COMP 648: Computer Vision Seminar

Contrastive Pre-training: SimCLR, CLIP, ALBEF

Ziyan Yang

RICE UNIVERSITY

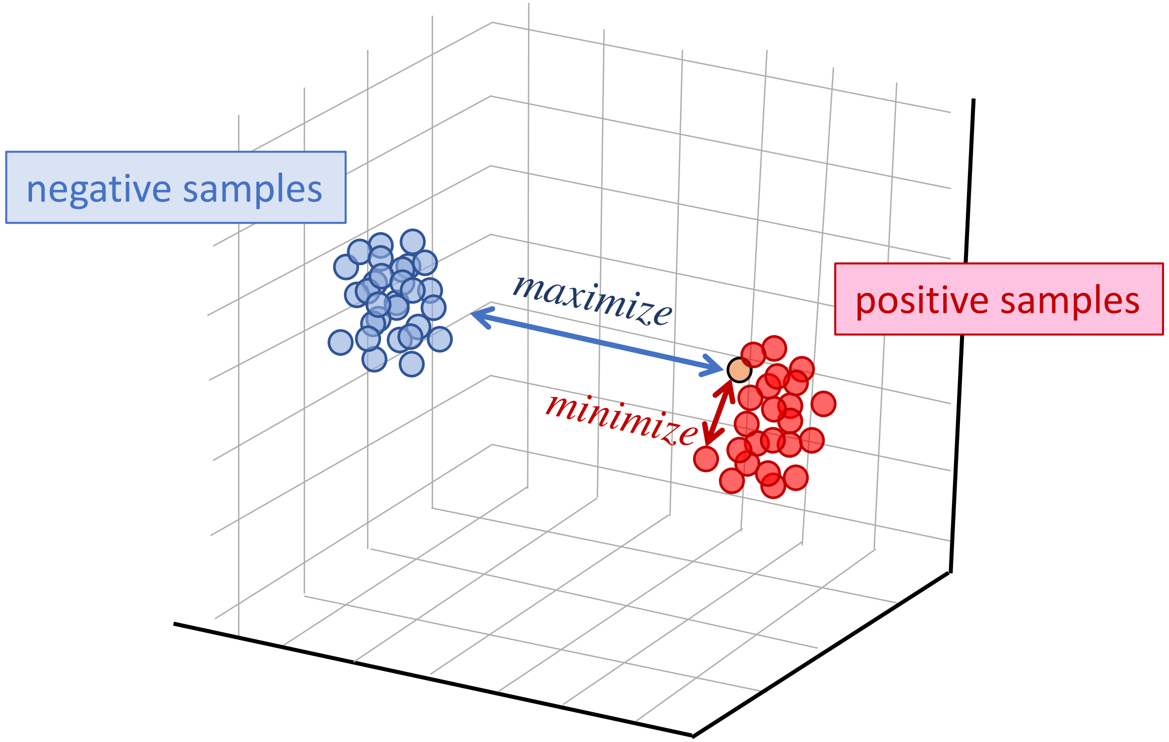
Pre-training

- Computer Vision:
  - ImageNet
    - Limited number of classes
    - Expensive to get labels
    - …

- Natural Language Processing:
  - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (https://arxiv.org/abs/1810.04805)
  - BERT is pretrained on BooksCorpus + English Wikipedia unlabeled text data
  - Outperform other models on 11 NLP downstream tasks

What is contrastive learning?

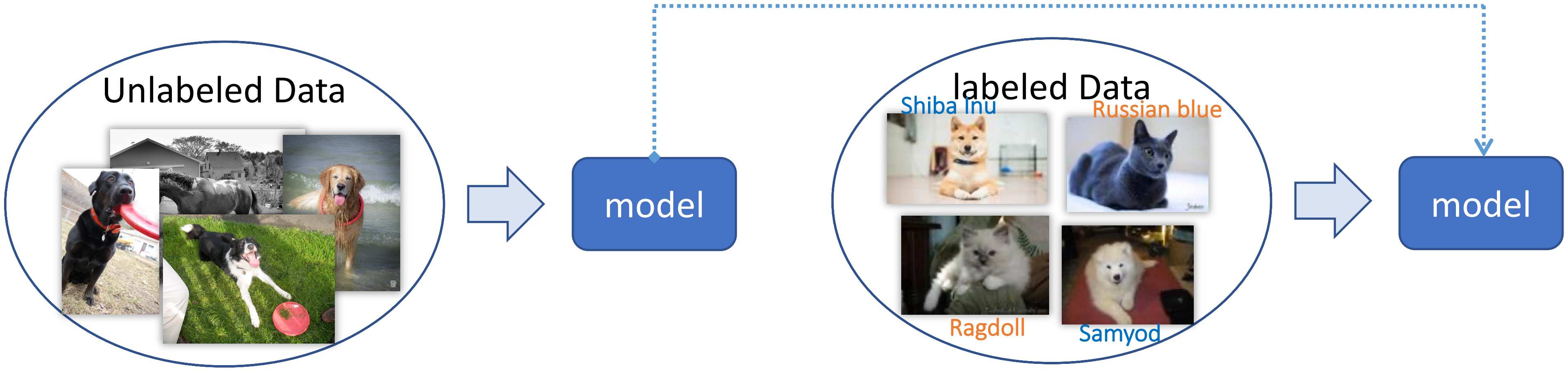
- Learn from both positive and negative samples:
    - For each sample $\boldsymbol{x}$
        - decrease its distance between positive samples $\boldsymbol{x}^{+}$
        - increase its distance between negative samples $\boldsymbol{x}^{-}$
        - Finally, $d\,(\,f\,(\boldsymbol{x}), f\,(\boldsymbol{x}^{+})\,) << d\,(\,f\,(\boldsymbol{x}), f\,(\boldsymbol{x}^{-})\,)$
- It is not necessary to know exact labels for samples (e.g., the label of an image), we only need to know if samples are positive or negative



SimCLR

A Simple Framework for Contrastive Learning of Visual Representations
Paper address: https://arxiv.org/pdf/2002.05709.pdf

- Task: Image Classification



- Goal: use contrastive learning to learn better image representations for classification tasks

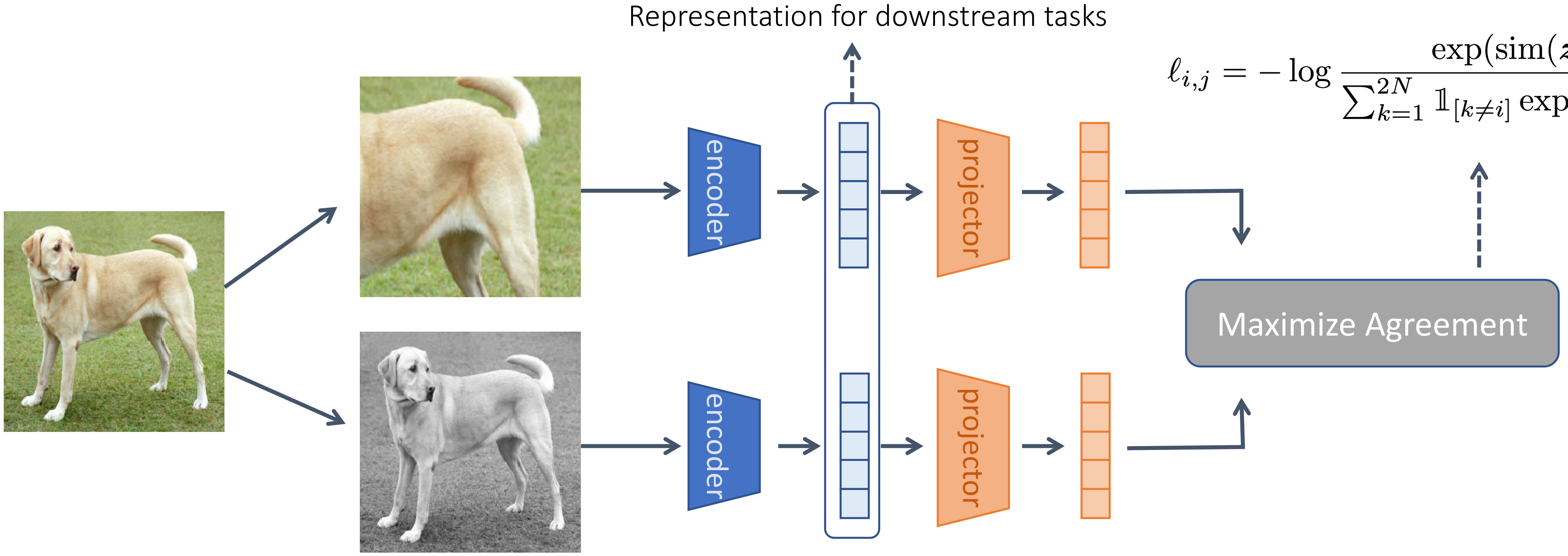
SimCLR

A Simple Framework for Contrastive Learning of Visual Representations
Paper address: https://arxiv.org/pdf/2002.05709.pdf

- Model Overview

$$\mathrm{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\|$$

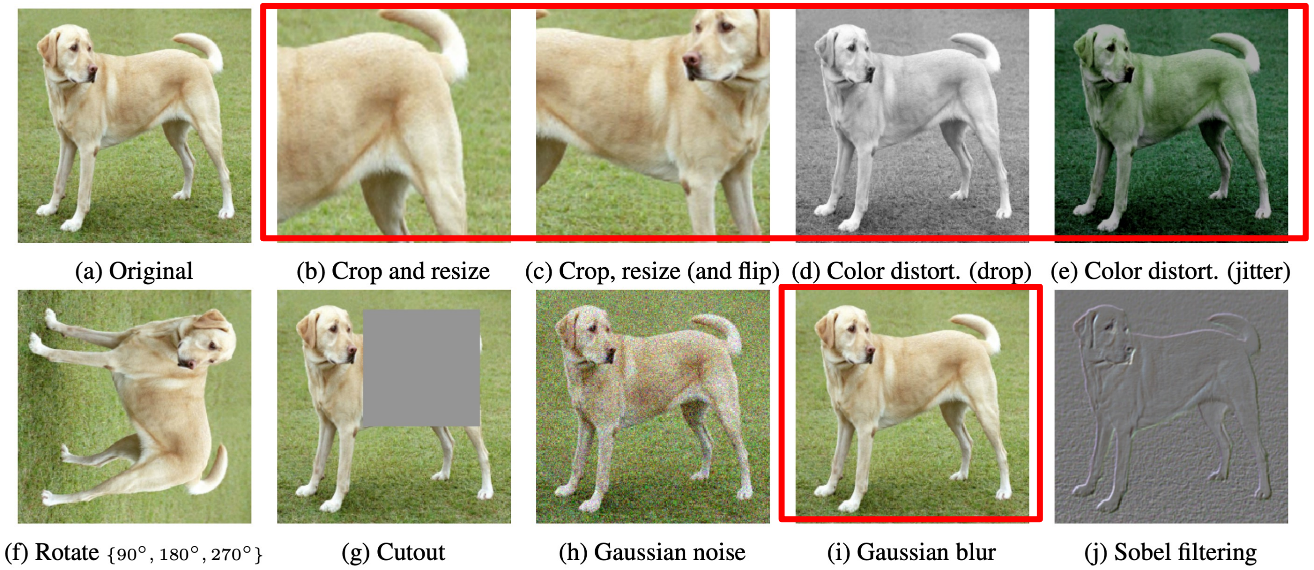
$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$



SimCLR

A Simple Framework for Contrastive Learning of Visual Representations
Paper address: https://arxiv.org/pdf/2002.05709.pdf

