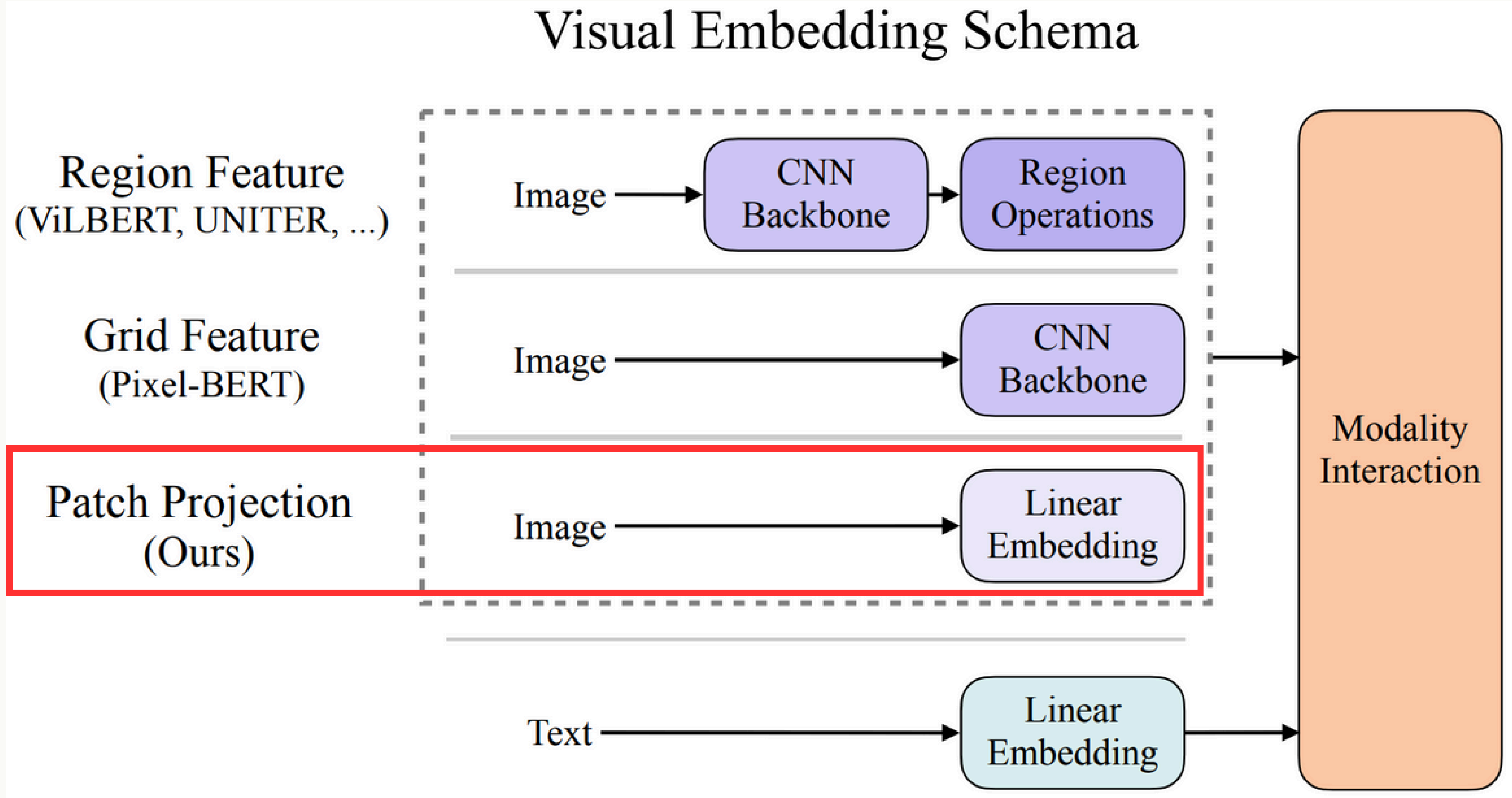
Visualization each Model

CNN Backbone

Divide the image into grids and extract features from each grid cell.

