NAVER AI LAB

Self-Supervised Vision-and-Language Pre-Training

Wonjae Kim | 20230112 | @HUST

1. SSL Methods
    1-1. Vision SSL Methods
    1-2. Language SSL Methods
    1-3. Considerations for SSL Methods
2. Vision-and-Language
    2-1. Evolution of VLP Models
    2-2. Considerations for VLP Models

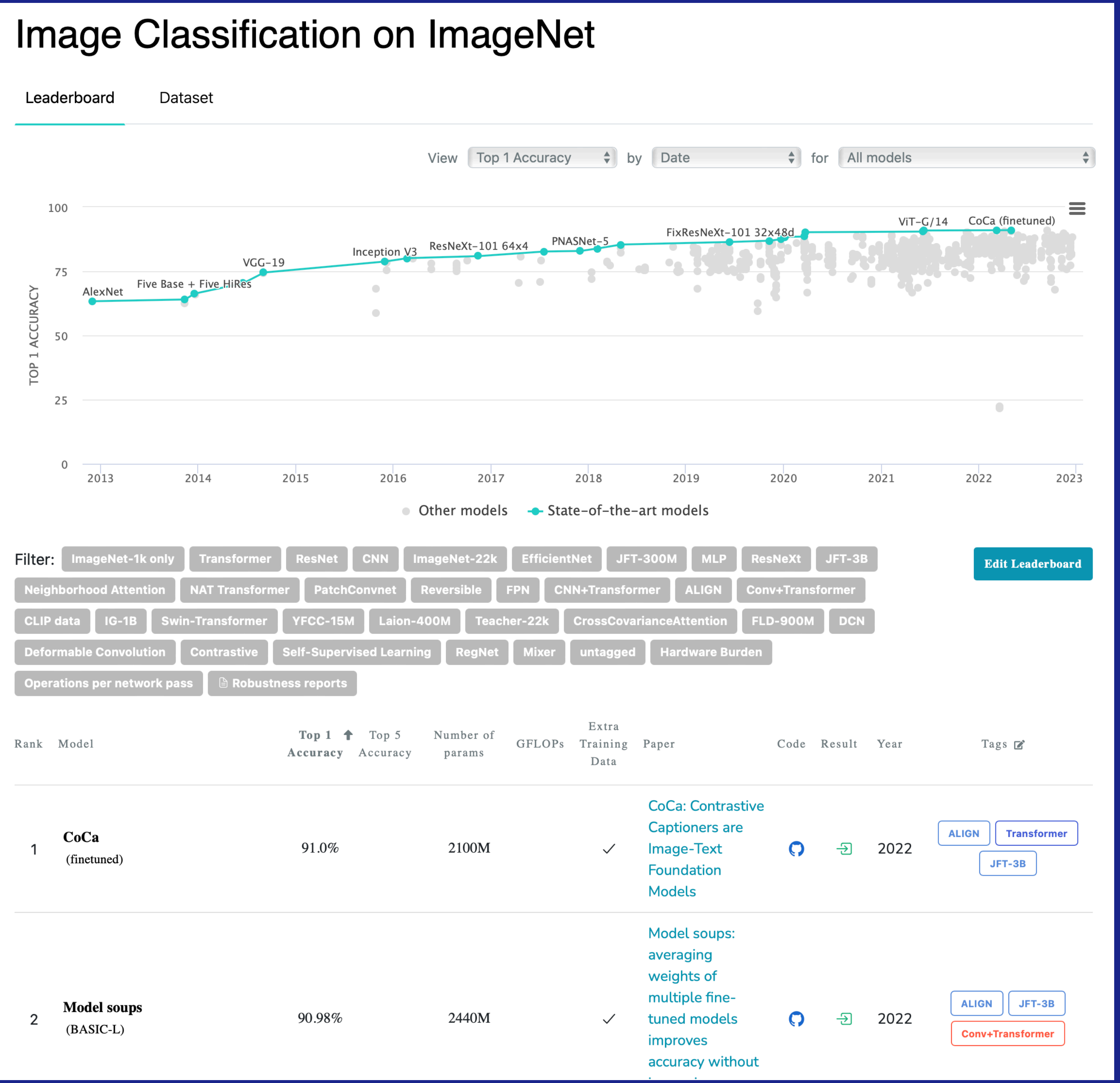
Good Old Pre-Training

```
def good_old_pretraining(weights, images, labels):
    """
    weights: neural net model weights
    images: iterable batches of images
    labels: iterable labels paired with images
    """
    while not done:
        logits = model_forward(weights, images)
        loss = cross_entropy(logits, labels)
        weights = optimize(weights, loss)
    return weights
```

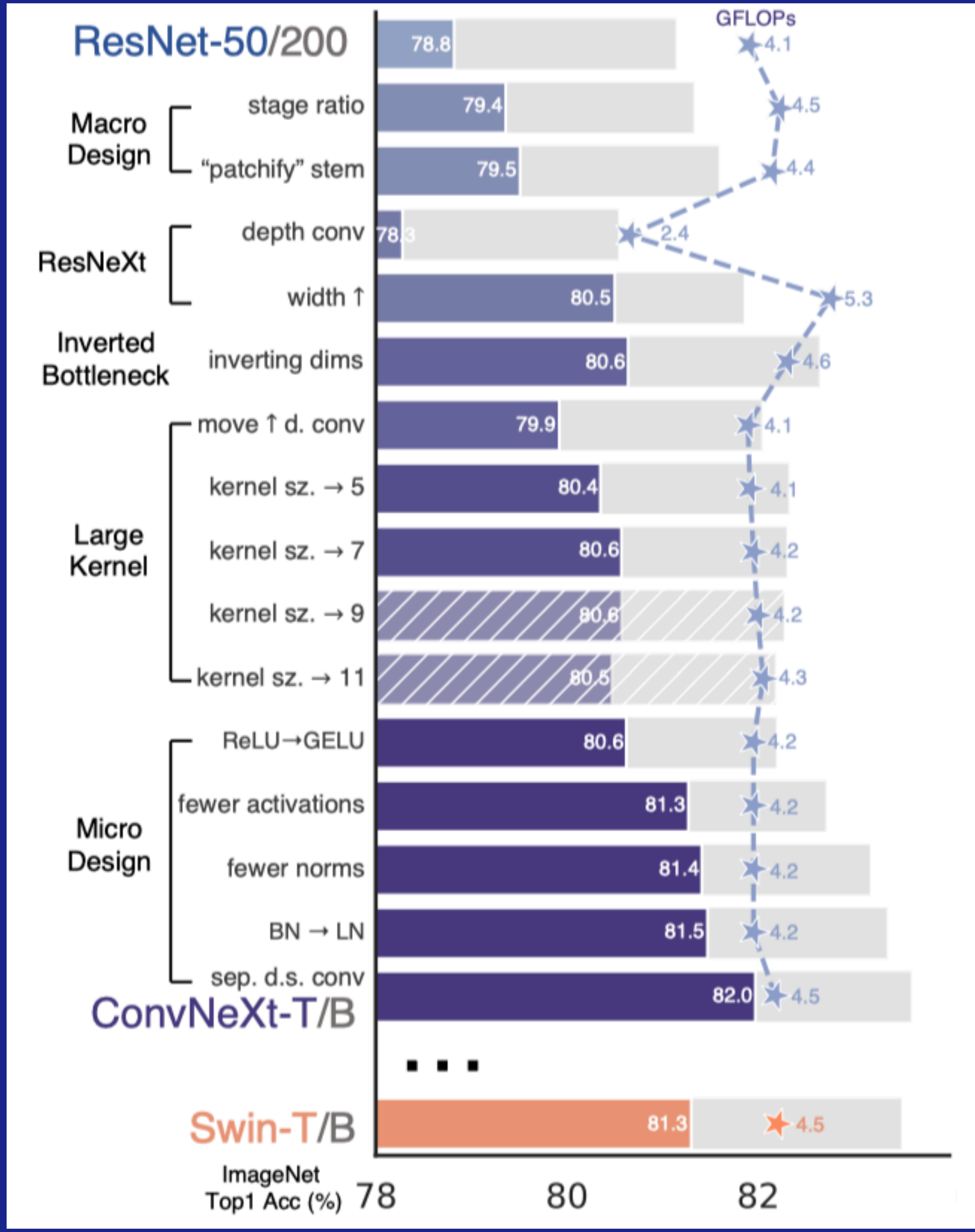
imagenet-1k[1], JFT-300M[2], Instagram-1B[3]

1. Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition.
2. Ngiam, Jiquan et al. "Domain Adaptive Transfer Learning with Specialist Models." ArXiv abs/1811.07056 (2018).
3. Yalniz, Ismet Zeki et al. "Billion-scale semi-supervised learning for image classification." ArXiv abs/1905.00546 (2019).