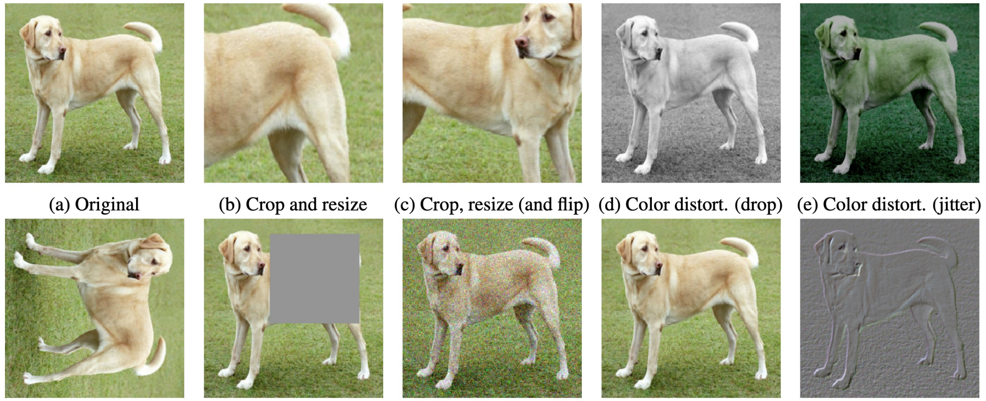
- Construct positive samples by data augmentation operators:



SimCLR

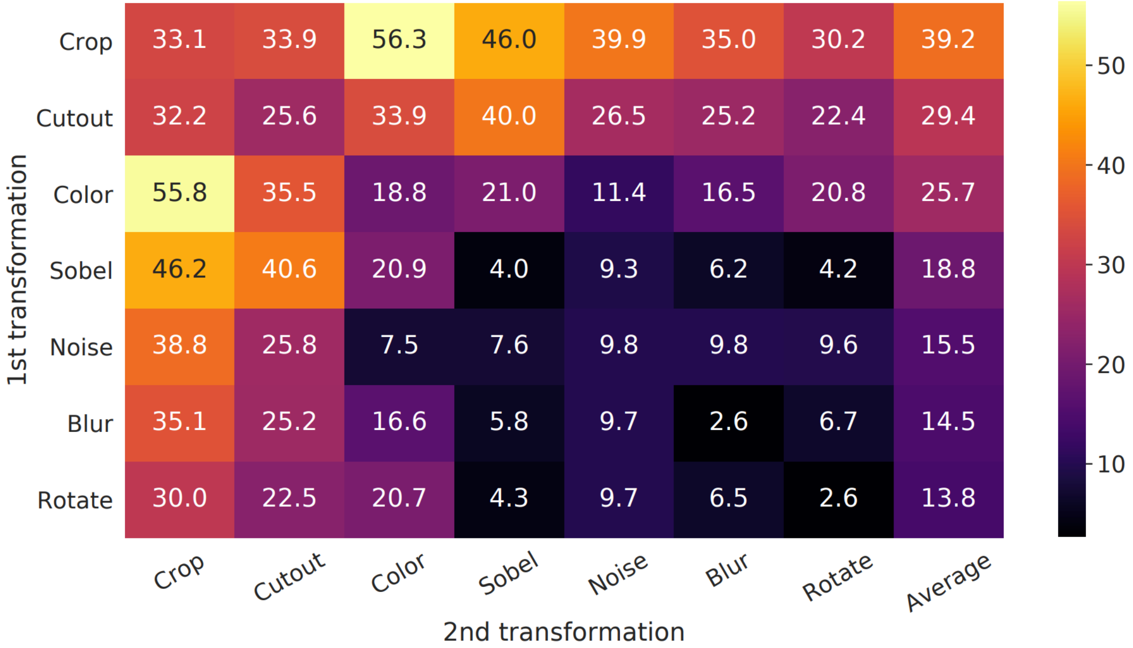
A Simple Framework for Contrastive Learning of Visual Representations
Paper address: https://arxiv.org/pdf/2002.05709.pdf

- Findings:
    - no single transformation suffices to learn good representations



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate {90°, 180°, 270°} (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

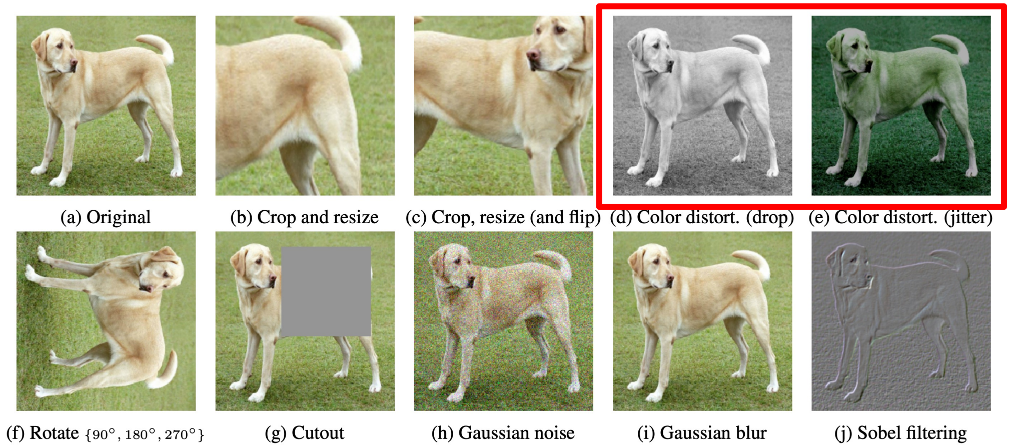


Linear evaluation, top-1 accuracy, ImageNet

SimCLR

A Simple Framework for Contrastive Learning of Visual Representations
Paper address: https://arxiv.org/pdf/2002.05709.pdf

- Findings:
  - Contrastive learning needs stronger data augmentation than supervised learning



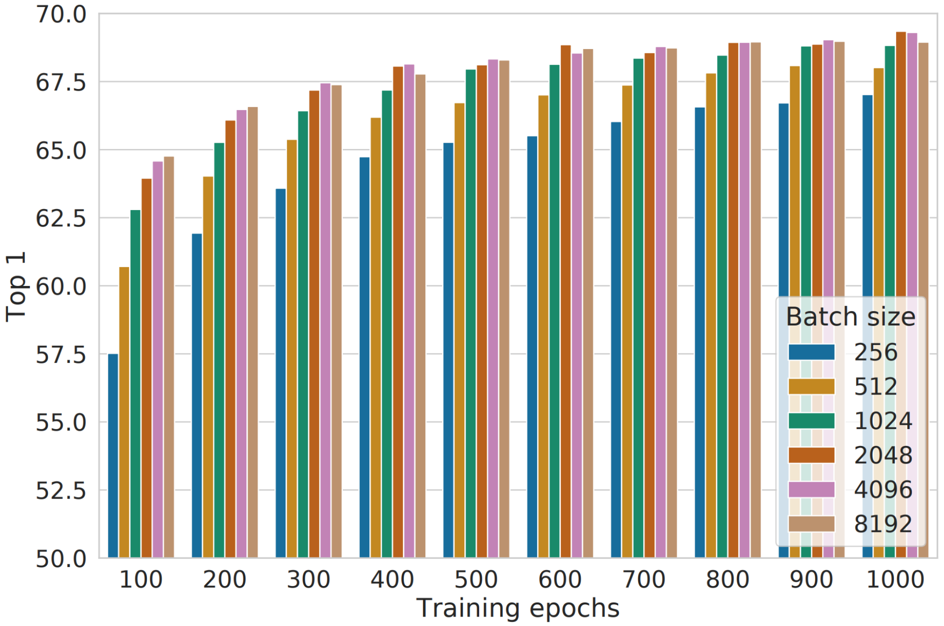
Methods	Color distortion strength 1/8	1/4	1/2	1	1 (+Blur)	AutoAug
SimCLR	59.6	61.0	62.6	63.2	64.5	61.1
Supervised	77.0	76.7	76.5	75.7	75.4	77.1

Table1: Top-1 accuracy of unsupervised ResNet-50 using linear evaluation and supervised ResNet-50 , under varied color distortion strength

SimCLR

A Simple Framework for Contrastive Learning of Visual Representations
Paper address: https://arxiv.org/pdf/2002.05709.pdf

- Findings:
    - Unsupervised contrastive learning benefits (more) from **bigger** models
    - Contrastive learning benefits (more) from **larger batch sizes** and **longer training**

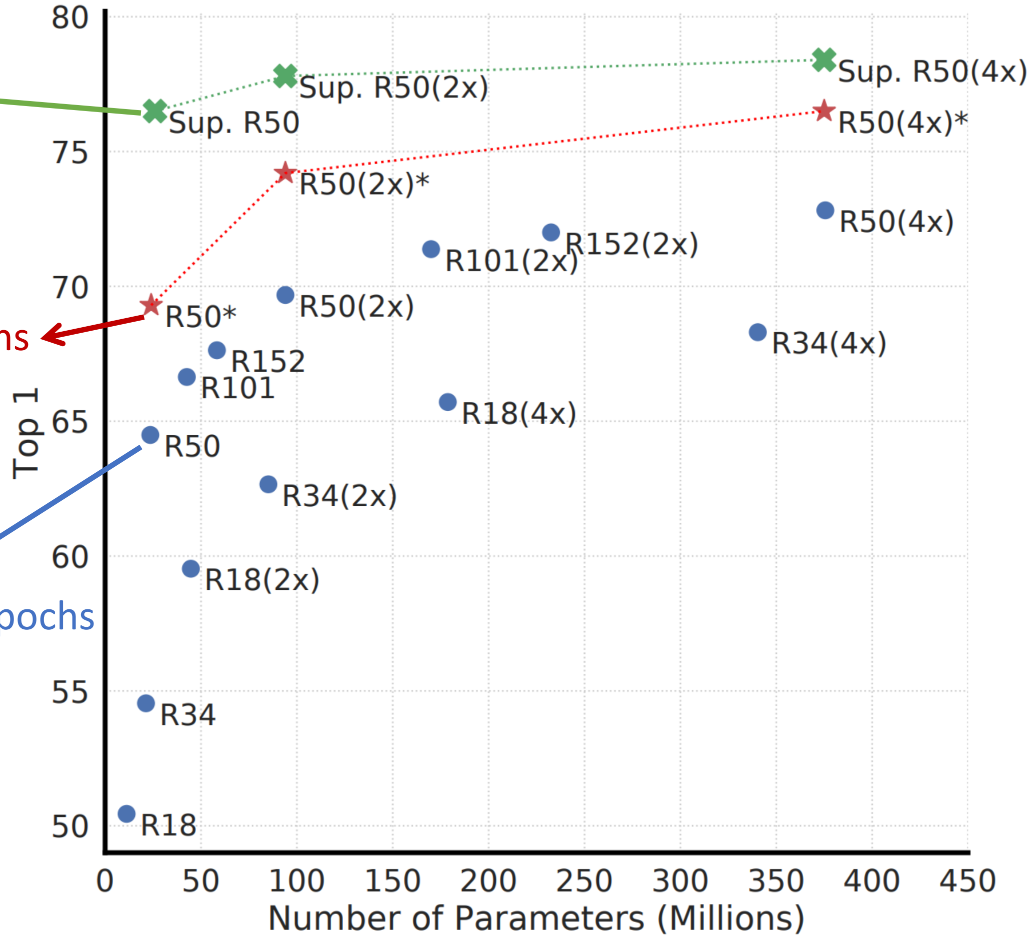


Top-1 accuracy
ImageNet

Supervised

SimCLR – 1000 epochs

SimCLR – 100 epochs



SimCLR

A Simple Framework for Contrastive Learning of Visual Representations
Paper address: https://arxiv.org/pdf/2002.05709.pdf

- Comparing with SOTA:

Method	Architecture	Param (M)	Top 1	Top 5
Methods using ResNet-50:				
Local Agg.	ResNet-50	24	60.2	-
MoCo	ResNet-50	24	60.6	-
PIRL	ResNet-50	24	63.6	-
CPC v2	ResNet-50	24	63.8	85.3
SimCLR (ours)	ResNet-50	24	**69.3**	**89.0**
Methods using other architectures:				
Rotation	RevNet-50 (4×)	86	55.4	-
BigBiGAN	RevNet-50 (4×)	86	61.3	81.9
AMDIM	Custom-ResNet	626	68.1	-
CMC	ResNet-50 (2×)	188	68.4	88.2
MoCo	ResNet-50 (4×)	375	68.6	-
CPC v2	ResNet-161 (∗)	305	71.5	90.1
SimCLR (ours)	ResNet-50 (2×)	94	74.2	92.0
SimCLR (ours)	ResNet-50 (4×)	375	**76.5**	**93.2**

Self-supervised

Method	Architecture	Label fraction 1%	10%
		Top 5	
Supervised baseline	ResNet-50	48.4	80.4
Methods using other label-propagation:			
Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	-	88.5
FixMatch (w. RandAug)	ResNet-50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet-50 (4×)	-	91.2
Methods using representation learning only:			
InstDisc	ResNet-50	39.2	77.4
BigBiGAN	RevNet-50 (4×)	55.2	78.8
PIRL	ResNet-50	57.2	83.8
CPC v2	ResNet-161(∗)	77.9	91.2
SimCLR (ours)	ResNet-50	75.5	87.8
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2
SimCLR (ours)	ResNet-50 (4×)	**85.8**	**92.6**

Semi-supervised

SimCLR

A Simple Framework for Contrastive Learning of Visual Representations
Paper address: https://arxiv.org/pdf/2002.05709.pdf

- Comparing with SOTA:

	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
SimCLR (ours)	**76.9**	**95.3**	80.2	48.4	**65.9**	60.0	61.2	**84.2**	**78.9**	89.2	**93.9**	**95.0**
Supervised	75.2	**95.7**	**81.2**	**56.4**	64.9	**68.8**	**63.8**	83.8	**78.7**	**92.3**	**94.1**	94.2
Fine-tuned:												
SimCLR (ours)	**89.4**	**98.6**	**89.0**	**78.2**	**68.1**	**92.1**	**87.0**	**86.6**	**77.8**	92.1	**94.1**	97.6
Supervised	88.7	98.3	**88.7**	**77.8**	67.0	91.4	**88.0**	86.5	**78.8**	**93.2**	**94.2**	**98.0**
Random init	88.3	96.0	81.9	**77.0**	53.7	91.3	84.8	69.4	64.1	82.7	72.5	92.5

Transfer Learning

Some code

```python
def info_nce_loss(self, features):

    labels = torch.cat([torch.arange(self.args.batch_size) for i in range(self.args.n_views)], dim=0)
    labels = (labels.unsqueeze(0) == labels.unsqueeze(1)).float()
    labels = labels.to(self.args.device)

    features = F.normalize(features, dim=1)

    similarity_matrix = torch.matmul(features, features.T)
    # assert similarity_matrix.shape == (
    #     self.args.n_views * self.args.batch_size, self.args.n_views * self.args.batch_size)
    # assert similarity_matrix.shape == labels.shape

    # discard the main diagonal from both: labels and similarities matrix
    mask = torch.eye(labels.shape[0], dtype=torch.bool).to(self.args.device)
    labels = labels[~mask].view(labels.shape[0], -1)
    similarity_matrix = similarity_matrix[~mask].view(similarity_matrix.shape[0], -1)
    # assert similarity_matrix.shape == labels.shape

    # select and combine multiple positives
    positives = similarity_matrix[labels.bool()].view(labels.shape[0], -1)

    # select only the negatives the negatives
    negatives = similarity_matrix[~labels.bool()].view(similarity_matrix.shape[0], -1)

    logits = torch.cat([positives, negatives], dim=1)
    labels = torch.zeros(logits.shape[0], dtype=torch.long).to(self.args.device)

    logits = logits / self.args.temperature
    return logits, labels
```

features: [batch*2, batch*2]

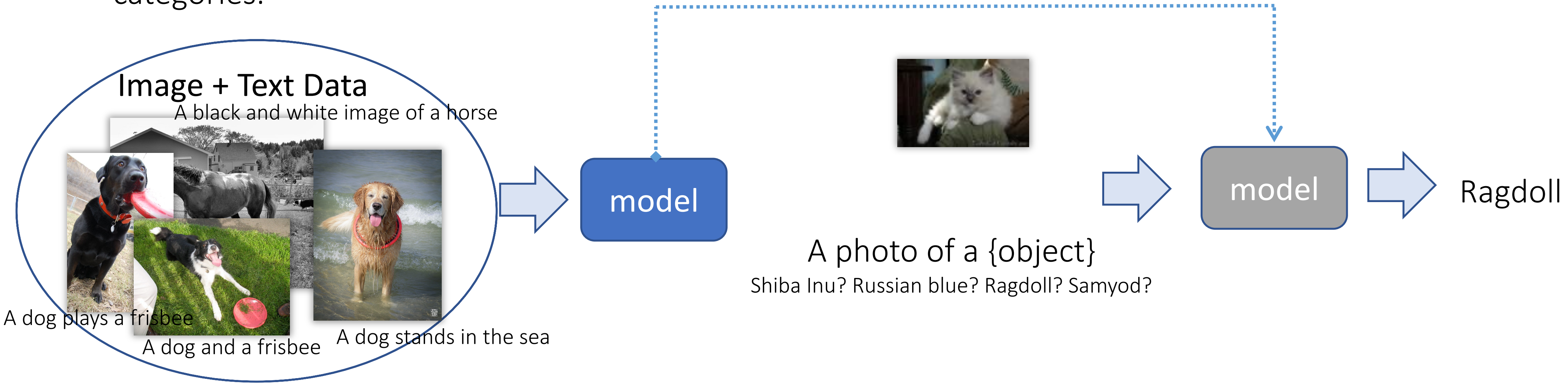
features: [batch*2, feature_dimension]

similarity_matrix: [batch*2, batch*2 - 1]

CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

- Task: Image Classification
  - However, previous works still need some labeled data, and can only predict results for pre-defined categories.

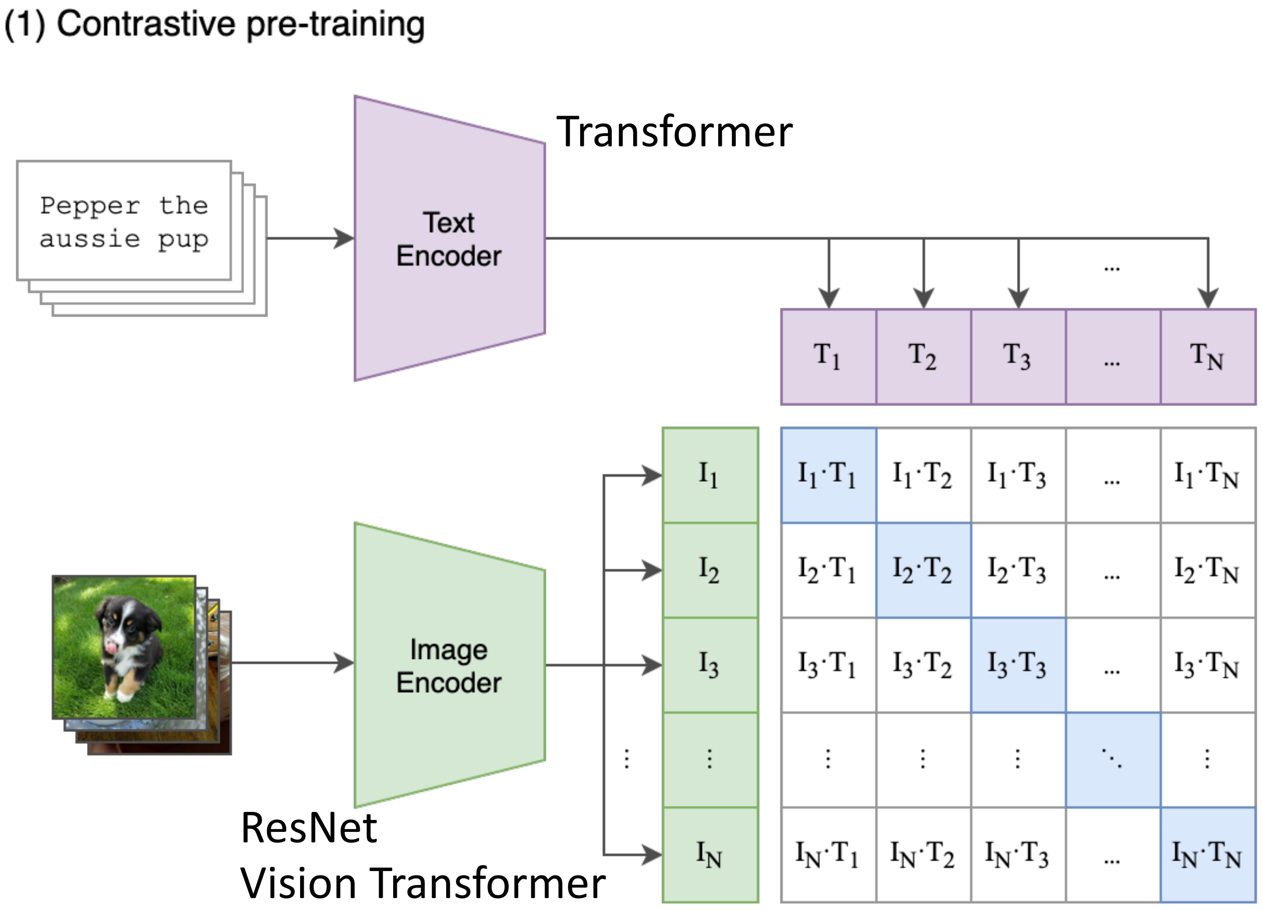


- Goal: learn directly from natural language using contrastive learning and enable zero-shot transfer of the model to downstream tasks

CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

- Model Overview
  - Training:
  - Data: WebImageText
    - 400 million (image, text) pairs

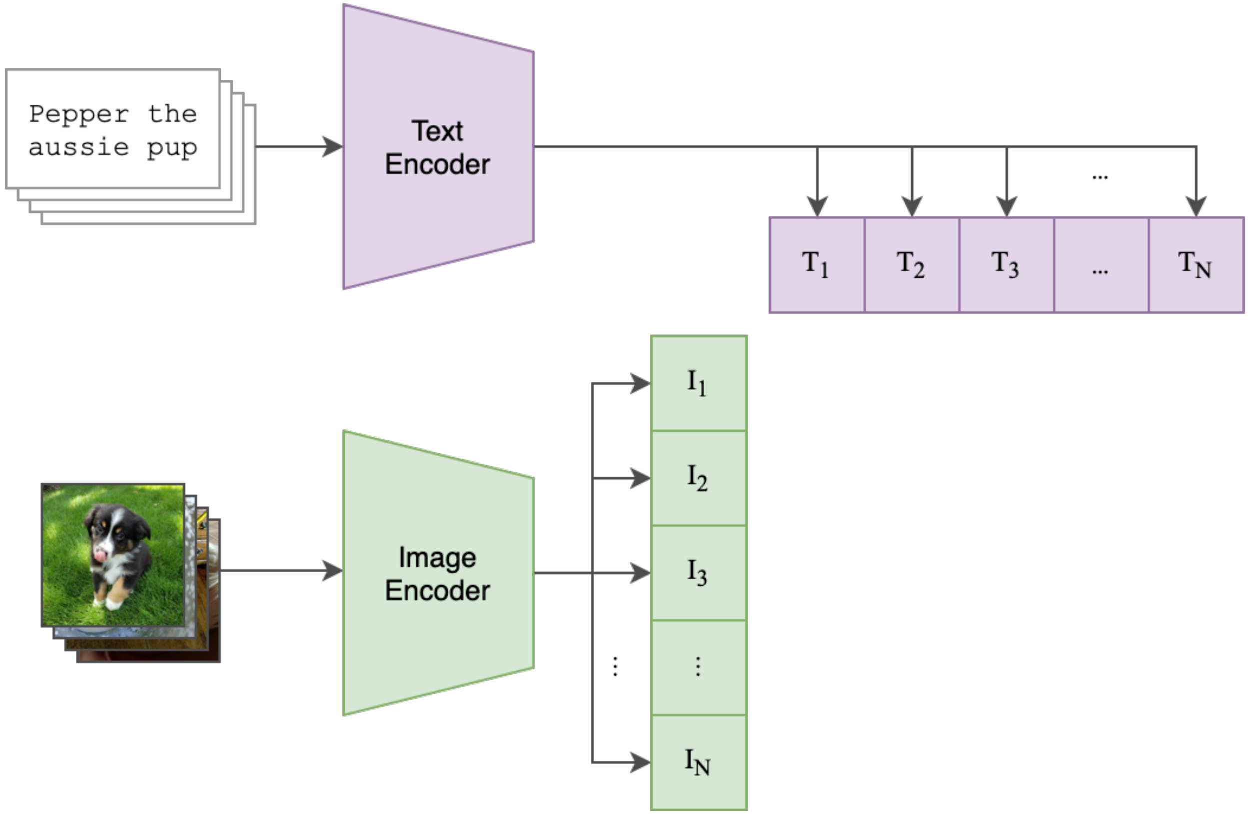


CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

- Algorithm:

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

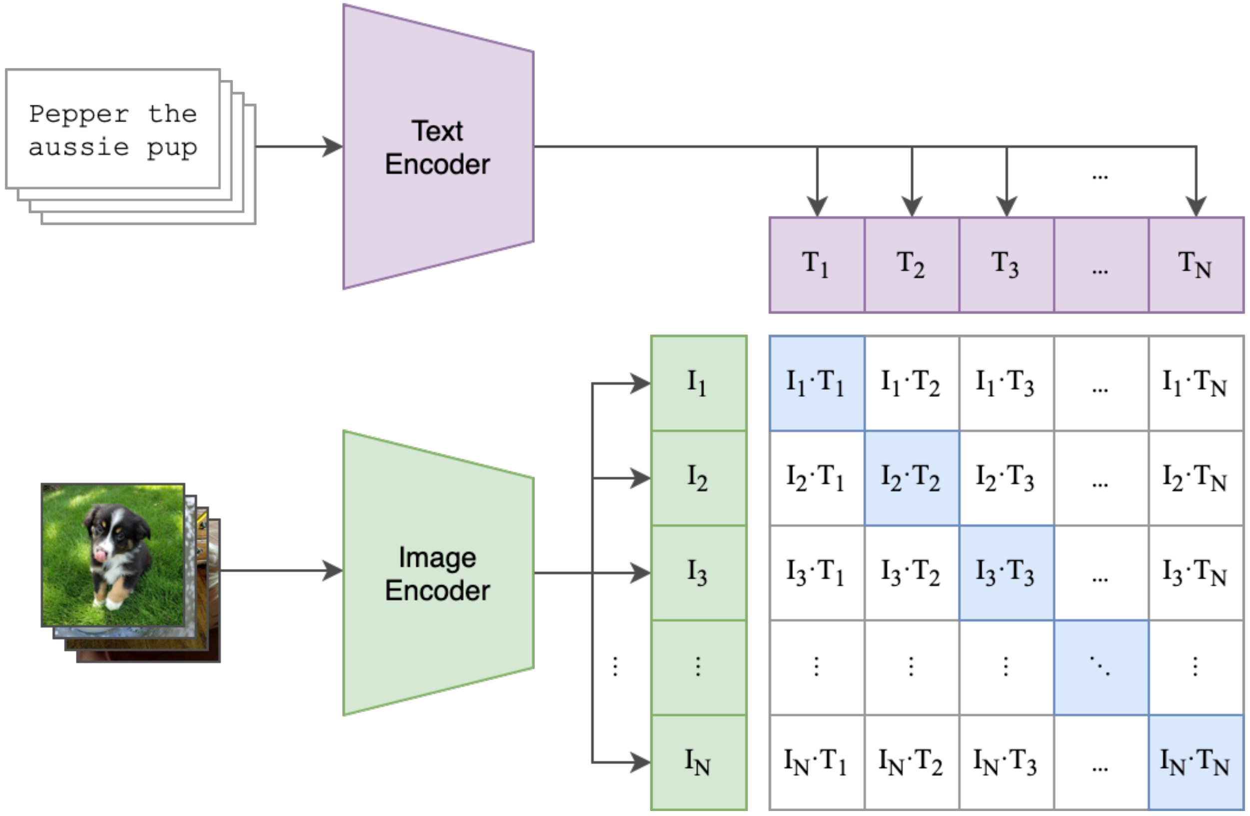


CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

- Algorithm:

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

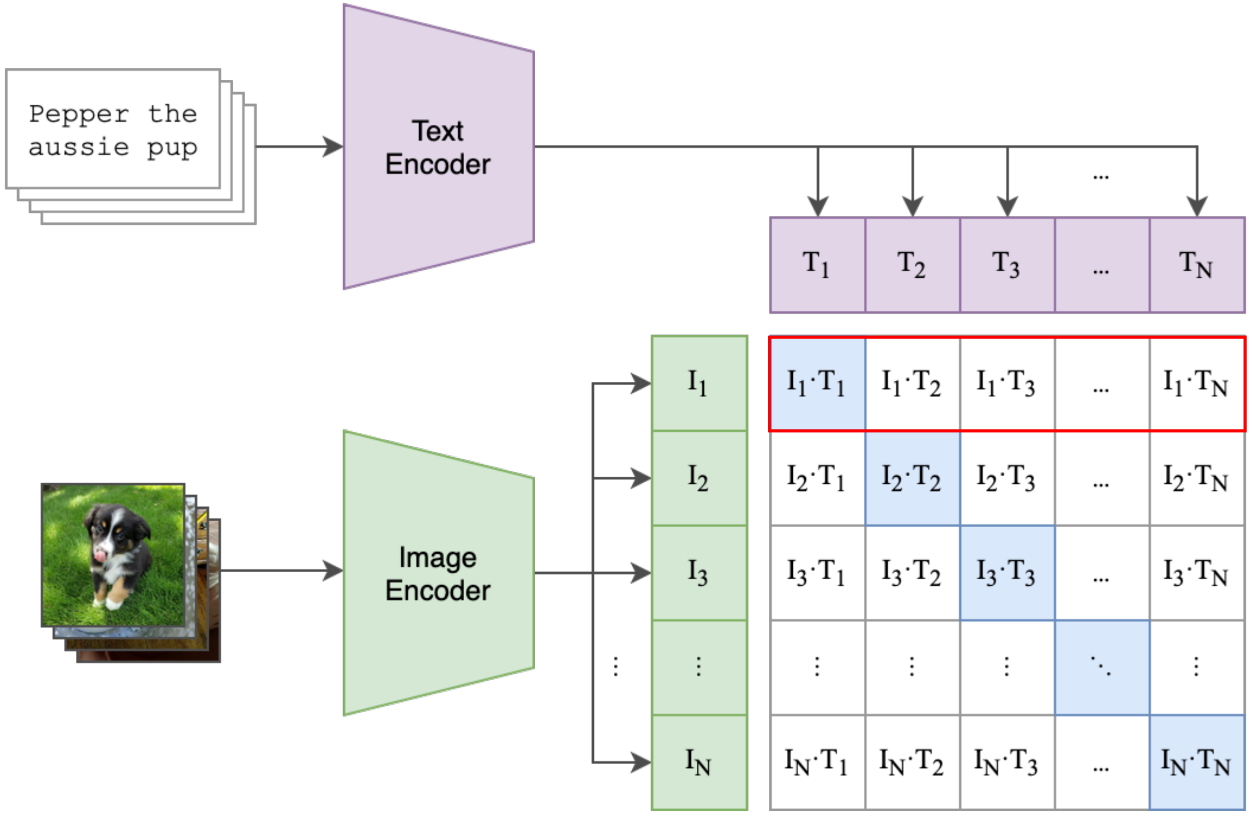


CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

- Algorithm:

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

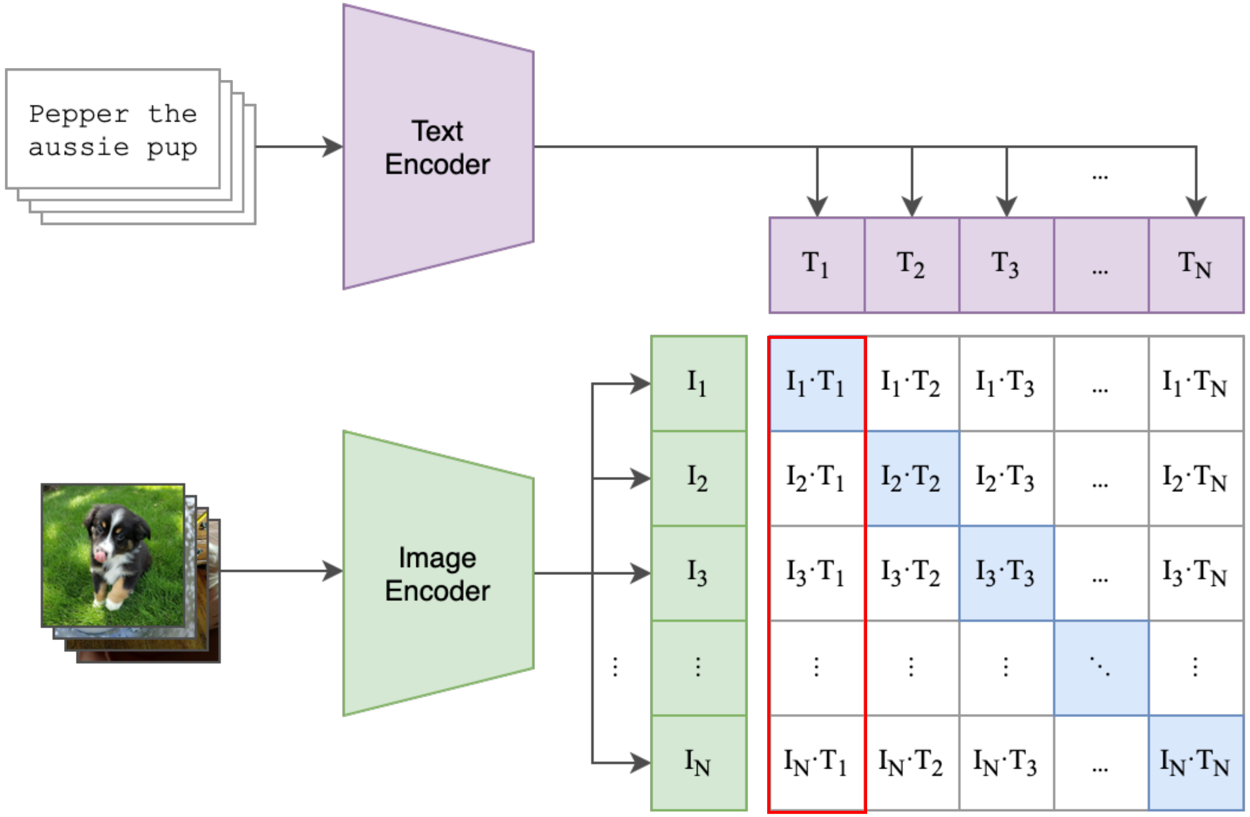


CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

- Algorithm:

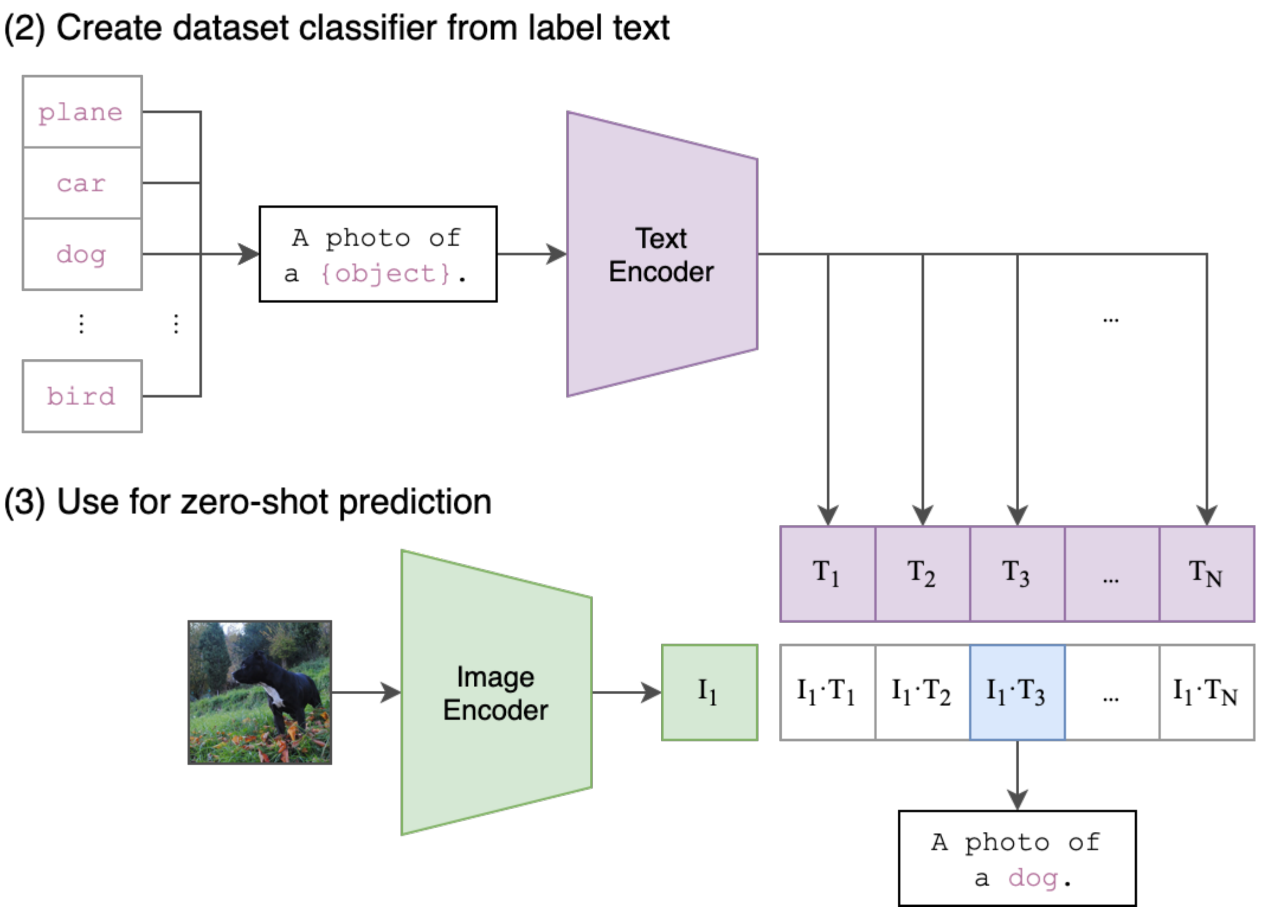
```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```



CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

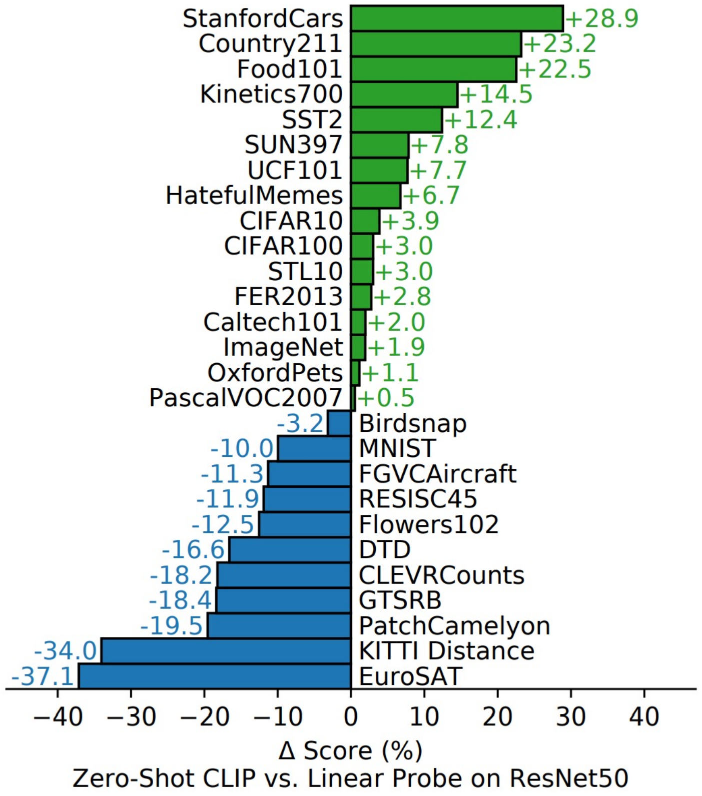
- Model Overview
    - Testing:

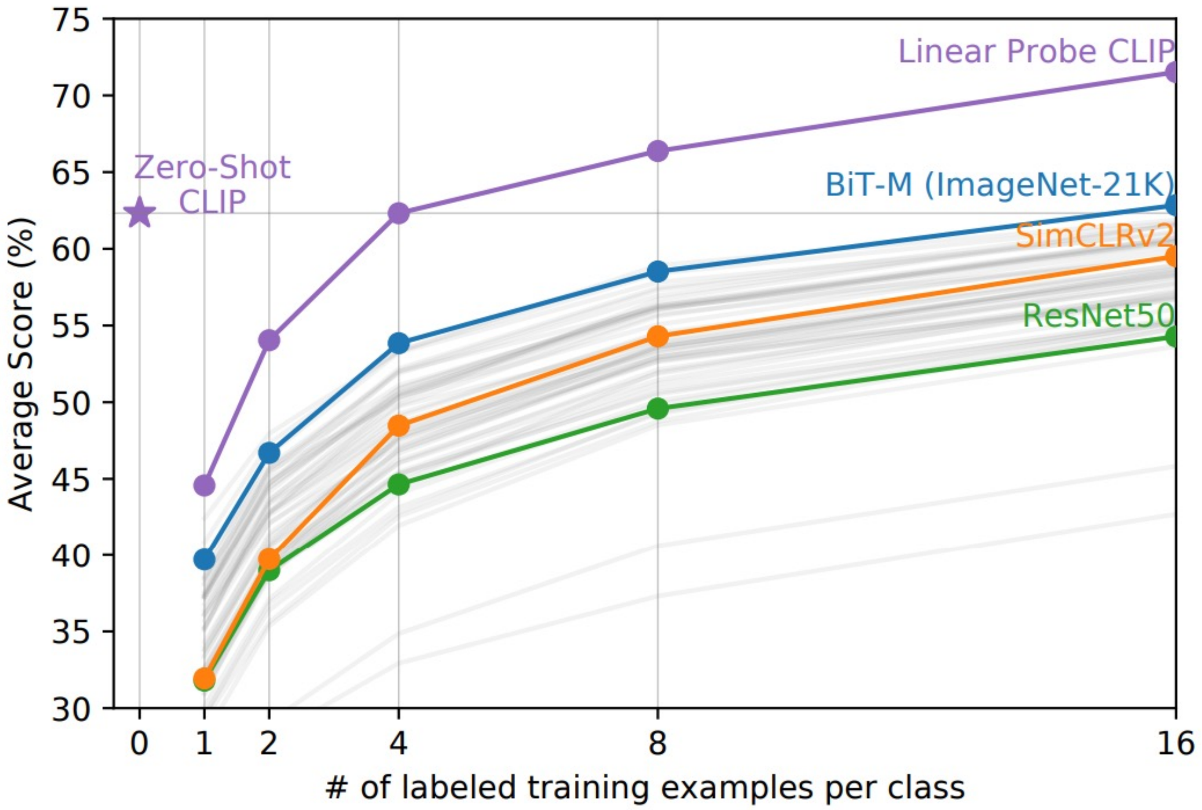


CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

- Results: across 27 datasets

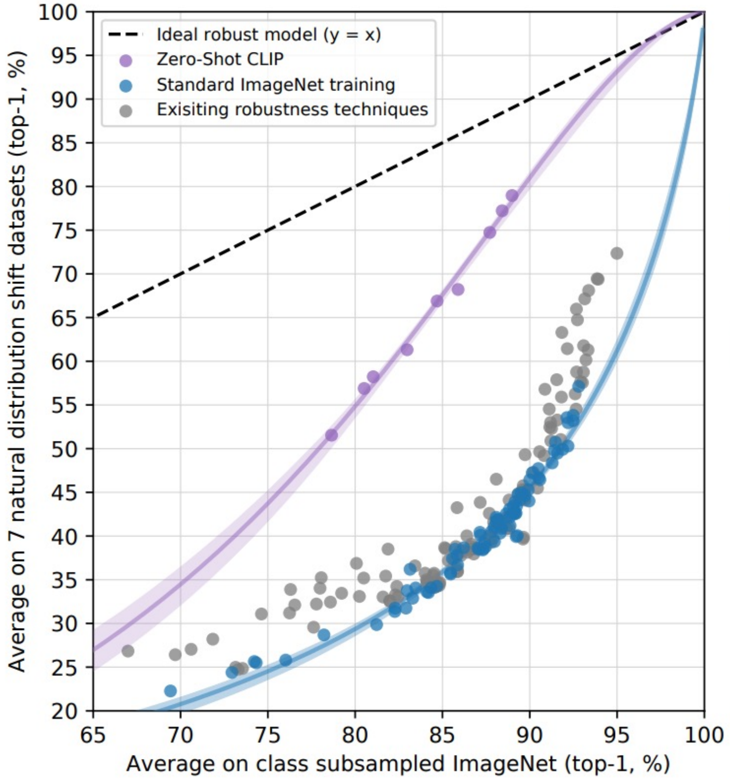


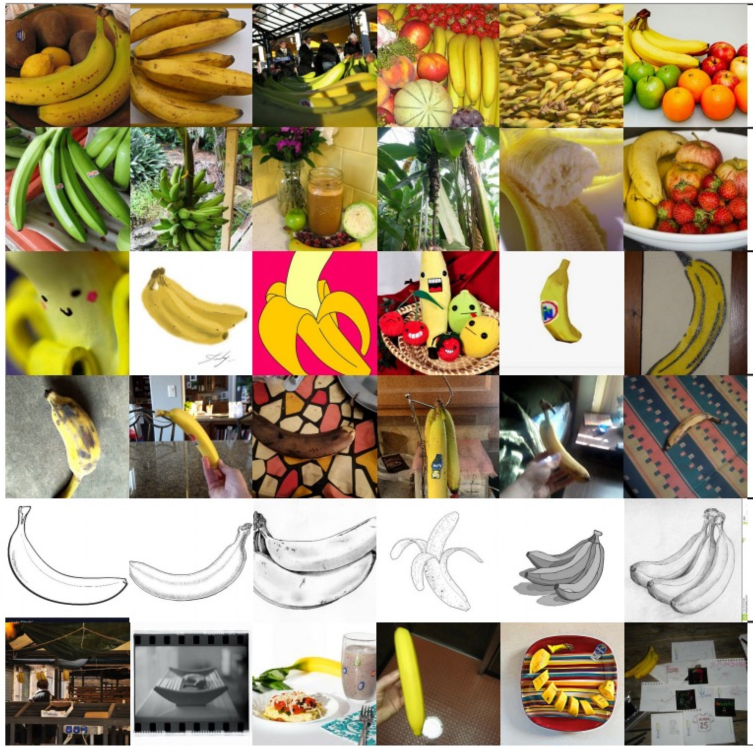


CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

- Results: Robustness to Natural Distribution Shift



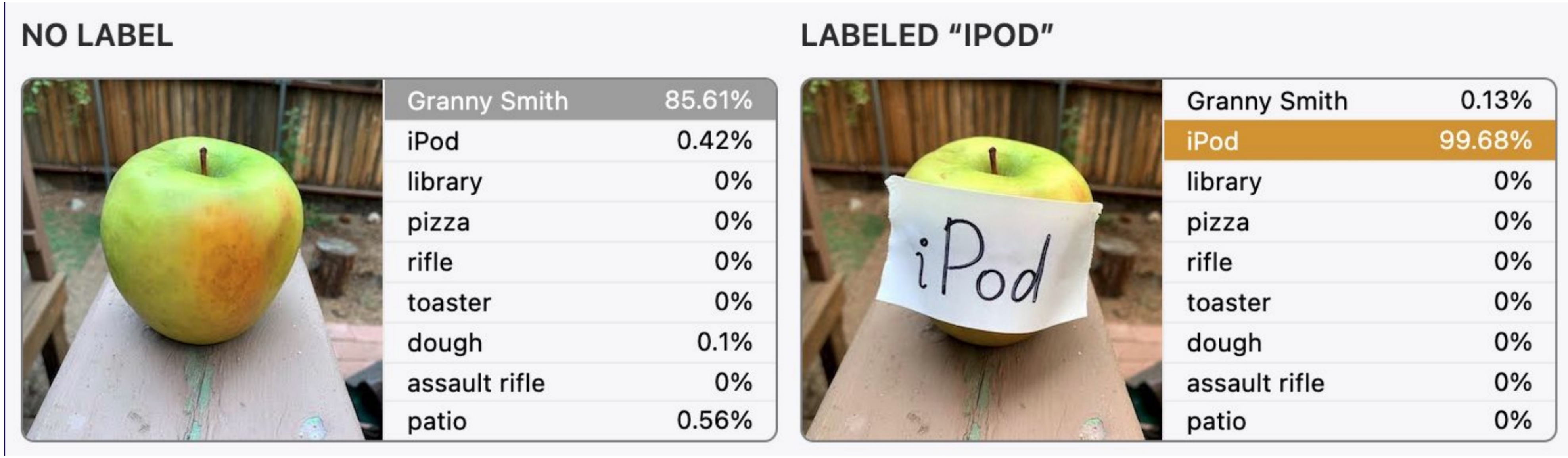


	ImageNet ResNet101	Zero-Shot CLIP	Δ Score
ImageNet	**76.2**	**76.2**	0%
ImageNetV2	64.3	**70.1**	+5.8%
ImageNet-R	37.7	**88.9**	+51.2%
ObjectNet	32.6	**72.3**	+39.7%
ImageNet Sketch	25.2	**60.2**	+35.0%
ImageNet-A	2.7	**77.1**	+74.4%

CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

- Limitations
  - Possible solution:
    - Editing a Classifier by Rewriting Its Prediction Rules
      https://proceedings.neurips.cc/paper/2021/file/c46489a2d5a9a9ecfc53b17610926ddd-Paper.pdf



CLIP

Learning Transferable Visual Models From Natural Language Supervision
Paper address: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf

- Limitations
    - Possible solution:
        - Learning to Prompt for Vision-Language Models https://arxiv.org/pdf/2109.01134.pdf

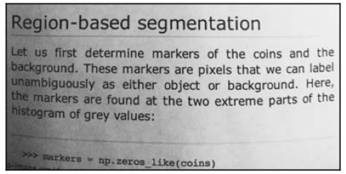
Caltech101

Prompt	Accuracy
a [CLASS].	82.68
a photo of [CLASS].	80.81
a photo of a [CLASS].	86.29

Some code

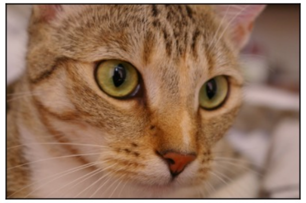
```
[ ] model, preprocess = clip.load("ViT-B/32")
    model.cuda().eval()
    input_resolution = model.visual.input_resolution
    context_length = model.context_length
    vocab_size = model.vocab_size