VISUAL EMBEDDING SCHEMA

Patch Projection



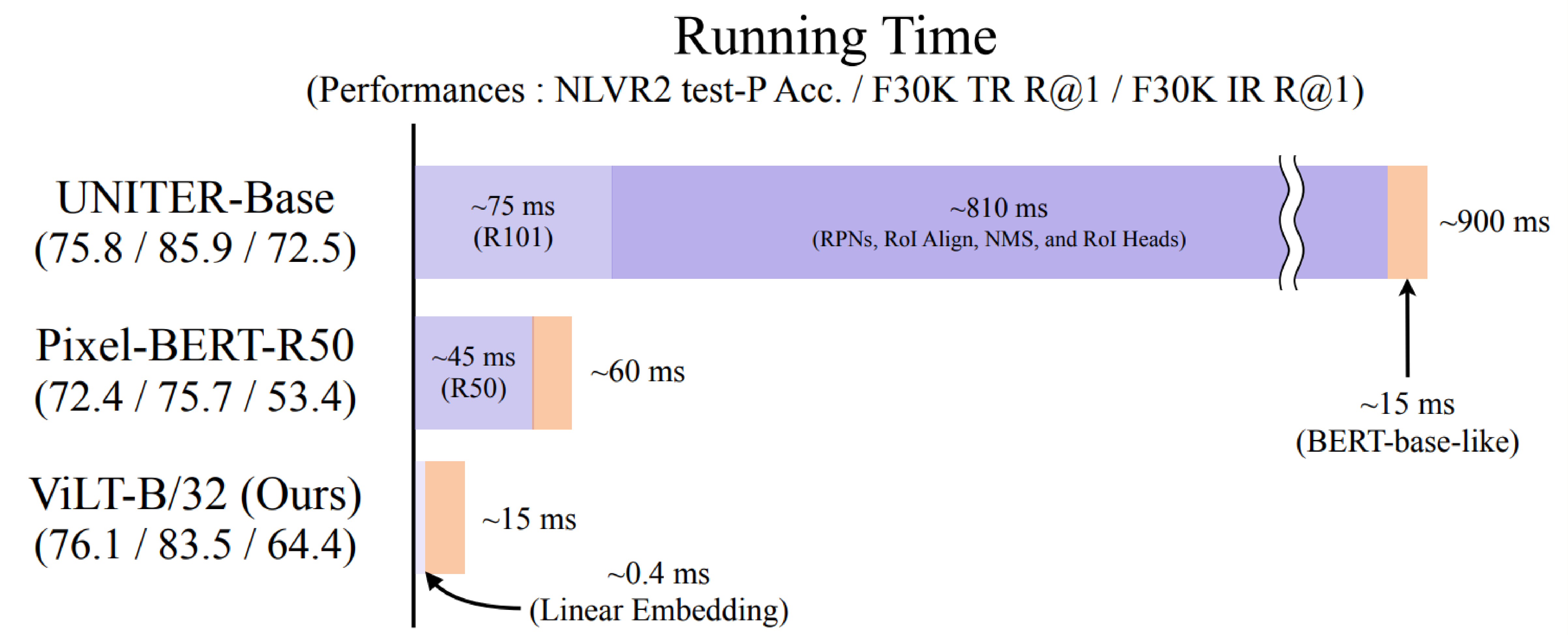
Visualization each Model

Linear Embedding

Divide the image into patches → Flatten → generate low-dimensional linear embeddings.

VISUAL EMBEDDING SCHEMA

Running Time



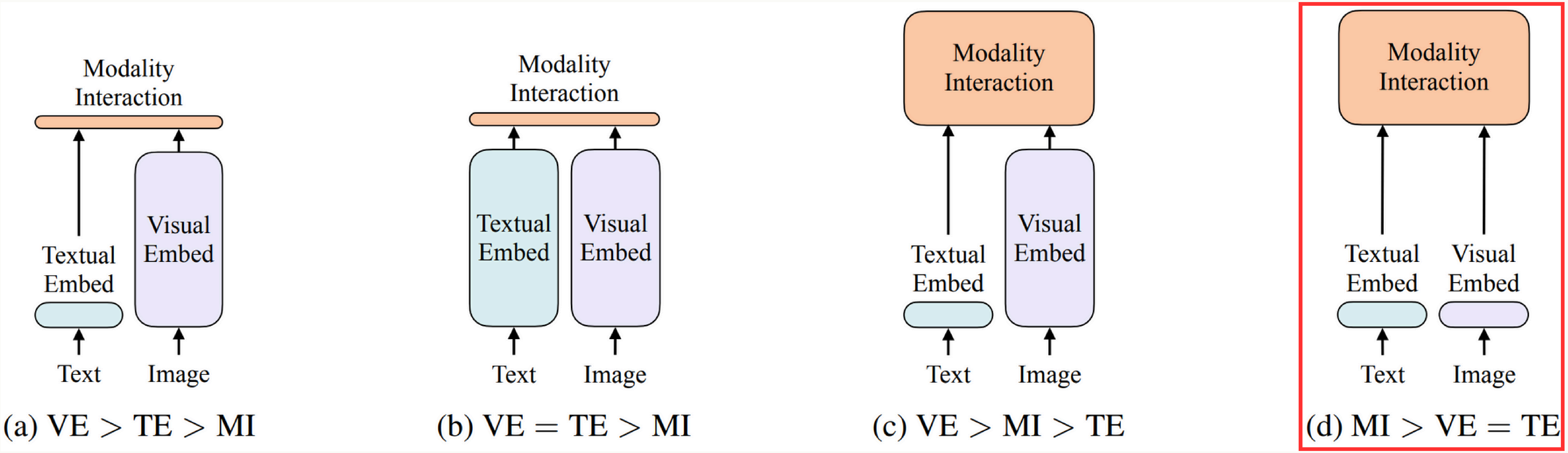
Visualization each Model

Region Operation process occupies almost Running Time.

ViLT-B/32 (Linear Embedding) is fastest.