NAVER
AI LAB



Rank	Model	Top 1 Accuracy	Top 5 Accuracy	Number of params	GFLOPs	Extra Training Data	Paper	Code	Result	Year	Tags
1	CoCa (finetuned)	91.0%		2100M		✓	CoCa: Contrastive Captioners are Image-Text Foundation Models			2022	ALIGN, Transformer, JFT-3B
2	Model soups (BASIC-L)	90.98%		2440M		✓	Model soups: averaging weights of multiple fine-tuned models improves accuracy without			2022	ALIGN, JFT-3B, Conv+Transformer



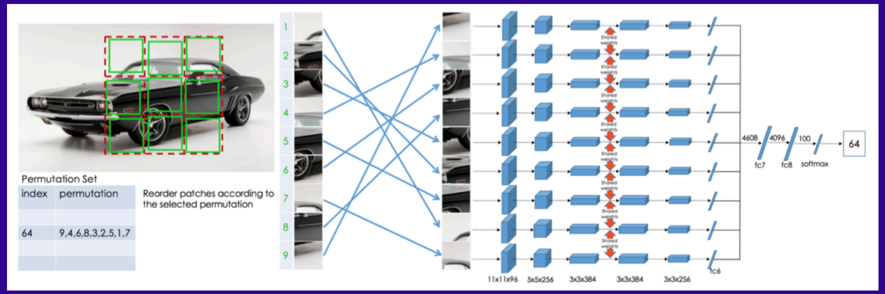
So why SSL in vision?

Pretext Modeling[1]

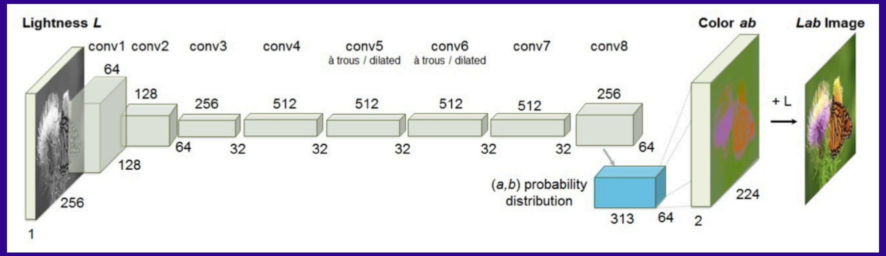
```
def pretext_modeling(weights, images):
    """
    distort: a function distorts images and returns distorted images and the description
of the distortion.
    """
    while not done:
        distorted_images, labels = distort(images)
        logits = model_forward(weights, distorted_images)
        loss = cross_entropy(logits, labels)
        weights = optimize(weights, loss)
    return weights
```

1. Misra, Ishan, and Laurens van der Maaten. "Self-supervised learning of pretext-invariant representations." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

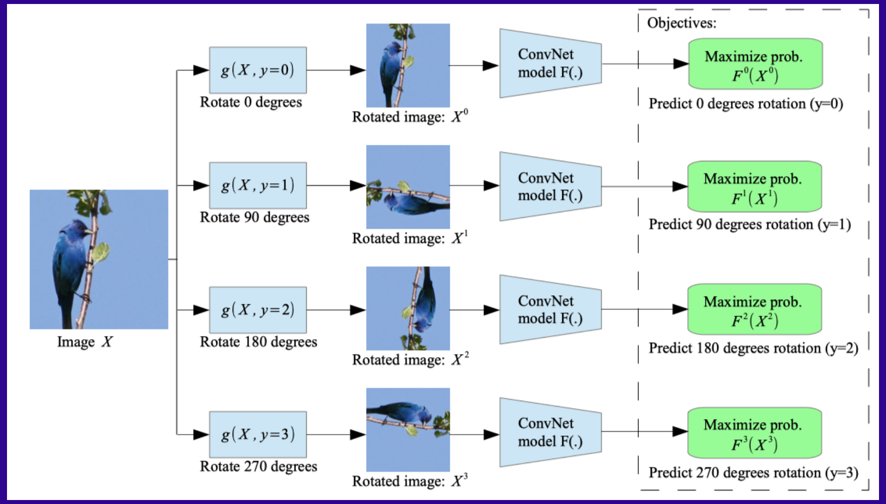
Jigsaw[1]



Colorization[2]



Rotation[3]



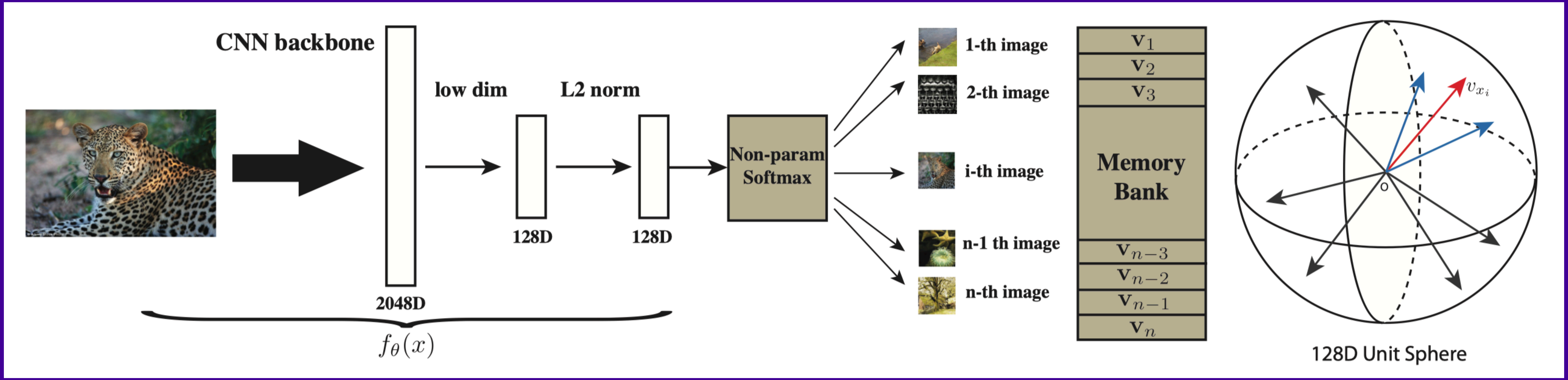
1. Noroozi, Mehdi and Paolo Favaro. "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles." ECCV (2016).
2. Zhang, Richard et al. "Colorful Image Colorization." ECCV (2016).
3. Gidaris, Spyros et al. "Unsupervised Representation Learning by Predicting Image Rotations." ICLR (2018).

Contrastive Learning[1]

```
def contrastive_learning(weights, images):
    """
    random_view: "view" in contrastive learning context is a fancy way of calling augmentation, typically includes random 
resized crop (RRC).
    """
    while not done:
        images1, images2 = random_view(images, n=2)
        z1 = model_forward(weights, images1, normalize=True)
        z2 = model_forward(weights, images2, normalize=True)
        logits1 = z1 @ z2.t() / tau
        logits2 = z2 @ z1.t() / tau
        labels = arange(batch_size)
        loss = cross_entropy(logits1, labels) + cross_entropy(logits2, labels)
        weights = optimize(weights, loss)
    return weights
```