    print("Model parameters:", f"{np.sum([int(np.prod(p.shape)) for p in model.parameters()]):,}")
    print("Input resolution:", input_resolution)
    print("Context length:", context_length)
    print("Vocab size:", vocab_size)
```

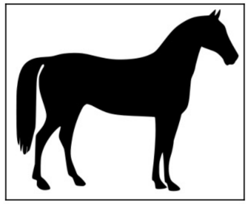
```
100%|████████████████████████████████████████| 338M/338M [00:05<00:00, 63.0MiB/s]
Model parameters: 151,277,313
Input resolution: 224
Context length: 77
Vocab size: 49408
```


Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

>>> markers = np.zeros_like(coins)





Some code

```
image_input = torch.tensor(np.stack(images)).cuda()
with torch.no_grad():
    image_features = model.encode_image(image_input).float()
```

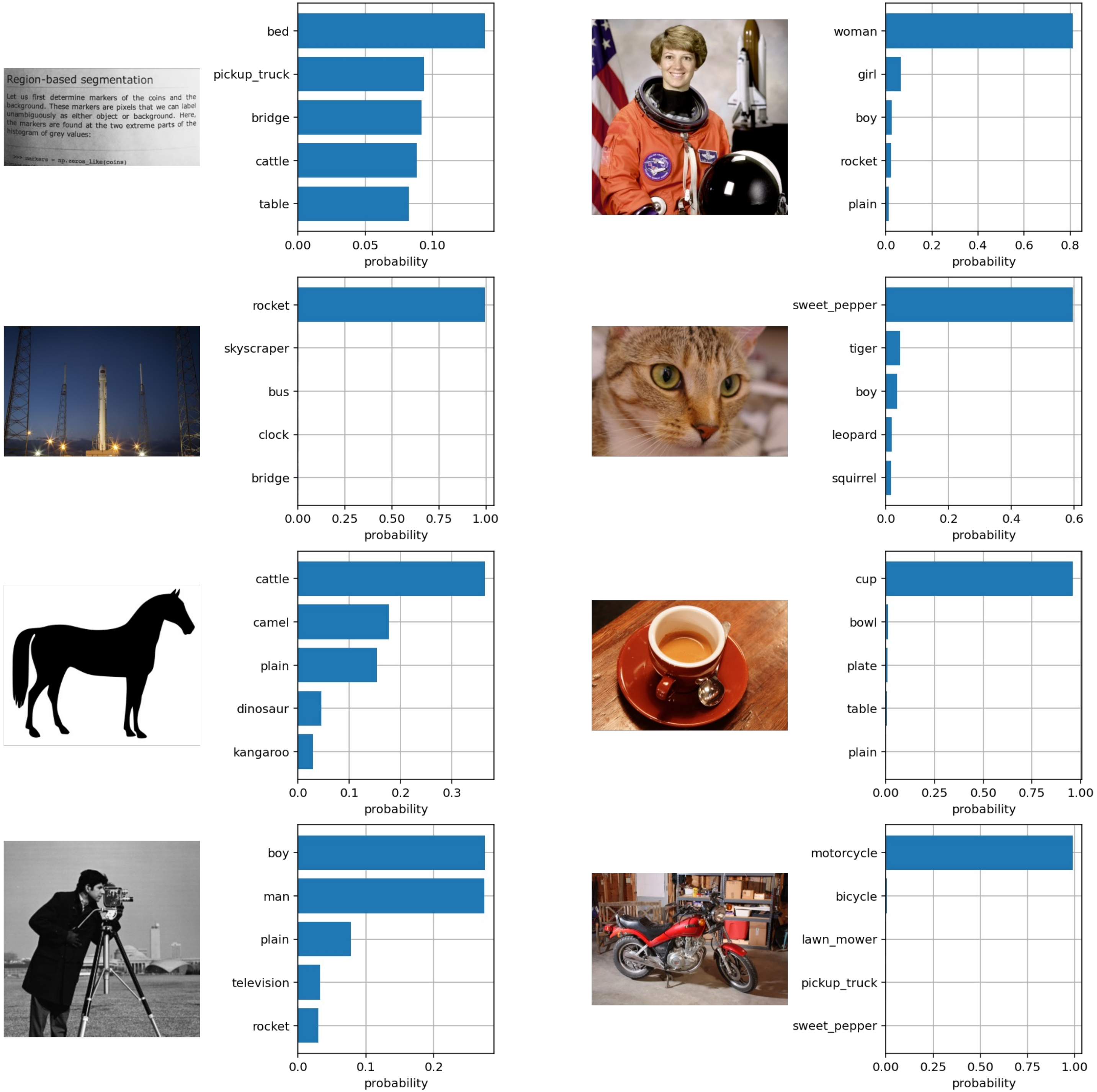
```
from torchvision.datasets import CIFAR100

cifar100 = CIFAR100(os.path.expanduser("~/.cache"), transform=preprocess, download=True)
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz to /root/.cache/cifar-100-python.tar.gz
169001984/? [00:06<00:00, 25734958.25it/s]

Extracting /root/.cache/cifar-100-python.tar.gz to /root/.cache
```

```
text_descriptions = [f"This is a photo of a {label}" for label in cifar100.classes]
text_tokens = clip.tokenize(text_descriptions).cuda()
```

```
with torch.no_grad():
    text_features = model.encode_text(text_tokens).float()
    text_features /= text_features.norm(dim=-1, keepdim=True)