VLPs ARCHITECTURE

4 Architectures



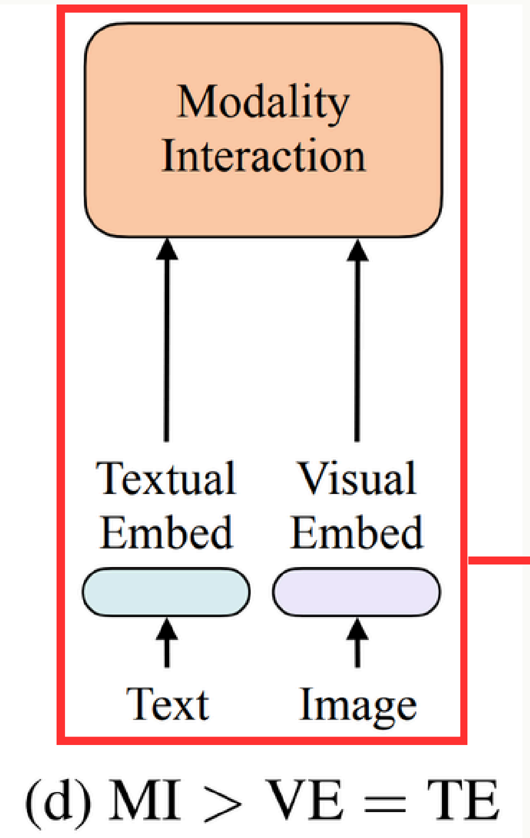
4 VLP Architectures

Equivalent expressiveness level of parameters and/or calculations
Interact in the deep network

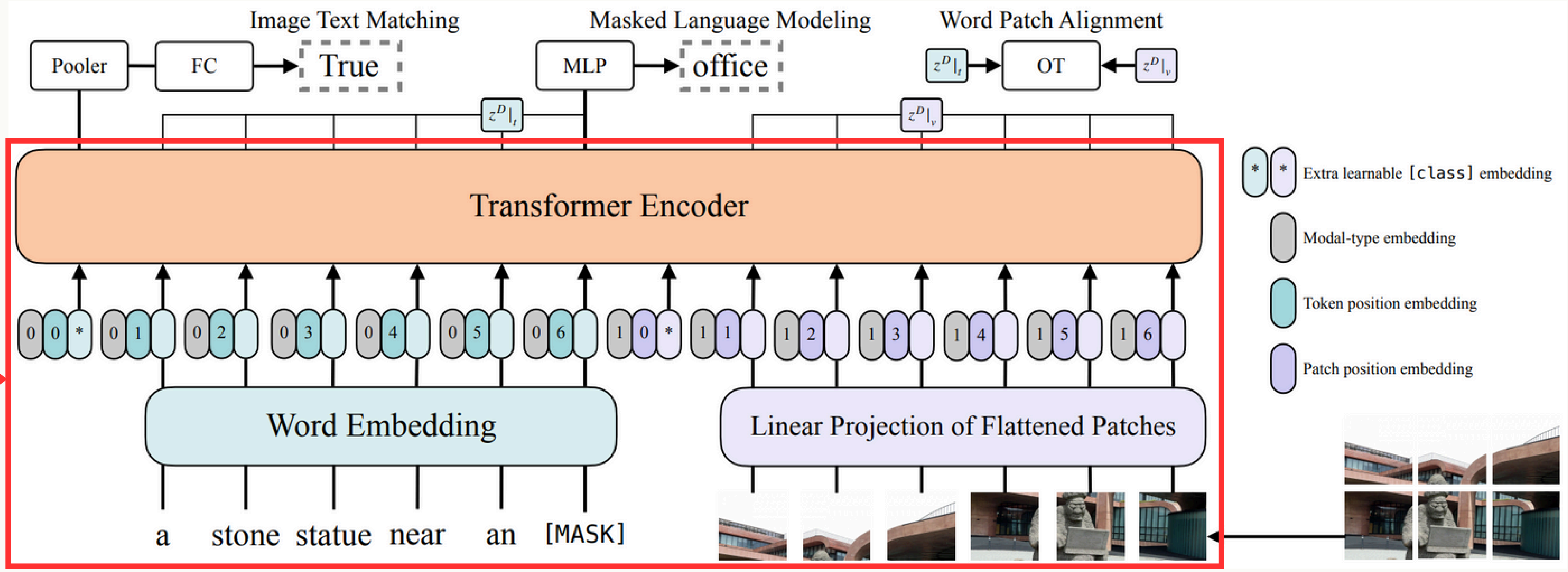
VE : visual embedder, TE : textual embedder, MI : modality interaction

ViLT ARCHITECTURE

In detail



VLP Architecture (d)



ViLT Architecture

Image and text with modal type embedding and position embedding.

Multi Head Self Attention, Feed Forward Network and Residual calculation is proceed.