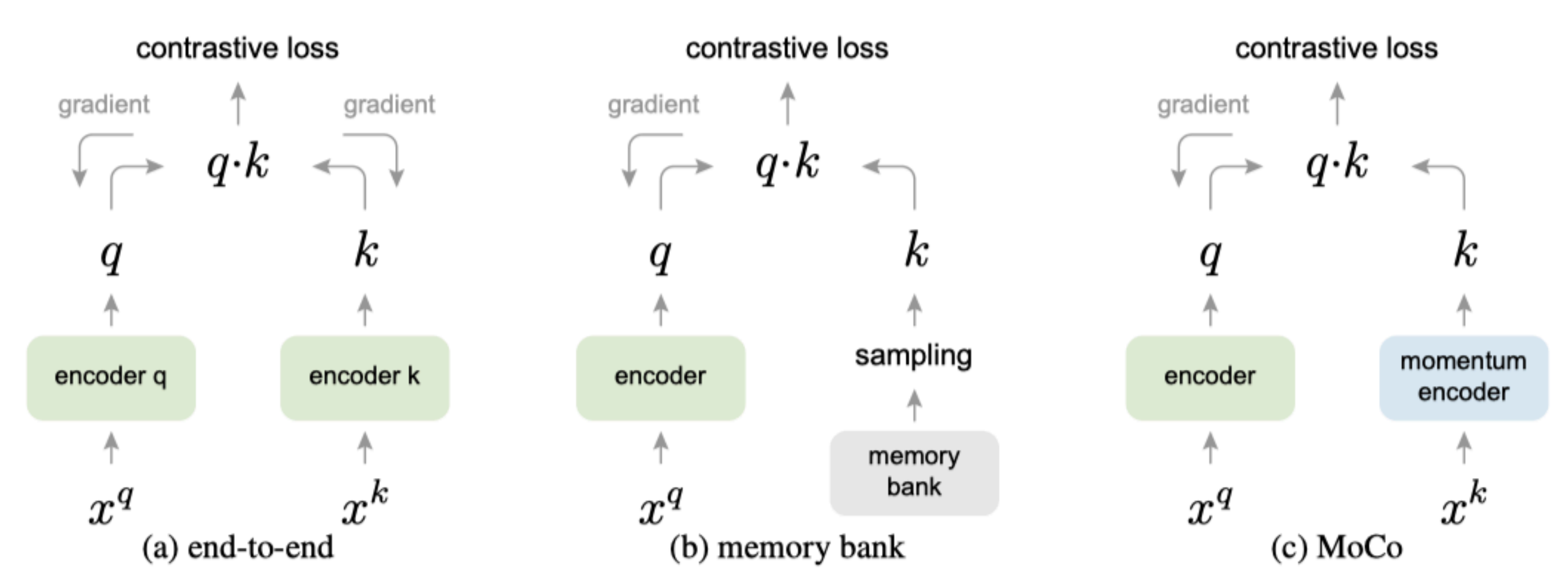

1. Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." arXiv preprint arXiv:1807.03748 (2018).

Instance Discrimination[1]



1. Wu, Zhirong, et al. "Unsupervised feature learning via non-parametric instance discrimination." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

Momentum Contrast[1]



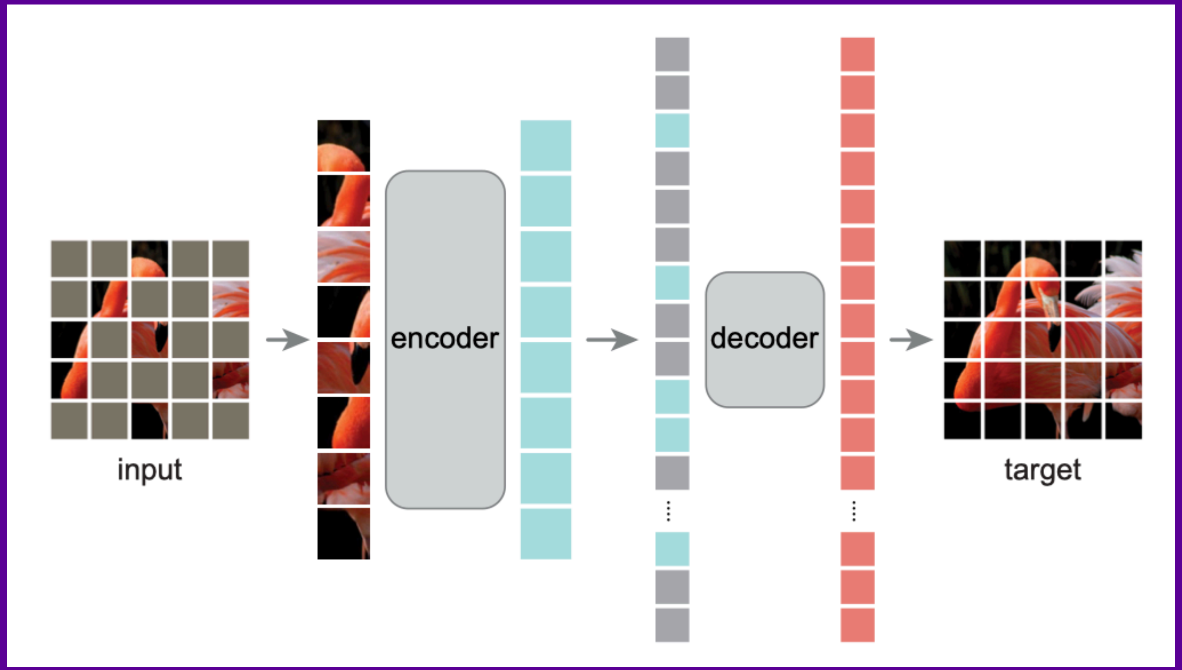
1. He, Kaiming, et al. "Momentum contrast for unsupervised visual representation learning." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

Masked Image Modeling[1]

```
def masked_image_modeling(weights, images):
    """
    images: iterable tokenized batches of images
    VQ-VAE tokenizer for BEiT, patch tokenizer for MAE, iBOT; SplitMask tried multiple tokenizers.
    """
    while not done:
        masked_images, targets = mask_images(images, prob=prob_value)
        logits = model_forward(weights, masked_images)
        loss = cross_entropy(logits, targets) if discrete else regression(logits, targets)
        weights = optimize(weights, loss)
    return weights
```

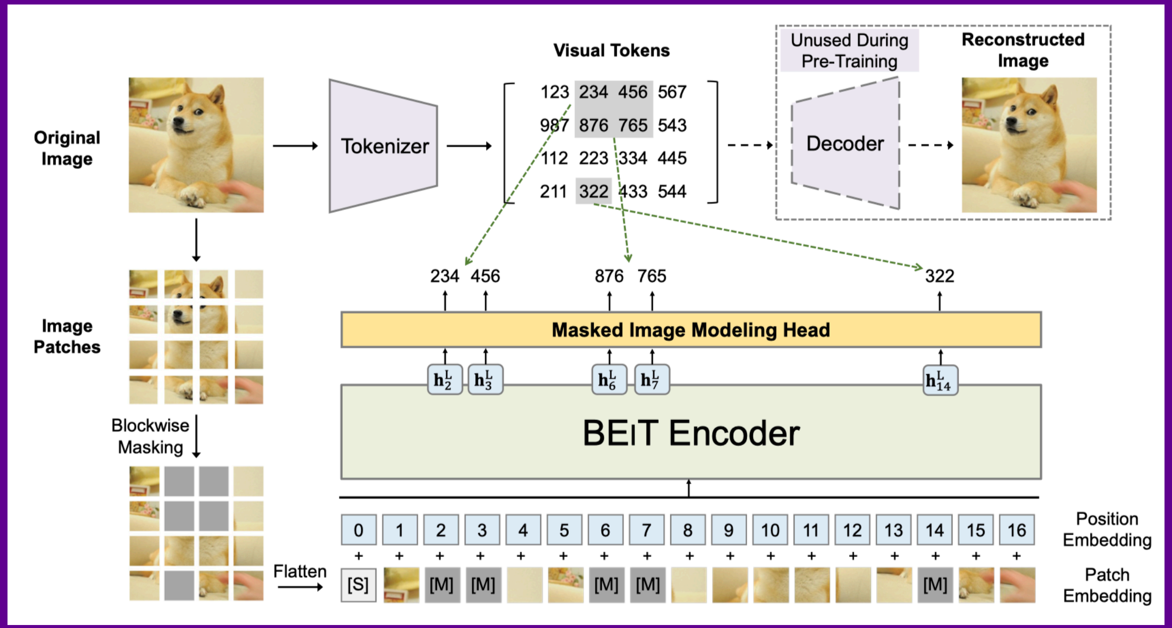
1. Xie, Zhenda, et al. "Simmim: A simple framework for masked image modeling." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

Masked Autoencoder[1]



1. He, Kaiming et al. "Masked Autoencoders Are Scalable Vision Learners." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

BEiT[1]



1. Bao, Hangbo, et al. "BEiT: BERT Pre-Training of Image Transformers." International Conference on Learning Representations. 2021.