text_probs = (100.0 * image_features @ text_features.T).softmax(dim=-1)
top_probs, top_labels = text_probs.cpu().topk(5, dim=-1)
```

Some code

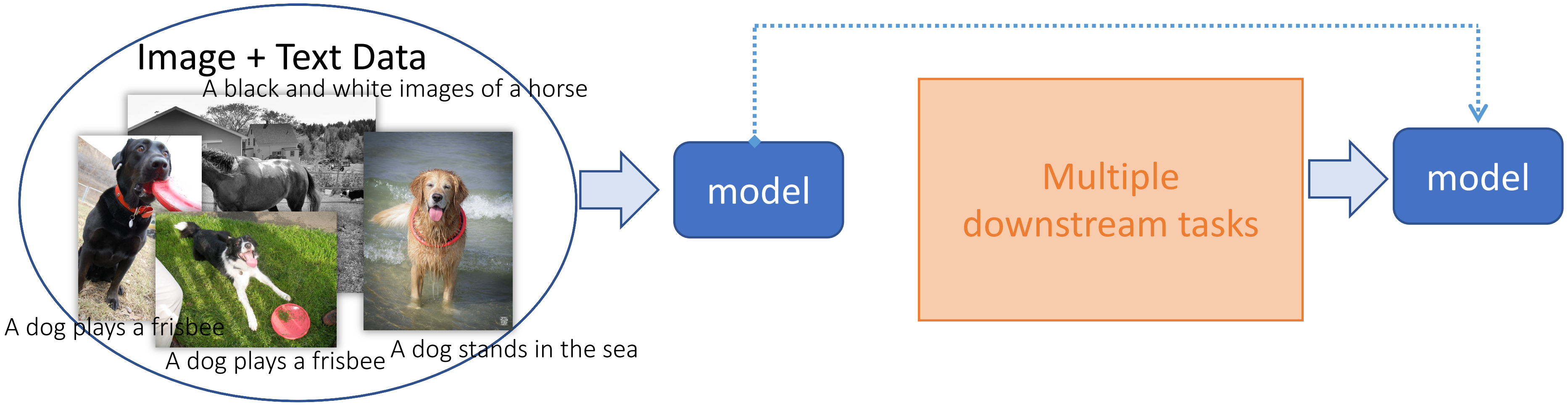


Region-based segmentation
Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:
>>> markers = np.zeros_like(coins)
bed
pickup_truck
bridge
cattle
table
0.00 0.05 0.10
probability
woman
girl
boy
rocket
plain
0.0 0.2 0.4 0.6 0.8
probability
rocket
skyscraper
bus
clock
bridge
0.00 0.25 0.50 0.75 1.00
probability
sweet_pepper
tiger
boy
leopard
squirrel
0.0 0.2 0.4 0.6
probability
cattle
camel
plain
dinosaur
kangaroo
0.0 0.1 0.2 0.3
probability
cup
bowl
plate
table
plain
0.00 0.25 0.50 0.75 1.00
probability
boy
man
plain
television
rocket
0.0 0.1 0.2
probability
motorcycle
bicycle
lawn_mower
pickup_truck
sweet_pepper
0.00 0.25 0.50 0.75 1.00
probability

ALBEF

Align before Fuse: Vision and Language Representation Learning with Momentum Distillation
Paper address: https://arxiv.org/pdf/2107.07651.pdf

- Task: Image-Text Retrieval, Visual Entailment, VQA, NLVR ...

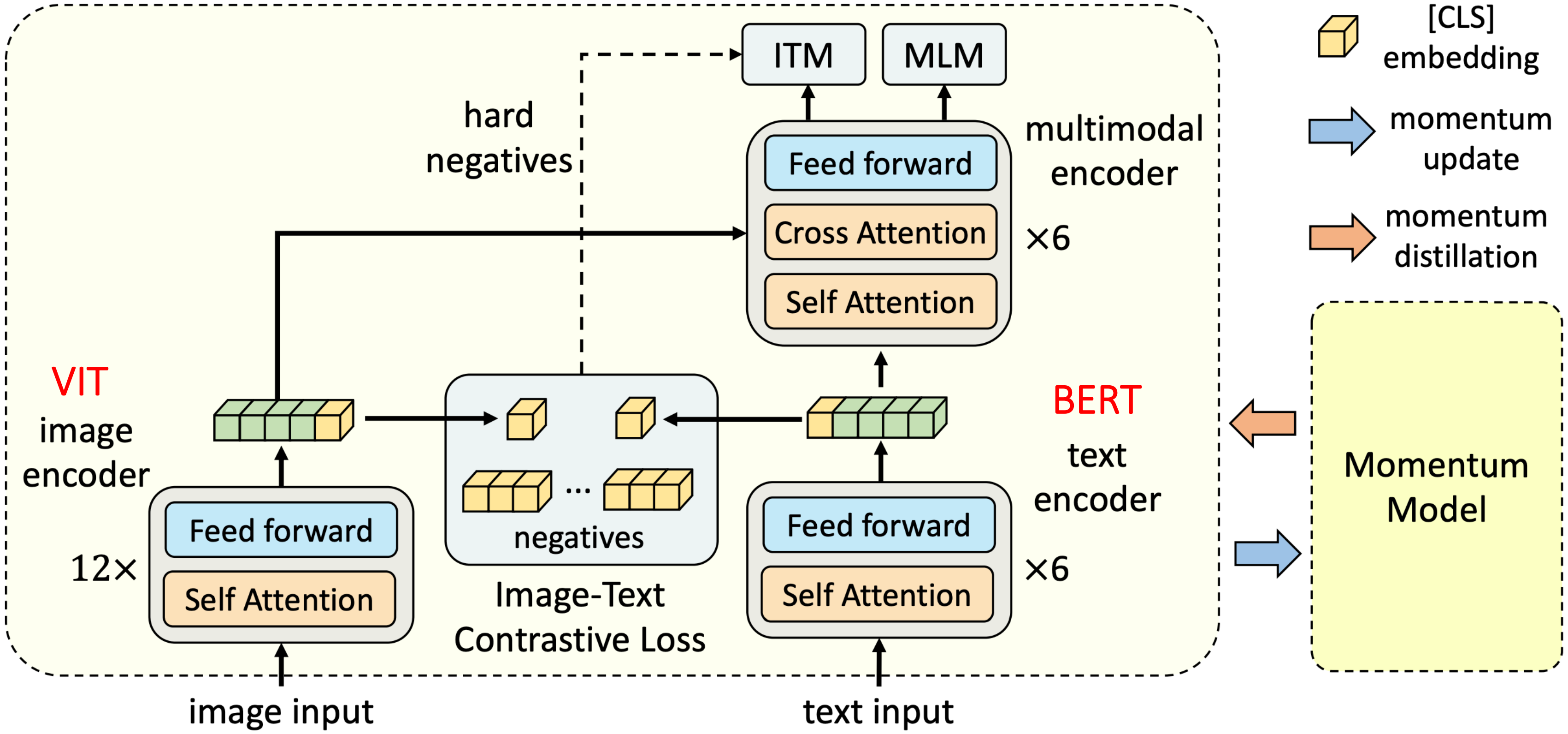


- Goal: use contrastive loss to align image and text tokens and get better multimodal representations for downstream vision-language tasks

ALBEF

Align before Fuse: Vision and Language Representation Learning with Momentum Distillation
Paper address: https://arxiv.org/pdf/2107.07651.pdf

- Model Overview
  - Pretraining:



ALBEF

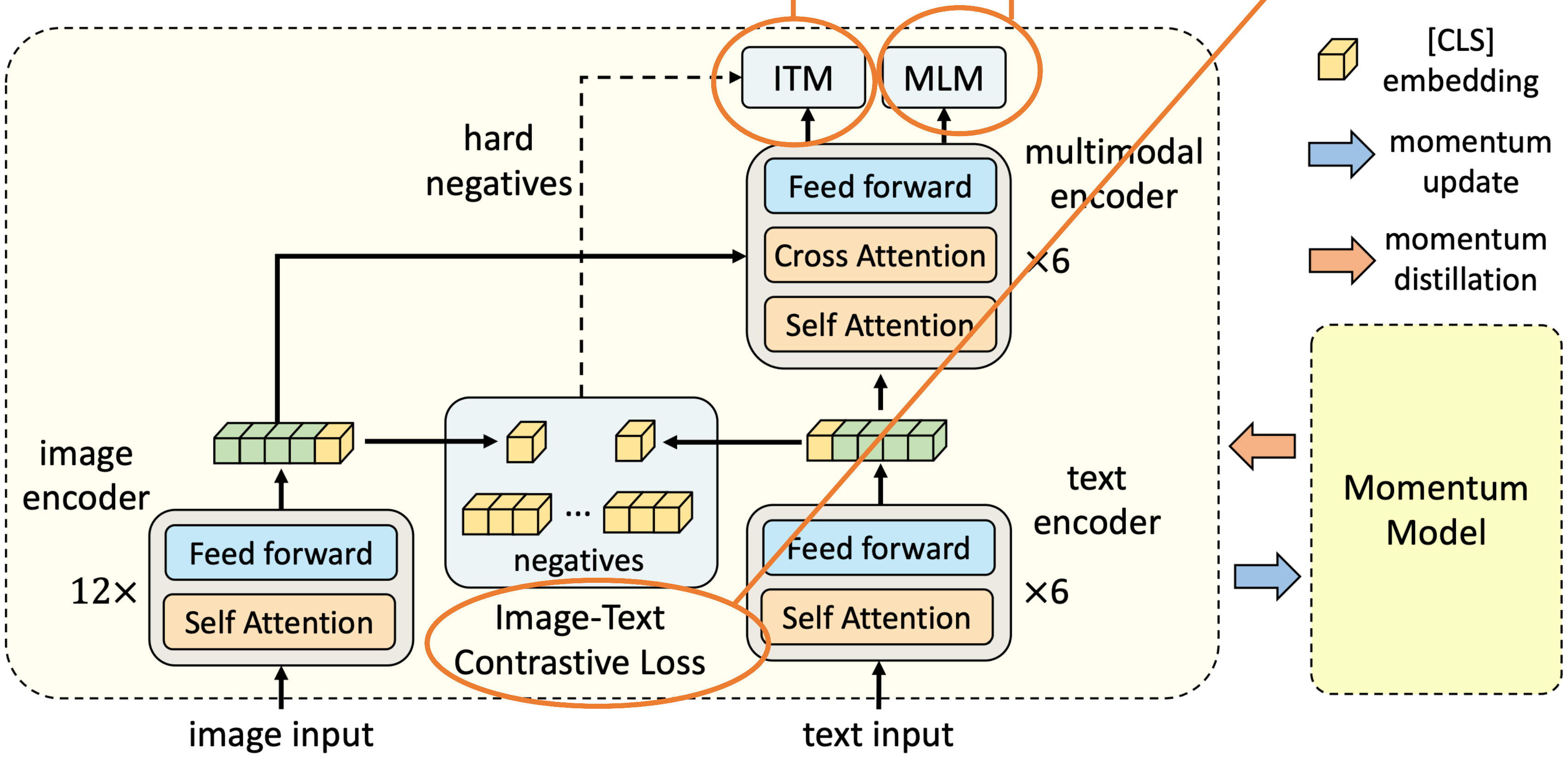
Align before Fuse: Vision and Language Representation Learning with Momentum Distillation
Paper address: https://arxiv.org/pdf/2107.07651.pdf

- Objectives

Cross entropy loss
Label: 0 or 1

Cross entropy loss
Label: masked token indices

Contrastive loss
Label: 0 or 1



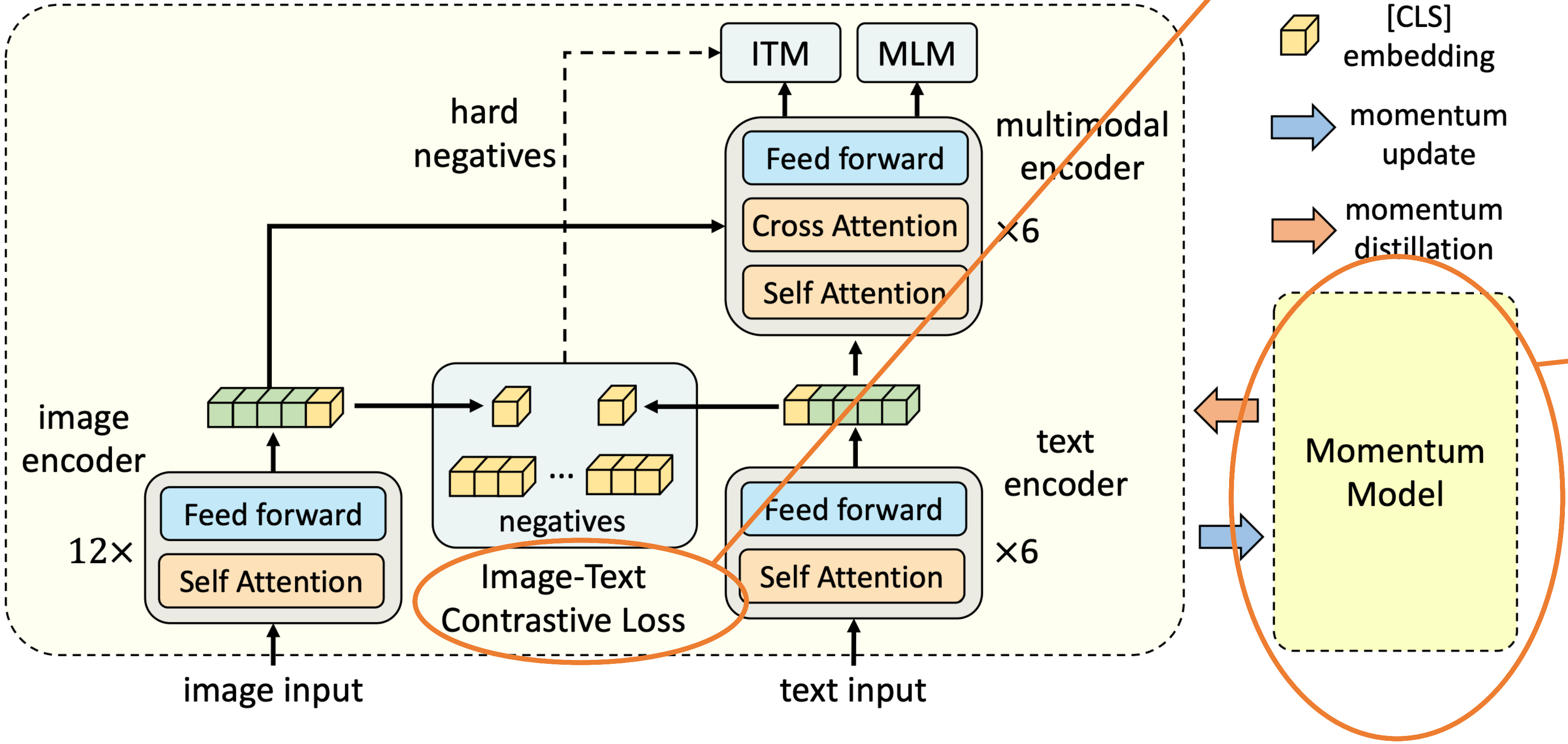
ALBEF

- Objectives
  - Pretraining:

$$s(I,T) = g_v(\boldsymbol{v}_{\text{cls}})^\top g'_w(\boldsymbol{w}'_{\text{cls}}) \qquad s(T,I) = g_w(\boldsymbol{w}_{\text{cls}})^\top g'_v(\boldsymbol{v}'_{\text{cls}})$$

$$p_m^{\text{i2t}}(I) = \frac{\exp(s(I,T_m)/\tau)}{\sum_{m=1}^{M}\exp(s(I,T_m)/\tau)}, \qquad p_m^{\text{t2i}}(T) = \frac{\exp(s(T,I_m)/\tau)}{\sum_{m=1}^{M}\exp(s(T,I_m)/\tau)}$$

$$\mathcal{L}_{\text{itc}} = \frac{1}{2}\mathbb{E}_{(I,T)\sim D}\left[\mathrm{H}(\boldsymbol{y}^{\text{i2t}}(I), \boldsymbol{p}^{\text{i2t}}(I)) + \mathrm{H}(\boldsymbol{y}^{\text{t2i}}(T), \boldsymbol{p}^{\text{t2i}}(T))\right]$$



Momentum encoders

$g'_v(\boldsymbol{v}'_{\text{cls}}) \qquad g'_w(\boldsymbol{w}'_{\text{cls}})$

memory bank

ALBEF

- Code for contrastive loss https://github.com/salesforce/ALBEF:

Model definition