VISUAL AND LANGUAGE TRANSFORMER

ViLT is similar with ViT-B/32

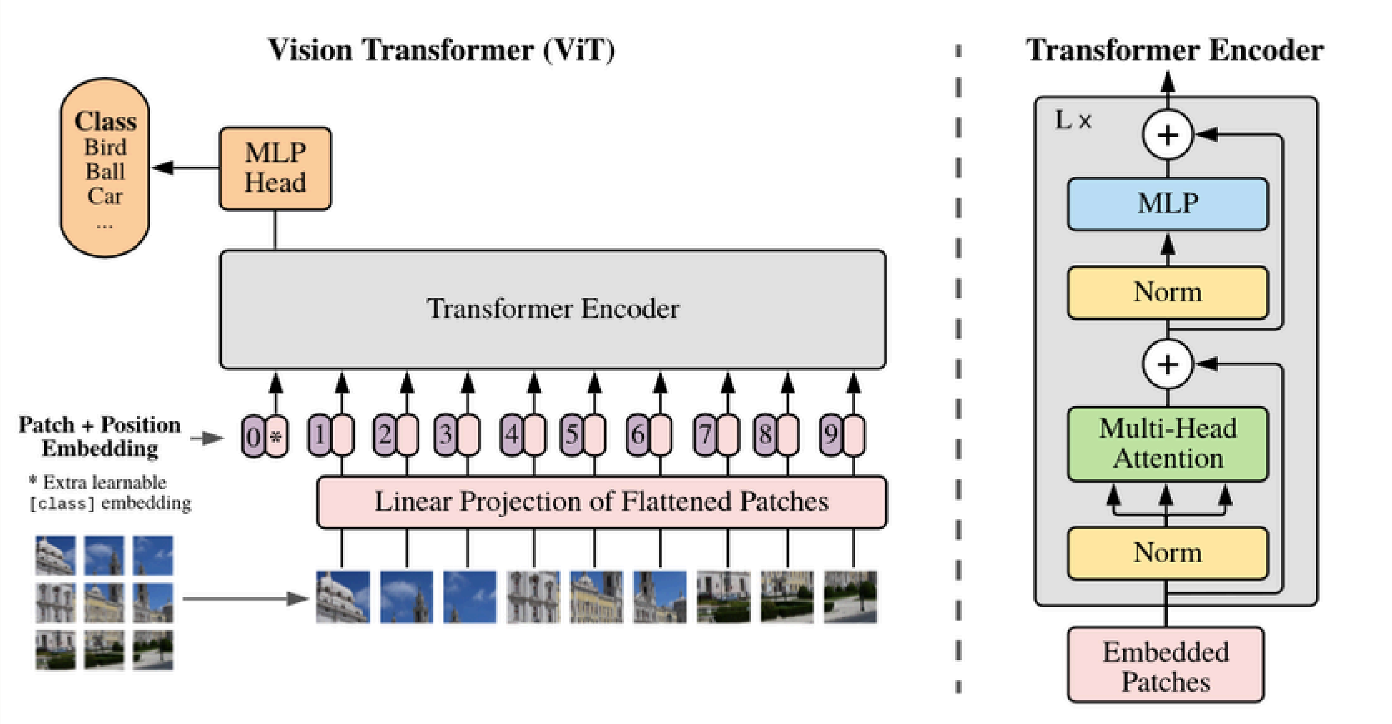
Minimalize visual embedding process(Patch Projection).

Follows single-stream approach.

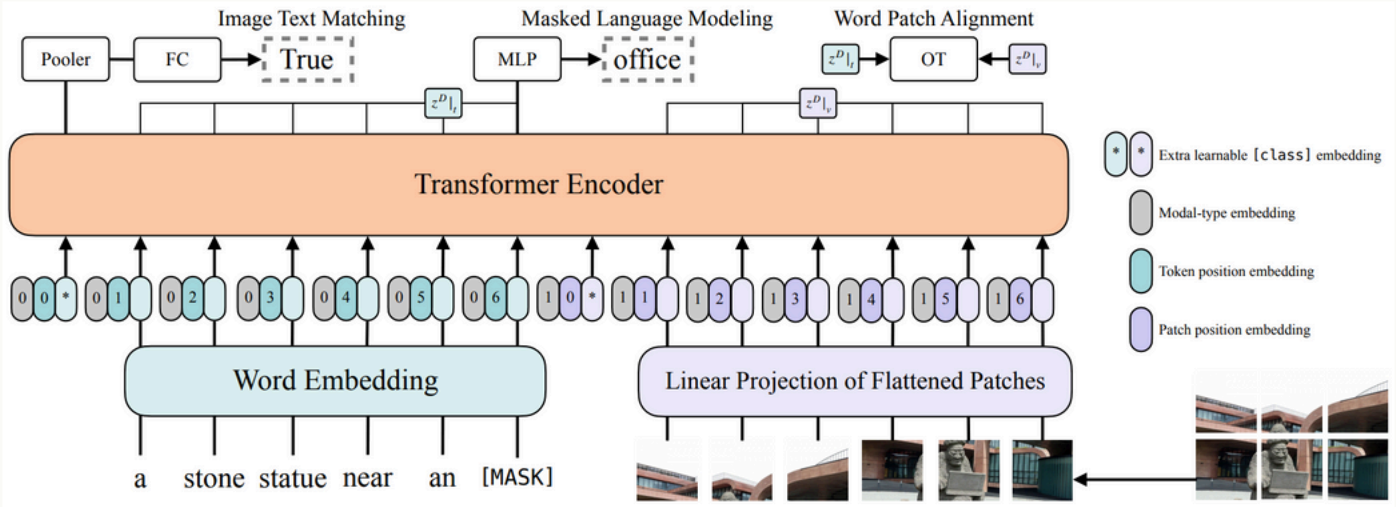
ViLT initializes with the weights of ViT.

Commonality

- Weights
- Hyperparameter
- Configuration



ViT Architecture



ViLT Architecture

Image source : https://arxiv.org/abs/2010.11929

Written by Kim EunHo

VISUAL AND LANGUAGE TRANSFORMER

ViLT / ViT Transformer Code

```
class VisionTransformer(nn.Module):
    def __init__(
        self,
        embed_dim=768,
        depth=12,
        num_heads=12,
        mlp_ratio=4.0,
        attn_drop_rate=0.0, ...):
        ...

class Block(nn.Module):
    def __init__(
        self,
        dim,
        num_heads,
        mlp_ratio=4.0,
        attn_drop=0.0, ...):
        ...
        self.attn = Attention(
            dim,
            num_heads=num_heads,
            attn_drop=attn_drop, ...)
        ...
        mlp_hidden_dim = int(dim * mlp_ratio)
        self.mlp = Mlp(
            in_features=dim,
            hidden_features=mlp_hidden_dim, ...)
        ...
```