The Home of SSL: Language

NAVER AI LAB

Deep Contextualized Word Representations[1]

```
def language_modeling_elmo(weights, texts):
    """
    texts: iterable tokenized batches of texts
    """
    while not done:
        logits = model_forward(weights, texts)
        reverse_logits = model_forward(weights, texts[::-1])
        loss = cross_entropy(logits[1:], texts[:-1])
        reverse_loss = cross_entropy(reverse_logits[1:], texts[:-1][::-1])
        weights = optimize(weights, loss + reverse_loss)
    return weights
```

1. Peters, Matthew E. et al. “Deep Contextualized Word Representations.” NAACL (2018).

NAVER AI LAB

Generative Pre-Training[123]

```
def language_modeling_gpt(weights, texts):
"""
texts: iterable tokenized batches of texts
"""
    while not done:
        causal_mask = lower_triangle(texts.length)
        logits = model_forward(weights, texts, attention_mask=causal_mask)
        loss = cross_entropy(logits[1:], texts[:-1])
        weights = optimize(weights, loss)
    return weights
```


1. Radford, Alec and Karthik Narasimhan. “Improving Language Understanding by Generative Pre-Training.” (2018).
2. Radford, Alec et al. “Language Models are Unsupervised Multitask Learners.” (2019).
3. Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[1][2][3]

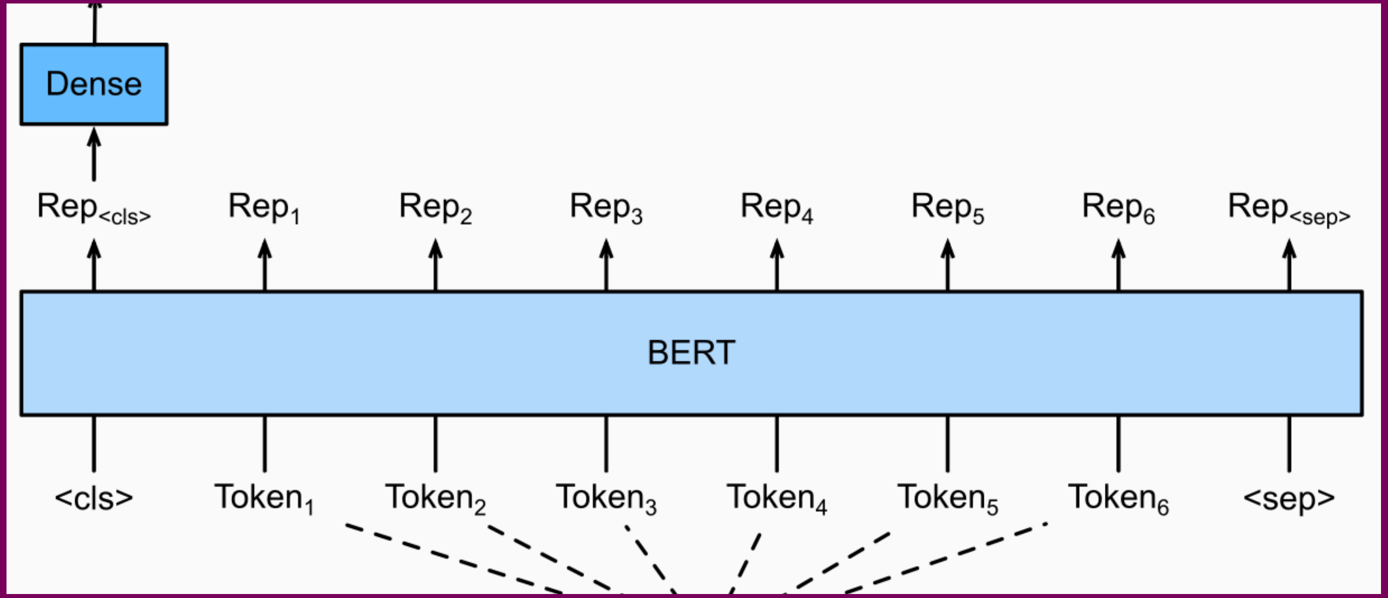
```
def language_modeling_bert(weights, texts):
"""
texts: iterable tokenized batches of texts
"""
    while not done:
        masked_texts, labels = mask_texts(texts, prob=0.15)
        logits = model_forward(weights, texts)
        loss = cross_entropy(logits, masked_texts)
        weights = optimize(weights, loss) return weights
```

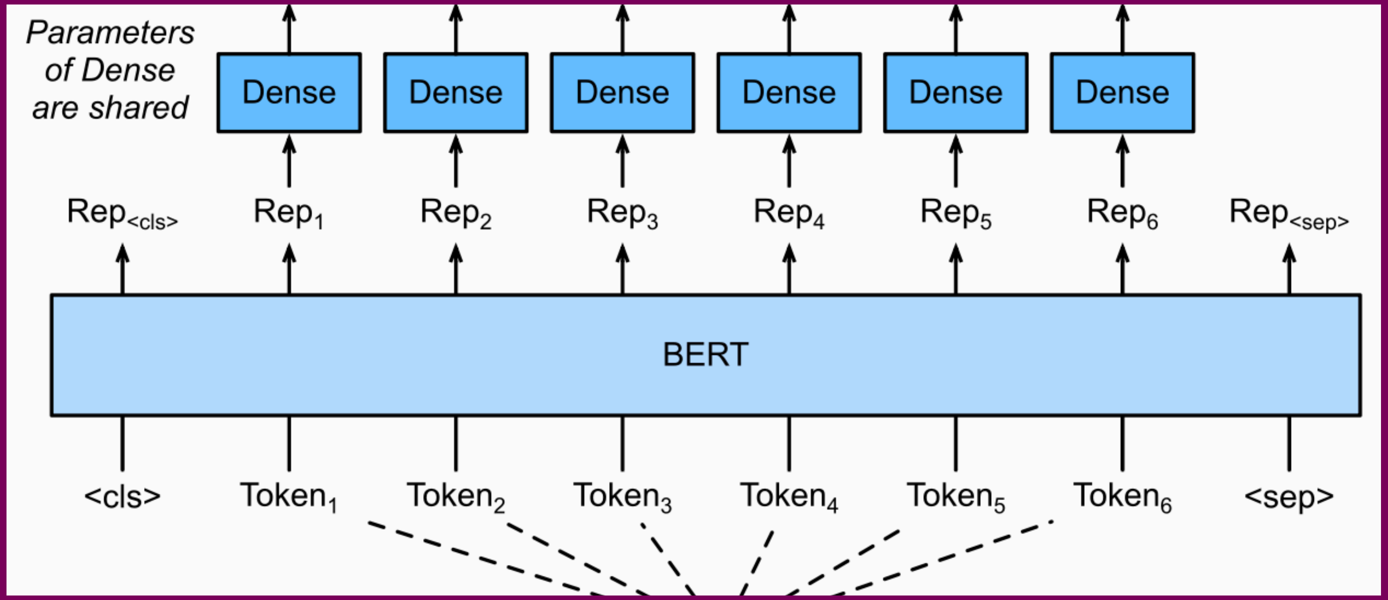

1. Kenton, Jacob Devlin Ming-Wei Chang, and Lee Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Proceedings of NAACL-HLT. 2019.
2. Liu, Yinhan, et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." (2019).
3. Lan, Zhenzhong, et al. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations." International Conference on Learning Representations. 2019.