```
self.visual_encoder = VisionTransformer(
    img_size=config['image_res'], 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))

if init_deit:
    checkpoint = torch.hub.load_state_dict_from_url(
        url="https://dl.fbaipublicfiles.com/deit/deit_base_patch16_224-b5f2ef4d.pth",
        map_location="cpu", check_hash=True)
    state_dict = checkpoint["model"]
    pos_embed_reshaped = interpolate_pos_embed(state_dict['pos_embed'], self.visual_encoder)
    state_dict['pos_embed'] = pos_embed_reshaped
    msg = self.visual_encoder.load_state_dict(state_dict,strict=False)
    print(msg)

vision_width = config['vision_width']
bert_config = BertConfig.from_json_file(config['bert_config'])

self.text_encoder = BertForMaskedLM.from_pretrained(text_encoder, config=bert_config)

text_width = self.text_encoder.config.hidden_size
self.vision_proj = nn.Linear(vision_width, embed_dim)
self.text_proj = nn.Linear(text_width, embed_dim)

self.temp = nn.Parameter(torch.ones([]) * config['temp'])
self.queue_size = config['queue_size']
self.momentum = config['momentum']
self.itm_head = nn.Linear(text_width, 2)

# create momentum models
self.visual_encoder_m = VisionTransformer(
    img_size=config['image_res'], 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.vision_proj_m = nn.Linear(vision_width, embed_dim)
self.text_encoder_m = BertForMaskedLM.from_pretrained(text_encoder, config=bert_config)
self.text_proj_m = nn.Linear(text_width, embed_dim)
```

Forward

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

image_feat = F.normalize(self.vision_proj(image_embeds[:,0,:]),dim=-1)

text_output = self.text_encoder.bert(text.input_ids, attention_mask = text.attention_mask,
                                     return_dict = True, mode = 'text')
text_embeds = text_output.last_hidden_state
text_feat = F.normalize(self.text_proj(text_embeds[:,0,:]),dim=-1)

# get momentum features
with torch.no_grad():
    self._momentum_update()
    image_embeds_m = self.visual_encoder_m(image)
    image_feat_m = F.normalize(self.vision_proj_m(image_embeds_m[:,0,:]),dim=-1)
    image_feat_all = torch.cat([image_feat_m.t(),self.image_queue.clone().detach()],dim=1)
    text_output_m = self.text_encoder_m.bert(text.input_ids, attention_mask = text.attention_mask,
                                        return_dict = True, mode = 'text')
    text_feat_m = F.normalize(self.text_proj_m(text_output_m.last_hidden_state[:,0,:]),dim=-1)
    text_feat_all = torch.cat([text_feat_m.t(),self.text_queue.clone().detach()],dim=1)
```

image_feat shape: batch, num_vision_tokens, 256
text_feat shape: batch, num_text_tokens, 256

```
    sim_i2t_m = image_feat_m @ text_feat_all / self.temp
    sim_t2i_m = text_feat_m @ image_feat_all / self.temp

    sim_targets = torch.zeros(sim_i2t_m.size()).to(image.device)
    sim_targets.fill_diagonal_(1)

    sim_i2t_targets = alpha * F.softmax(sim_i2t_m, dim=1) + (1 - alpha) * sim_targets
    sim_t2i_targets = alpha * F.softmax(sim_t2i_m, dim=1) + (1 - alpha) * sim_targets
```

get soft labels

```
sim_i2t = image_feat @ text_feat_all / self.temp
sim_t2i = text_feat @ image_feat_all / self.temp
```

calculate similarity

```
loss_i2t = -torch.sum(F.log_softmax(sim_i2t, dim=1)*sim_i2t_targets,dim=1).mean()
loss_t2i = -torch.sum(F.log_softmax(sim_t2i, dim=1)*sim_t2i_targets,dim=1).mean()

loss_ita = (loss_i2t+loss_t2i)/2
```

calculate contrastive loss

ALBEF

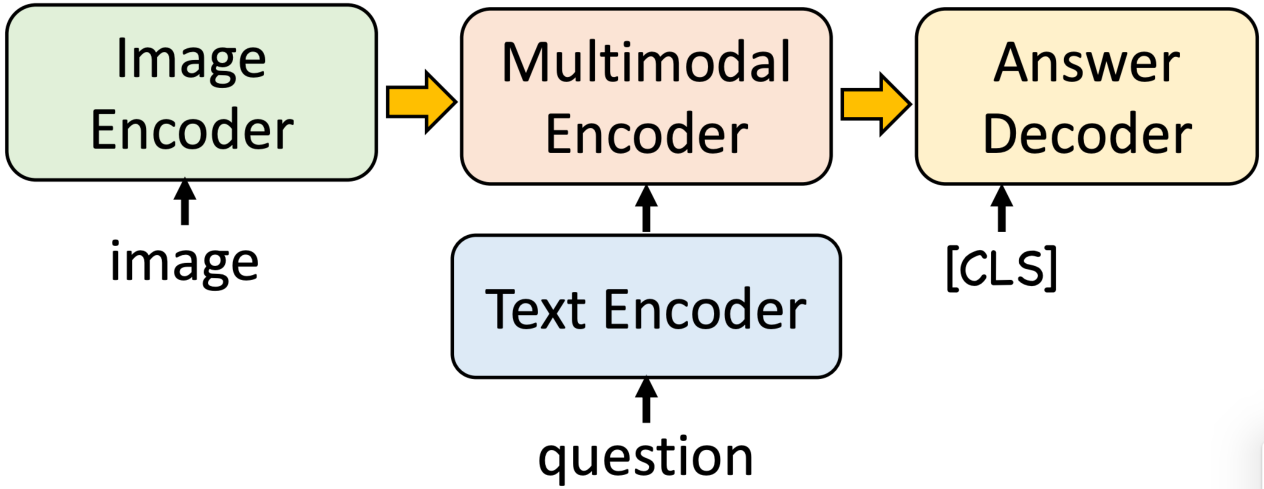
Align before Fuse: Vision and Language Representation Learning with Momentum Distillation
Paper address: https://arxiv.org/pdf/2107.07651.pdf

- Pretraining:
  - Data:
    - Conceptual Captions
    - SBU Captions
    - COCO
    - noisier Conceptual dataset
  - Total number of images: 14.1M

ALBEF

Align before Fuse: Vision and Language Representation Learning with Momentum Distillation
Paper address: https://arxiv.org/pdf/2107.07651.pdf

- Finetuning:
    - Retrieval: use ITM score
    - VQA:



ALBEF

Align before Fuse: Vision and Language Representation Learning with Momentum Distillation
Paper address: https://arxiv.org/pdf/2107.07651.pdf

- Results
  - Retrieval: use ITM score
    - Finetuned on Flickr30K and COCO

Method	# Pre-train Images	Flickr30K (1K test set) TR			Flickr30K (1K test set) IR			MSCOCO (5K test set) TR			MSCOCO (5K test set) IR		
		R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
UNITER	4M	87.3	98.0	99.2	75.6	94.1	96.8	65.7	88.6	93.8	52.9	79.9	88.0
VILLA	4M	87.9	97.5	98.8	76.3	94.2	96.8	-	-	-	-	-	-
OSCAR	4M	-	-	-	-	-	-	70.0	91.1	95.5	54.0	80.8	88.5
ALIGN	1.2B	95.3	99.8	100.0	84.9	97.4	98.6	77.0	93.5	96.9	59.9	83.3	89.8
ALBEF	4M	94.3	99.4	99.8	82.8	96.7	98.4	73.1	91.4	96.0	56.8	81.5	89.2
ALBEF	14M	**95.9**	**99.8**	**100.0**	**85.6**	**97.5**	**98.9**	**77.6**	**94.3**	**97.2**	**60.7**	**84.3**	**90.5**

ALBEF

Align before Fuse: Vision and Language Representation Learning with Momentum Distillation
Paper address: https://arxiv.org/pdf/2107.07651.pdf

- Results
  - Retrieval: use ITM score
    - Zero-shot

Method	# Pre-train Images	Flickr30K (1K test set)					
		TR			IR		
		R@1	R@5	R@10	R@1	R@5	R@10
UNITER [2]	4M	83.6	95.7	97.7	68.7	89.2	93.9
CLIP [6]	400M	88.0	98.7	99.4	68.7	90.6	95.2
ALIGN [7]	1.2B	88.6	98.7	99.7	75.7	93.8	96.8
ALBEF	4M	90.5	98.8	99.7	76.8	93.7	96.7
ALBEF	14M	**94.1**	**99.5**	**99.7**	**82.8**	**96.3**	**98.1**

ALBEF

Align before Fuse: Vision and Language Representation Learning with Momentum Distillation
Paper address: https://arxiv.org/pdf/2107.07651.pdf

- Results
  - Other tasks:

Method	VQA		$NLVR^2$		SNLI-VE	
	test-dev	test-std	dev	test-P	val	test
VisualBERT [13]	70.80	71.00	67.40	67.00	-	-
VL-BERT [10]	71.16	-	-	-	-	-
LXMERT [1]	72.42	72.54	74.90	74.50	-	-
12-in-1 [12]	73.15	-	-	78.87	-	76.95
UNITER [2]	72.70	72.91	77.18	77.85	78.59	78.28
VL-BART/T5 [54]	-	71.3	-	73.6	-	-
ViLT [21]	70.94	-	75.24	76.21	-	-
OSCAR [3]	73.16	73.44	78.07	78.36	-	-
VILLA [8]	73.59	73.67	78.39	79.30	79.47	79.03
ALBEF (4M)	74.54	74.70	80.24	80.50	80.14	80.30
ALBEF (14M)	**75.84**	**76.04**	**82.55**	**83.14**	**80.80**	**80.91**

Questions?