ViLT Architecture

```
def get_b16_config():
  """Returns the ViT-B/16 configuration."""
  ...
  config.model_name = 'ViT-B_16'
  config.transformer.num_heads = 12
  config.transformer.num_layers = 12
  config.hidden_size = 768
  config.transformer.mlp_dim = 3072
  config.transformer.attention_dropout_rate = 0.0
  ...
  return config

def get_b32_config():
  """Returns the ViT-B/32 configuration."""
  config = get_b16_config()
  config.model_name = 'ViT-B_32'
  config.patches.size = (32, 32)
  return config
```

ViT Architecture

Written by Kim EunHo

MODALITY INTERACTION SCHEMA

Two approaches

Single-stream approach
- Use single encoder
- Simple and low cost
- Have limitation.

Dual-stream approach
- Use dual encoder
- Good performance
- Complex and high cost.

Dual-stream needs additional parameters(dual encoder), so ViLT follows single-stream approach.

MODEL OVERVIEW

ViLT Formula

$$\bar{t} = [t_{\text{class}}; t_1 T; \cdots; t_L T] + T^{\text{pos}} \tag{1}$$

$$\bar{v} = [v_{\text{class}}; v_1 V; \cdots; v_N V] + V^{\text{pos}} \tag{2}$$

$$z^0 = [\bar{t} + t^{\text{type}}; \bar{v} + v^{\text{type}}] \tag{3}$$

$$\hat{z}^d = \text{MSA}(\text{LN}(z^{d-1})) + z^{d-1}, \qquad d = 1 \ldots D \tag{4}$$

$$z^d = \text{MLP}(\text{LN}(\hat{z}^d)) + \hat{z}^d, \qquad d = 1 \ldots D \tag{5}$$

$$p = \tanh(z_0^D W_{\text{pool}}) \tag{6}$$

ViLT Formula

(1) : Word Embedding and Position Embedding for input text.

(2) : Patch Projection and Position Embedding for input image.

(3) : Sum with their corresponding modal-type embedding vectors and concatenate.

MODEL OVERVIEW

ViLT Formula

$$\bar{t} = [t_{\text{class}}; t_1 T; \cdots ; t_L T] + T^{\text{pos}} \quad (1)$$

$$\bar{v} = [v_{\text{class}}; v_1 V; \cdots ; v_N V] + V^{\text{pos}} \quad (2)$$

$$z^0 = [\bar{t} + t^{\text{type}}; \bar{v} + v^{\text{type}}] \quad (3)$$

$$\hat{z}^d = \text{MSA}(\text{LN}(z^{d-1})) + z^{d-1}, \qquad d = 1 \ldots D \quad (4)$$

$$z^d = \text{MLP}(\text{LN}(\hat{z}^d)) + \hat{z}^d, \qquad d = 1 \ldots D \quad (5)$$

$$p = \tanh(z_0^D W_{\text{pool}}) \quad (6)$$

ViLT Formula

d = Index of transformer layer
D = Final transformer layer

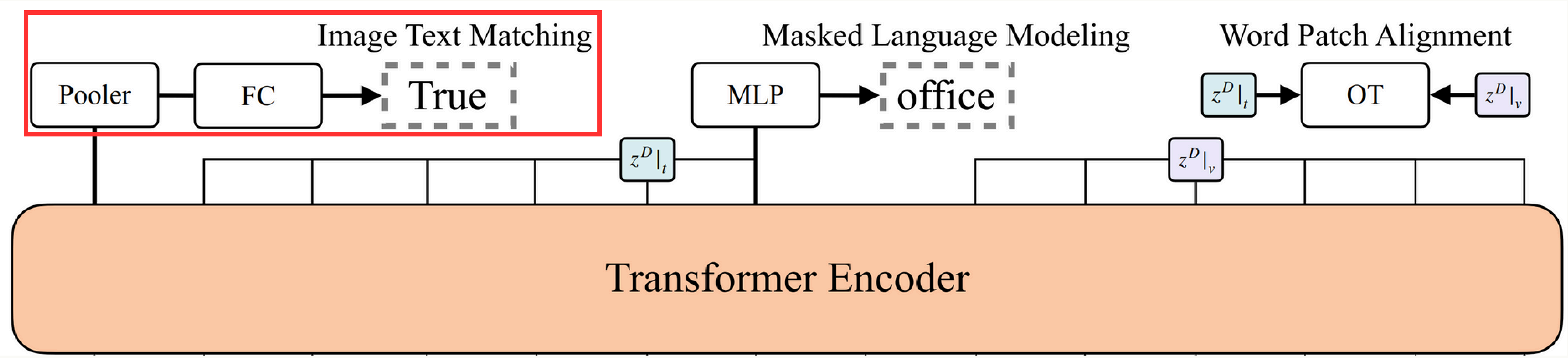
(4) : Apply Layer Normalization, MSA, and Residual calculation to z^{d-1}.

(5) : Apply Layer Normalization, MLP and Residual calucation to $\hat{z}^d$.

(6) : Multiply final output and weight matrix, then pass through activation function.

PRE-TRAINING OBJECTIVES

Image-Text Matching(ITM)



ViLT Architecture

Model predicts → image-text pair is a "match(1)" or "mismatch(0)"(Binary classification).