Considerations for SSL Methods

1. What representations to use?
2. What metrics to use?
3. What parts to fine-tune?

1. What representations to use?



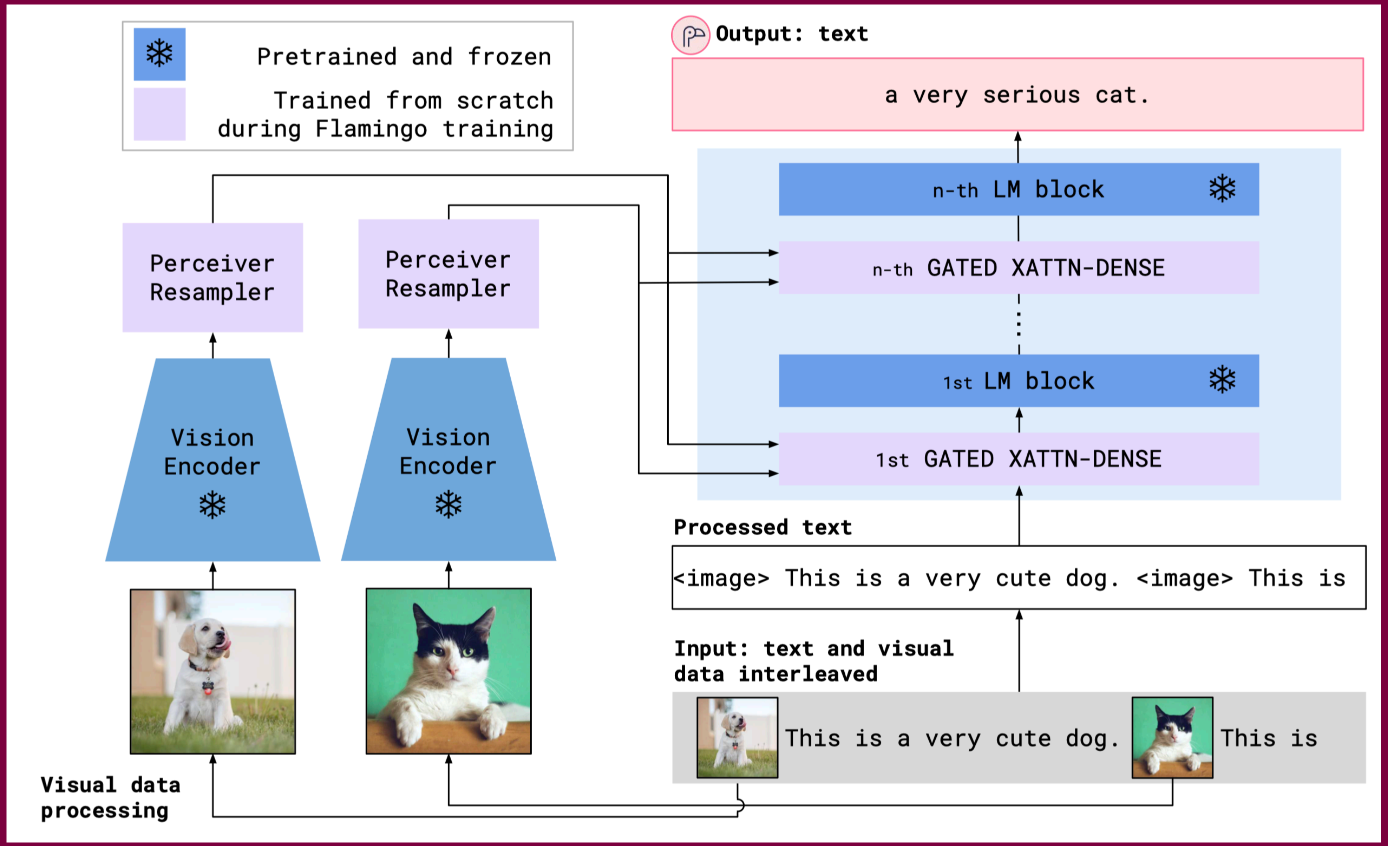


2. What metrics to use?[1]

		MIM 1	MLM 2	FLAVA_C 3	FLAVA_MM 4	FLAVA w/o init 5	FLAVA 6	CLIP 7	CLIP 8
Datasets	Eval method	PMD	PMD	PMD	PMD	(PMD+IN-1k+CCNews+BC)		PMD	400M [83]
MNLI [111]	fine-tuning	–	73.23	70.99	76.82	78.06	**80.33**	32.85	33.52
CoLA [110]	fine-tuning	–	39.55	17.58	38.97	44.22	**50.65**	11.02	25.37
MRPC [29]	fine-tuning	–	73.24	76.31	79.14	78.91	**84.16**	68.74	69.91
QQP [49]	fine-tuning	–	86.68	85.94	88.49	98.61	**88.74**	59.17	65.33
SST-2 [97]	fine-tuning	–	87.96	86.47	89.33	90.14	**90.94**	83.49	88.19
QNLI [88]	fine-tuning	–	82.32	71.85	84.77	86.40	**87.31**	49.46	50.54
RTE [7, 25, 36, 40]	fine-tuning	–	50.54	51.99	51.99	54.87	**57.76**	53.07	55.23
STS-B [1]	fine-tuning	–	78.89	57.28	84.29	83.21	**85.67**	13.70	15.98
NLP Avg.		–	71.55	64.80	74.22	75.55	**78.19**	46.44	50.50
ImageNet [90]	linear eval	41.79	–	74.09	74.34	73.49	**75.54**	72.95	80.20
Food101 [11]	linear eval	53.30	–	87.77	87.53	87.39	**88.51**	85.49	91.56
CIFAR10 [58]	linear eval	76.20	–	**93.44**	92.37	92.63	92.87	91.25	94.93
CIFAR100 [58]	linear eval	55.57	–	**78.37**	78.01	76.49	77.68	74.40	81.10
Cars [56]	linear eval	14.71	–	**72.12**	72.07	66.81	70.87	62.84	85.92
Aircraft [74]	linear eval	13.83	–	**49.74**	48.90	44.73	47.31	40.02	51.40
DTD [20]	linear eval	55.53	–	76.86	76.91	75.80	**77.29**	73.40	78.46
Pets [79]	linear eval	34.48	–	**84.98**	84.93	82.77	84.82	79.61	91.66
Caltech101 [32]	linear eval	67.36	–	94.91	95.32	94.95	**95.74**	93.76	95.51
Flowers102 [76]	linear eval	67.23	–	96.36	**96.39**	95.58	96.37	94.94	97.12
MNIST [60]	linear eval	96.40	–	98.39	98.58	**98.70**	98.42	97.38	99.01
STL10 [21]	linear eval	80.12	–	98.06	98.31	98.32	**98.89**	97.29	99.09
EuroSAT [41]	linear eval	95.48	–	97.00	96.98	97.04	**97.26**	95.70	95.38
GTSRB [100]	linear eval	63.14	–	78.92	77.93	77.71	**79.46**	76.34	88.61
KITTI [35]	linear eval	86.03	–	87.83	88.84	88.70	**89.04**	84.89	86.56
PCAM [106]	linear eval	85.10	–	85.02	85.51	**85.72**	85.31	83.99	83.72
UCF101 [98]	linear eval	46.34	–	82.69	82.90	81.42	**83.32**	77.85	85.17
CLEVR [52]	linear eval	61.51	–	79.35	**81.66**	80.62	79.66	73.64	75.89
FER 2013 [38]	linear eval	50.98	–	59.96	60.87	58.99	**61.12**	57.04	68.36
SUN397 [113]	linear eval	52.45	–	81.27	81.41	81.05	**82.17**	79.96	82.05
SST [83]	linear eval	57.77	–	56.67	**59.25**	56.40	57.11	56.84	74.68
Country211 [83]	linear eval	8.87	–	27.27	26.75	27.01	**28.92**	25.12	30.10
Vision Avg.		57.46	–	79.14	79.35	78.29	**79.44**	76.12	82.57
VQAv2 [39]	fine-tuning	–	–	67.13	71.69	71.29	**72.49**	59.81	54.83
SNLI-VE [114]	fine-tuning	–	–	73.27	78.36	78.14	**78.89**	73.53	74.27
Hateful Memes [53]	fine-tuning	–	–	55.58	70.72	**77.45**	76.09	56.59	63.93
Flickr30K [81] TR R@1	zero-shot	–	–	68.30	**69.30**	64.50	67.70	60.90	82.20
Flickr30K [81] TR R@5	zero-shot	–	–	93.50	92.90	90.30	**94.00**	88.90	96.60
Flickr30K [81] IR R@1	zero-shot	–	–	60.56	63.16	60.04	**65.22**	56.48	62.08
Flickr30K [81] IR R@5	zero-shot	–	–	86.68	87.70	86.46	**89.38**	83.60	85.68
COCO [66] TR R@1	zero-shot	–	–	43.08	**43.48**	39.88	42.74	37.12	52.48
COCO [66] TR R@5	zero-shot	–	–	75.82	**76.76**	72.84	**76.76**	69.48	76.68
COCO [66] IR R@1	zero-shot	–	–	37.59	**38.46**	34.95	38.38	33.29	33.07
COCO [66] IR R@5	zero-shot	–	–	67.28	**67.68**	64.63	67.47	62.47	58.37
Multimodal Avg.		–	–	66.25	69.11	67.32	**69.92**	62.02	67.29
Macro Avg.		19.15	23.85	70.06	74.23	73.72	**75.85**	61.52	66.78

1. Singh, Amanpreet, et al. "Flava: A foundational language and vision alignment model." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

3. What parts to fine-tune?[1]



1. Alayrac, Jean-Baptiste, et al. "Flamingo: a visual language model for few-shot learning." arXiv preprint arXiv:2204.14198 (2022).