Before predict, randomly replace the image with a different image with the 0.5 probability.
(If image has changed → “mismatch(0)”, otherwise “match(1)”)

Calculate the negative log-likelihood loss with logit p as the ITM Loss.

PRE-TRAINING OBJECTIVES

Image-Text Matching(ITM) Code

```python
def compute_itm_wpa(pl_module, batch):
    pos_len = len(batch["text"]) // 2
    neg_len = len(batch["text"]) - pos_len

    itm_labels = torch.cat([torch.ones(pos_len), torch.zeros(neg_len)]).to(
        pl_module.device
    )
    itm_labels = itm_labels[torch.randperm(itm_labels.size(0))]
    infer = pl_module.infer(batch, mask_text=False, mask_image=False)

    itm_logits = pl_module.itm_score(infer["cls_feats"])
    itm_loss = F.cross_entropy(itm_logits, itm_labels.long())

    ret = {
        "itm_loss": itm_loss, ...}

    return ret
```

ITM Loss Code

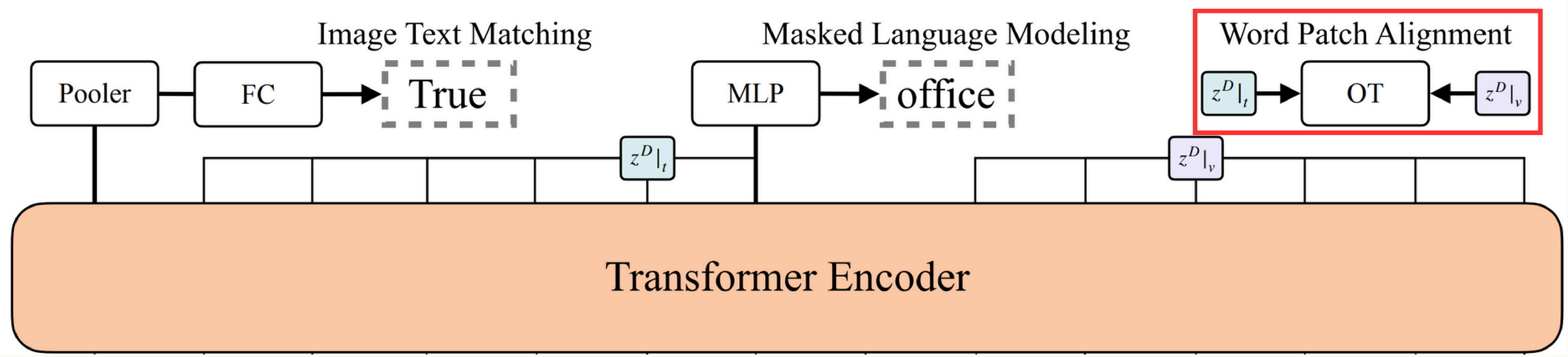
Define the number of pos_len, neg_len.

Add 1 to positive samples and 0 to negative samples, mix, and inference.

Apply cross_entropy to the inference result to obtain ITM Loss and return it.

PRE-TRAINING OBJECTIVES

Word Patch Alignment(WPA)



ViLT Architecture

WPA learn alignment between image patches and text tokens with IPOT.

IPOT? - Techniques used to optimize the relationship between text and images.

PRE-TRAINING OBJECTIVES

Word Patch Alignment(WPA) Code

```python
def compute_itm_wpa(pl_module, batch):
    with torch.cuda.amp.autocast(enabled=False):
        ...
        cost = cost_matrix_cosine(txt_emb.float(), img_emb.float())
        joint_pad = txt_pad.unsqueeze(-1) | img_pad.unsqueeze(-2)
        cost.masked_fill_(joint_pad, 0)

        txt_len = (txt_pad.size(1) - txt_pad.sum(dim=1, keepdim=False)).to(
            dtype=cost.dtype
        )
        img_len = (img_pad.size(1) - img_pad.sum(dim=1, keepdim=False)).to(
            dtype=cost.dtype
        )
        T = ipot(
            cost.detach(), txt_len, txt_pad, img_len, img_pad, joint_pad, 0.5, 50, 1
        )
        distance = trace(cost.matmul(T.detach()))

    dist_pos = distance.masked_select(itm_labels == 1)
    dist_neg = distance.masked_select(itm_labels == 0)
    ot_loss = (dist_pos.sum() - dist_neg.sum()) / (dist_pos.size(0) + dist_neg.size(0))

    ret = {
        "itm_wpa_loss": 0.1 * ot_loss, ... }

    return ret
```

WPA Loss Code

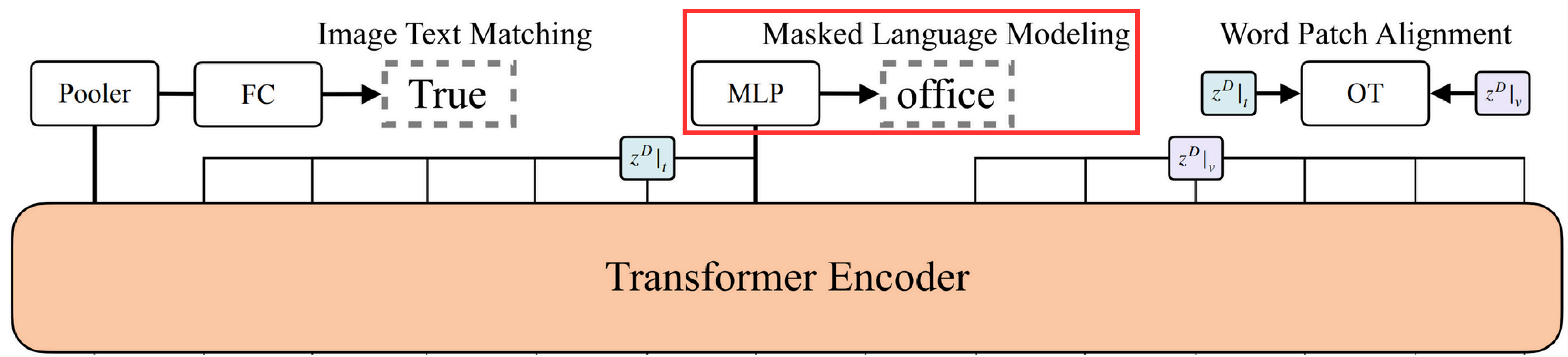
Calcuate cost and obtain joint padding mask.

Find the length of the sequence and calculate the IPOT.

Compute a WAP Loss function based on the distance difference between positive and negative samples and return it.

PRE-TRAINING OBJECTIVES

Masked Language Modeling(MLM)



ViLT Architecture

Mask a text token(0.15) and predict the Ground Truth label of the masked text token.

The MLM loss is computed as the negative log-likelihood loss for the masked tokens.

PRE-TRAINING OBJECTIVES

Masked Language Modeling(MLM) Code

```
def compute_mlm(pl_module, batch):
    infer = pl_module.infer(batch, mask_text=True, mask_image=False)

    mlm_logits = pl_module.mlm_score(infer["text_feats"])
    mlm_labels = infer["text_labels"]

    mlm_loss = F.cross_entropy(
        mlm_logits.view(-1, pl_module.hparams.config["vocab_size"]),
        mlm_labels.view(-1),
        ignore_index=-100,
    )

    ret = {
        "mlm_loss": mlm_loss, ...}

    return ret
```

MLM Loss Code

Only mask text data.

Find logit and labels based on masked text data.

Reshape the logit and label, and apply cross entropy to obtain the MLM Loss.

ANOTHER TECHNIQUES

Whole-word Masking

Masks all consecutive subword tokens by BERT tokenizer(0.15).
(ex. “giraffe” → [“gi”], [“##raf”], [“##fe”] → [MASK], [MASK], [MASK])

Applied to original and Chinese BERT.

If you do not mask all tokens, only use language token and don’t use image information.

ANOTHER TECHNIQUES

Image Augmentation

Increase the generalization power of a vision model.

RandAugment was applied, but “Color inversion” and “Cutout” were not applied.

Because they may damage or eliminate information included in the textual information.

EXPERIMENTS

Comparison of VLP Models and ViLT

Visual Embed	Model	#Params (M)	#FLOPs (G)	Time (ms)
Region	ViLBERT^{36+36}	274.3	958.1	~900
	VisualBERT^{36+128}	170.3	425.0	~925
	LXMERT^{36+20}	239.8	952.0	~900
	UNITER-Base^{36+60}	154.7	949.9	~900
	OSCAR-Base^{50+35}	154.7	956.4	~900
	VinVL-Base^{50+35}	157.3	1023.3	~650
	Unicoder-VL$^{100+?}$	170.3	419.7	~925
	ImageBERT^{100+44}	170.3	420.6	~925
Grid	Pixel-BERT-X152$^{146+?}$	144.3	185.8	~160
	Pixel-BERT-R50$^{260+?}$	94.9	136.8	~60
Linear	ViLT-B/32^{200+40}	87.4	55.9	~15

Params, FLOPs, Time of VLP Models

Visual Embed	Model	CNN Backbone	RoI Head	NMS	Trans. Layers
Region	ViLBERT	R101	C4	PC	~15
	VisualBERT	X152	FPN	PC	12
	LXMERT	R101	C4	PC	~12
	UNITER-Base	R101	C4	PC	12
	OSCAR-Base	R101	C4	PC	12
	VinVL-Base	X152	C4	CA	12
	Unicoder-VL	X152	FPN	PC	12
	ImageBERT	X152	FPN	PC	12
Grid	Pixel-BERT-X152	X152	-	-	12
	Pixel-BERT-R50	R50	-	-	12
Linear	ViLT-B/32	-	-	-	12

Component of VLP Models

C4 : Conv4

PC : Per-class method

FPN : Feature Pyramid Network

CA : Class-agnostic method

EXPERIMENTS

Performance comparison by Visual Embed method

Visual Embed	Model	Time (ms)	VQAv2 test-dev	NLVR2 dev	NLVR2 test-P
Region	w/o VLP SOTA	~900	70.63	54.80	53.50
	ViLBERT	~920	70.55	-	-
	VisualBERT	~925	70.80	67.40	67.00
	LXMERT	~900	72.42	74.90	74.50
	UNITER-Base	~900	72.70	75.85	75.80
	OSCAR-Base†	~900	73.16	78.07	78.36
	VinVL-Base†‡	~650	75.95	82.05	83.08
Grid	Pixel-BERT-X152	~160	74.45	76.50	77.20
	Pixel-BERT-R50	~60	71.35	71.70	72.40
Linear	ViLT-B/32	~15	70.33	74.41	74.57
	ViLT-B/32ⓐ	~15	70.85	74.91	75.57
	ViLT-B/32ⓐ⊕	~15	71.26	75.70	76.13

Performance comparison

ViLT is fastest but performance is similar with other VLP Models.