Vision-and-Language[1]

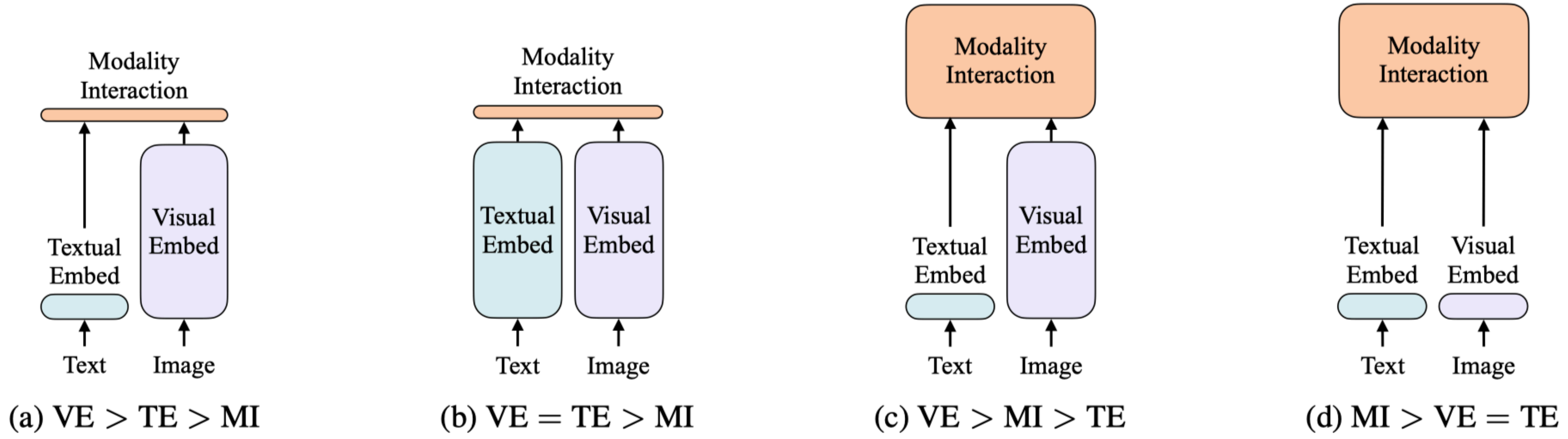


Figure 2. Four categories of vision-and-language models. The height of each rectangle denotes its relative computational size. VE, TE, and MI are short for visual embedder, textual embedder, and modality interaction, respectively.

1. Kim, Wonjae et al. "ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision." ICML (2021).

ViLBERT[1]

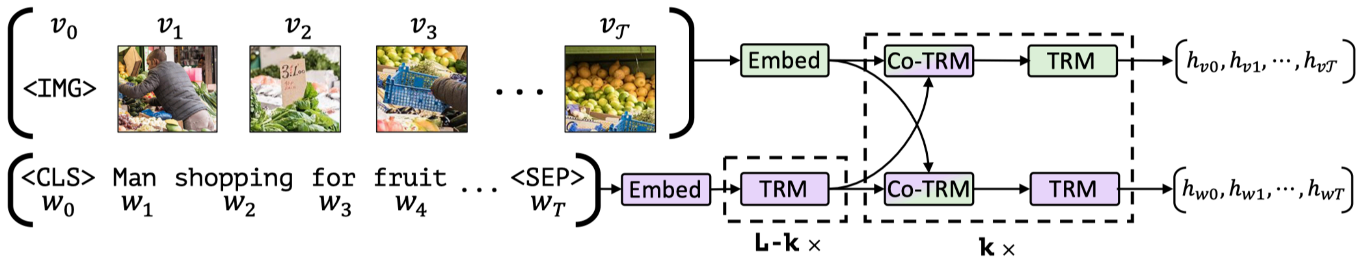
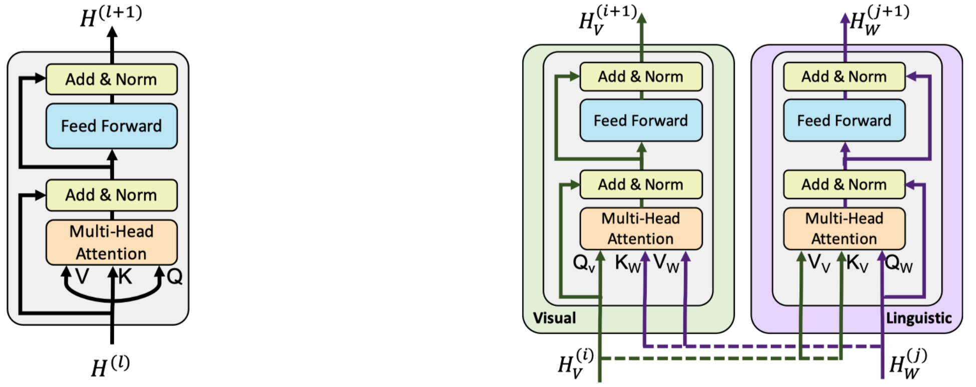


Figure 1: Our ViLBERT model consists of two parallel streams for visual (green) and linguistic (purple) processing that interact through novel co-attentional transformer layers. This structure allows for variable depths for each modality and enables sparse interaction through co-attention. Dashed boxes with multiplier subscripts denote repeated blocks of layers.



(a) Standard encoder transformer block

(b) Our co-attention transformer layer

Figure 2: We introduce a novel co-attention mechanism based on the transformer architecture. By exchanging key-value pairs in multi-headed attention, this structure enables vision-attended language features to be incorporated into visual representations (and vice versa).

```
def vilbert(weights, images, texts):
    while not done:
        images = BUTD_tokenize(images)
        masked_images, image_targets = mask_images(images, prob=0.15)
        masked_texts, text_targets = mask_texts(texts, prob=0.15)

        z_texts = model_forward(weights.text_encoder, texts)
        z_neg_texts = shuffle(z_texts, dim=batch_dim) # you can sample negs in other ways
        z_masked_texts = model_forward(weights.text_encoder, masked_texts)

        z = model_forward(weights.multimodal_encoder, images, z_texts)
        z_negs = model_forward(weights.multimodal_encoder, images, z_neg_texts)
        z_masked = model_forward(weights.multimodal_encoder, masked_images, z_masked_texts)

        losses = compute_mrm(weights, z_masked, image_targets) \
            + compute_mlm(weights, z_masked, text_targets) \
            + compute_itm(weights, z, z_negs)
        weights = optimize(weights, loss)
    return weights
```

```
def compute_mrm(weights, z_masked, image_targets):
    z_masked_images, z_masked_texts = split_z(z_masked)
    logits = model_forward(weights.mrm_head, z_masked_images)
    loss = loss_fn(logits, image_targets)
    return loss
```

```
def compute_mlm(weights, z_masked, text_targets):
    z_masked_images, z_masked_texts = split_z(z_masked)
    logits = model_forward(weights.mlm_head, z_masked_texts)
    loss = cross_entropy(logits, text_targets)
    return loss
```

```
def compute_itm(weights, z, z_negs):
    logits = model_forward(weights.itm_head, z)
    neg_logits = model_forward(weights.itm_head, z_negs)
    targets, neg_targets = ones(batch_length), zeros(batch_length)
    loss = cross_entropy(logits, targets) + cross_entropy(neg_logits, neg_targets)
    return loss
```

1. Lu, Jiasen et al. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks." NeurIPS (2019).

NAVER AI LAB

UNITER[1]