EXPERIMENTS

Performance measurment

Visual Embed	Model	Time (ms)	Zero-Shot Text Retrieval Flickr30k (1K) R@1	R@5	R@10	MSCOCO (5K) R@1	R@5	R@10	Zero-Shot Image Retrieval Flickr30k (1K) R@1	R@5	R@10	MSCOCO (5K) R@1	R@5	R@10
Region	ViLBERT	~900	-	-	-	-	-	-	31.9	61.1	72.8	-	-	-
	Unicoder-VL	~925	64.3	85.8	92.3	-	-	-	48.4	76.0	85.2	-	-	-
	UNITER-Base	~900	80.7	95.7	98.0	-	-	-	66.2	88.4	92.9	-	-	-
	ImageBERT†	~925	70.7	90.2	94.0	44.0	71.2	80.4	54.3	79.6	87.5	32.3	59.0	70.2
Linear	ViLT-B/32	~15	69.7	91.0	96.0	53.4	80.7	88.8	51.3	79.9	87.9	37.3	67.4	79.0
	ViLT-B/32⊕	~15	73.2	93.6	96.5	56.5	82.6	89.6	55.0	82.5	89.8	40.4	70.0	81.1

Performance measurement in Zero-Shot Text Retrieval

Visual Embed	Model	Time (ms)	Text Retrieval Flickr30k (1K) R@1	R@5	R@10	MSCOCO (5K) R@1	R@5	R@10	Image Retrieval Flickr30k (1K) R@1	R@5	R@10	MSCOCO (5K) R@1	R@5	R@10
Region	w/o VLP SOTA	~900	67.4	90.3	95.8	50.4	82.2	90.0	48.6	77.7	85.2	38.6	69.3	80.4
	ViLBERT-Base	~920	-	-	-	-	-	-	58.2	84.9	91.5	-	-	-
	Unicoder-VL	~925	86.2	96.3	99.0	62.3	87.1	92.8	71.5	91.2	95.2	48.4	76.7	85.9
	UNITER-Base	~900	85.9	97.1	98.8	64.4	87.4	93.1	72.5	92.4	96.1	50.3	78.5	87.2
	OSCAR-Base†	~900	-	-	-	70.0	91.1	95.5	-	-	-	54.0	80.8	88.5
	VinVL-Base†‡	~650	-	-	-	74.6	92.6	96.3	-	-	-	58.1	83.2	90.1
Grid	Pixel-BERT-X152	~160	87.0	98.9	99.5	63.6	87.5	93.6	71.5	92.1	95.8	50.1	77.6	86.2
	Pixel-BERT-R50	~60	75.7	94.7	97.1	59.8	85.5	91.6	53.4	80.4	88.5	41.1	69.7	80.5
Linear	ViLT-B/32	~15	81.4	95.6	97.6	61.8	86.2	92.6	61.9	86.8	92.8	41.3	72.0	82.5
	ViLT-B/32ⓐ	~15	83.7	97.2	98.1	62.9	87.1	92.7	62.2	87.6	93.2	42.6	72.8	83.4
	ViLT-B/32ⓐ⊕	~15	83.5	96.7	98.6	61.5	86.3	92.7	64.4	88.7	93.8	42.7	72.9	83.1

Performance measurement in Text Retrieval

ViLT is also have similar performance in downstream task.

ABLATION STUDY

ViLT-B/32

Training	Ablation			VQAv2	NLVR2		Flickr30k R@1 (1K)		MSCOCO R@1 (5K)	
Steps	w	m	a	test-dev	dev	test-P	TR (ZS)	IR (ZS)	TR (ZS)	IR (ZS)
25K	X	X	X	68.96 ± 0.07	70.83 ± 0.19	70.83 ± 0.23	75.39 (45.12)	52.52 (31.80)	53.72 (31.55)	34.88 (21.58)
50K	X	X	X	69.80 ± 0.01	71.93 ± 0.27	72.92 ± 0.82	78.13 (55.57)	57.36 (40.94)	57.00 (39.56)	37.47 (27.51)
100K	X	X	X	70.16 ± 0.01	73.54 ± 0.02	74.15 ± 0.27	79.39 (66.99)	60.50 (47.62)	60.15 (51.25)	40.45 (34.59)
100K	O	X	X	70.33 ± 0.01	74.41 ± 0.21	74.57 ± 0.09	81.35 (69.73)	61.86 (51.28)	61.79 (53.40)	41.25 (37.26)
100K	O	O	X	70.21 ± 0.05	72.76 ± 0.50	73.54 ± 0.47	78.91 (63.67)	58.76 (46.96)	59.53 (47.75)	40.08 (32.28)
100K	O	X	O	70.85 ± 0.13	74.91 ± 0.29	75.57 ± 0.61	83.69 (69.73)	62.22 (51.28)	62.88 (53.40)	42.62 (37.26)
200K	O	X	O	71.26 ± 0.06	75.70 ± 0.32	76.13 ± 0.39	83.50 (73.24)	64.36 (54.96)	61.49 (56.51)	42.70 (40.42)

Ablation Study of ViLT-B/32 Model

w Whole-Word Masking

m Masked Patch Prediction as objective

a Image Augmentation

More Training Steps, Whole-Word Masking and Image Augmentation are benefical, but Masked Patch Prediction is not.

Q&A