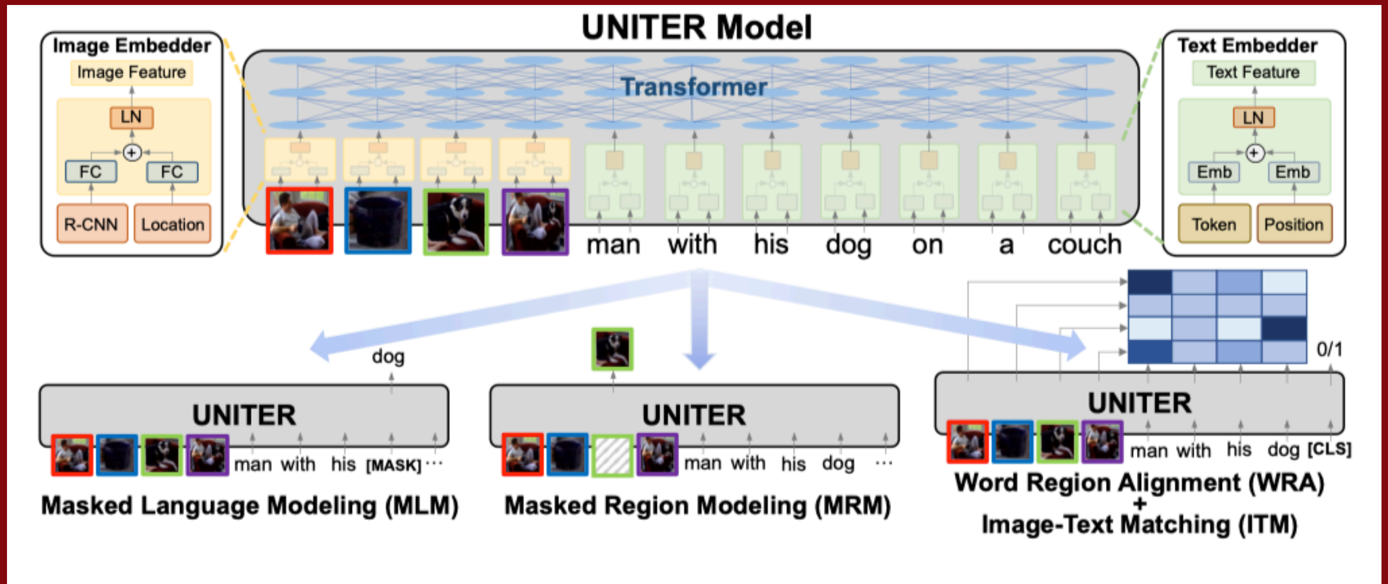


Fig. 1: Overview of the proposed UNITER model (best viewed in color), consisting of an Image Embedder, a Text Embedder and a multi-layer Transformer, learned through four pre-training tasks

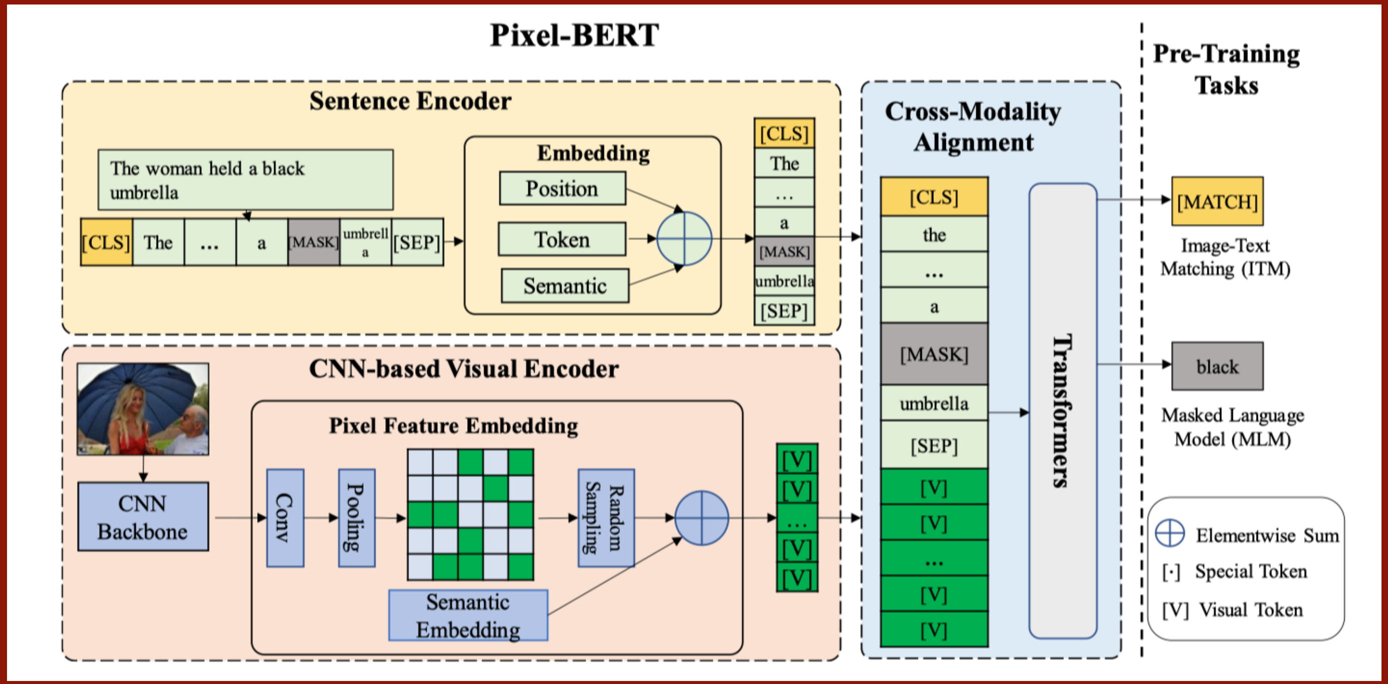
```
def uniter(weights, images, texts):
    while not done:
        images = BUTD_tokenize(images)
        masked_images, image_targets = mask_images(images, prob=0.15)
        masked_texts, text_targets = mask_texts(texts, prob=0.15)
        neg_texts = shuffle(texts, dim=batch_dim) # you can sample negs in other ways

        z = model_forward(weights.multimodal_encoder, images, texts)
        z_negs = model_forward(weights.multimodal_encoder, images, neg_texts)
        z_masked = model_forward(weights.multimodal_encoder, masked_images, masked_texts)

        losses = compute_mrm(weights, z_masked, image_targets) \
            + compute_mlm(weights, z_masked, text_targets) \
            + compute_itm(weights, z, z_negs) \
            + compute_wra(weights, z, z_negs)
        weights = optimize(weights, loss)
    return weights
```

1. Chen, Yen-Chun et al. "UNITER: UNiversal Image-TExt Representation Learning." ECCV (2020).

Pixel-BERT[1]



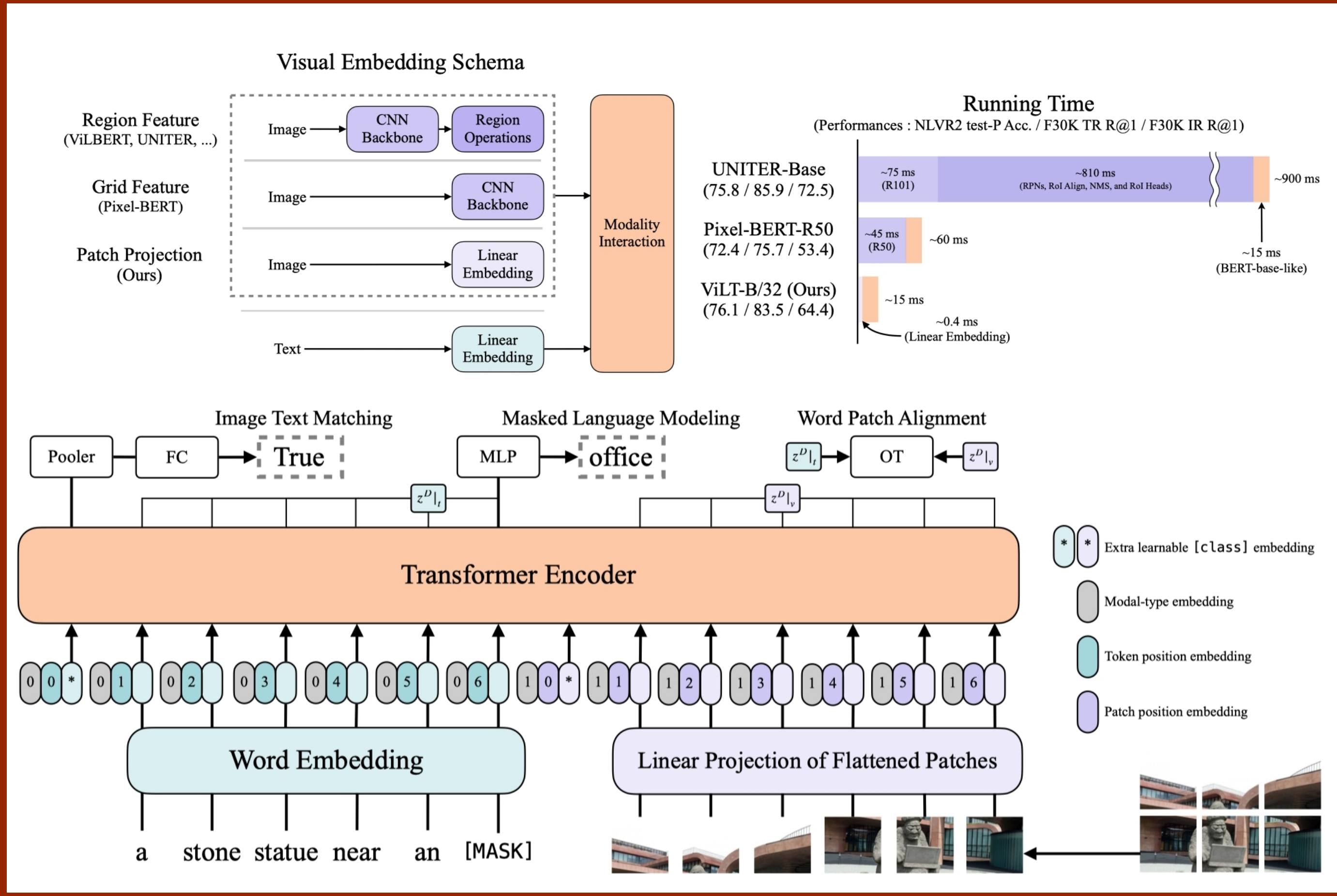

```
def pixelbert(weights, images, texts):
    while not done:
        images = model_forward(weights.cnn, images)
        masked_texts, text_targets = mask_texts(texts,
prob=0.15)
        neg_texts = shuffle(texts, dim=batch_dim) # you can
sample negs in other ways

        z = model_forward(weights.multimodal_encoder, images,
texts)
        z_negs = model_forward(weights.multimodal_encoder,
images, neg_texts)
        z_masked = model_forward(weights.multimodal_encoder,
images, masked_texts)

        losses = compute_mlm(weights, z_masked, text_targets)
\
            + compute_itm(weights, z, z_negs)
        weights = optimize(weights, loss)
    return weights
```

1. Huang, Zhicheng et al. “Pixel-BERT: Aligning Image Pixels with Text by Deep Multi-Modal Transformers.” ArXiv abs/2004.00849 (2020)

ViLT[1]




```
def vilt(weights, images, texts):
    while not done:
        masked_texts, text_targets = mask_texts(texts, prob=0.15)
        neg_texts = shuffle(texts, dim=batch_dim) # you can sample negs in other ways

        z = model_forward(weights.multimodal_encoder, images, texts)
        z_negs = model_forward(weights.multimodal_encoder, images, neg_texts)
        z_masked = model_forward(weights.multimodal_encoder, images, masked_texts)

        losses = compute_mlm(weights, z_masked, text_targets) \
            + compute_itm(weights, z, z_negs)\
            + compute_wpa(weights, z, z_negs)
        weights = optimize(weights, loss)
    return weights
```

1. Kim, Wonjae et al. “ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision.” ICML (2021).

Notable VLP models after ViLT

- ALBEF[1]
- UFO[2]
- BLIP[3]
- VLC[4]
- CoCa[5]

1. Li, Junnan et al. "Align before Fuse: Vision and Language Representation Learning with Momentum Distillation." Neurips (2021).
2. Wang, Jianfeng, et al. "UFO: A unified transformer for vision-language representation learning." arXiv preprint arXiv:2111.10023 (2021).
3. Li, Junnan, et al. "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation." arXiv preprint arXiv:2201.12086 (2022).
4. Gui, Liangke, et al. "Training Vision-Language Transformers from Captions Alone." arXiv preprint arXiv:2205.09256 (2022).
5. Yu, Jiahui, et al. "Coca: Contrastive captioners are image-text foundation models." arXiv preprint arXiv:2205.01917 (2022).

NAVER AI LAB

Considerations for VLP Models

1. What tasks to use?
2. What datasets to use?

1. What tasks to use?

- Classification (mostly evaluated by accuracy)
  - Visual Question Answering (DAQUAR, VQA, VQAv2, COCO-QA, FM-IQA, VG-QA, Visual7W, GQA, TextVQA, DocVQA, ...)
  - Visual Reasoning (SHAPES, CLEVR, NLVR, VCR)
  - Visual Entailment (SNLI-VE)
- Retrieval (mostly evaluated by recall)
  - Cross-Modal Retrieval (COCO, Flickr30K, Recipe1M+, ...)
  - Visual Grounding (RefCOCO, CLEVR-Ref+, Flickr30K Entities, ...)
- Generation (mostly evaluated by BLUE and CIDEr)
  - Image Captioning (COCO, NoCaps, Multi30K, ...)
  - Image Dense Captioning (Visual Genome)

Summary of VLP models' Performance

Model	Params	Text Retrieval Flickr30K (1K) @1	@5	@10	Text Retrieval MSCOCO (5K) @1	@5	@10	Image Retrieval Flickr30K (1K) @1	@5	@10	Image Retrieval MSCOCO (5K) @1	@5	@10
ALBEF† [26]	163M	94.3	99.4	99.8	73.1	91.4	96.0	**82.8**	**96.7**	**98.4**	56.8	81.5	89.2
VinVL$_{\text{LARGE}}$ [59]	452M	-	-	-	75.4	92.9	96.2	-	-	-	**58.8**	83.5	90.3
UNITER$_{\text{LARGE}}$ [7]	371M	87.3	98.0	99.2	65.7	88.6	93.8	75.6	94.1	96.8	52.9	79.9	88.0
METER-Swin$_{\text{BASE}}$ [29]	288M	92.4	99.0	99.5	76.2	93.2	96.8	79.0	95.6	98.0	54.9	81.4	89.3
PixelBERT [18]	144M	87.0	98.9	99.5	63.6	87.5	93.6	71.5	92.1	95.8	50.1	77.6	86.2
ViLT [22]	86M	83.5	96.7	98.6	61.5	86.3	92.7	64.4	88.7	93.8	42.7	72.9	83.1
VLC-Base (ours – 5.6M)	86M	89.2	99.2	99.8	71.3	91.2	95.8	72.4	93.4	96.5	50.7	78.9	88.0
VLC-Large (ours – 5.6M)	307M	**94.4**	**99.6**	**99.9**	**76.7**	**94.5**	**97.3**	79.1	95.8	98.2	58.4	**84.0**	**91.1**

Model	Params	VQAv2 test-dev	VQAv2 test-std	NLVR^2 dev	NLVR^2 test
Supervised ImageNet Bounded Boxes					
ViLBERT [33]	274M	70.55	70.92	-	-
LXMERT [48]	240M	72.42	72.54	74.90	74.50
VisualBERT [27]	170M	70.80	71.00	67.4	67.0
UNITER$_{\text{LARGE}}$ [7]	371M	73.82	74.02	79.12	79.98
OSCAR$_{\text{LARGE}}$ [29]	371M	73.61	73.82	79.12	80.37
VinVL$_{\text{LARGE}}$ † [59] (5.6M)	452M	76.52	76.60	**82.67**	**83.98**
Supervised ImageNet Classes					
METER-Swin$_{\text{BASE}}$ ‡ [12]	288M	76.43	76.42	82.23	82.47
ALBEF [26]	163M	74.54	74.70	80.24	80.50
Visual Parsing [55]	180M	74.00	74.17	77.61	78.05
PixelBERT [18]	144M	74.45	74.55	76.5	77.2
ViLT [22]	86M	71.26	-	75.70	76.13
No supervised classes or bounding boxes					
VLC-Base (ours – 4M)	86M	72.98	73.03	77.04	78.51
VLC-Base (ours – 5.6M)	86M	74.02	74.0	77.70	79.04
VLC-Large (ours – 5.6M)	307M	**76.95**	**77.02**	82.27	83.52
Pre-trained or initialized with > 10M data					
METER-CLIP-ViT$_{\text{BASE}}$ [12] (4M)	280M	77.68	77.64	82.33	83.05
X-VLM [58] (16M)	216M	78.22	78.37	84.41	84.76
BLIP [25] (129M)	252M	78.25	78.32	82.15	82.24
OFA [50] (54M)	930M	82.0	82.0	-	-
CoCa [57] (4.8B)	2.1B	82.3	82.3	86.1	87.0

2. What datasets to use?

2019 ~ 2021 (before ViLT)

Name	#Image-Text Pairs
COCO	567K
SBU Captions	1M
Conceptual Captions	3M
Visual Genome	5.4M

2021 ~ (after ViLT)

Name	#Image-Text Pairs
Localized Narratives	2M
English Wikipedia Image Text	6M
Conceptual Captions 12M	12M
Reddit Captions	12M
YFCC 100M CLIP filtered	30M
LAION 400M	400M
COYO 700M	700M
LAION 5B	5B

Q&A

As a lot of details are omitted for the conciseness of the lecture, any questions are welcomed.