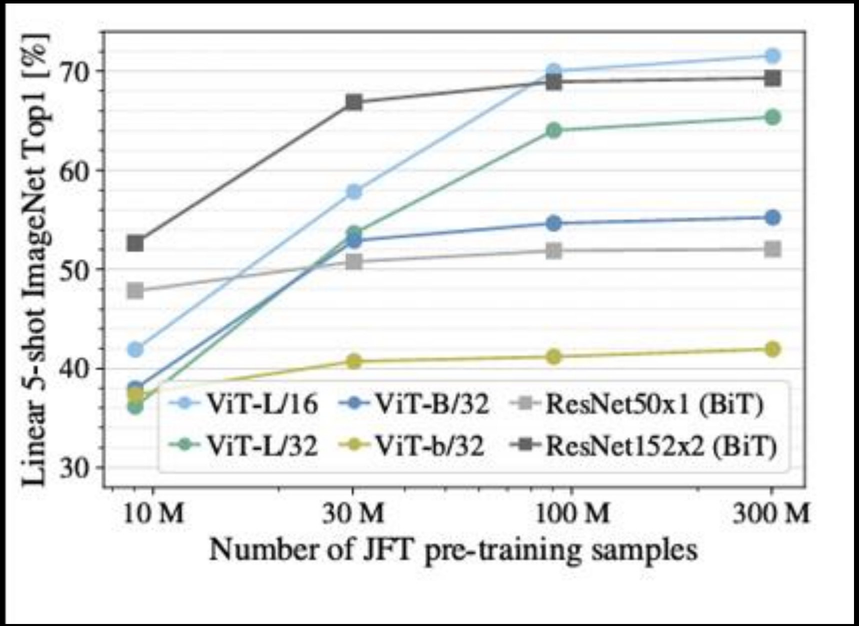
AIL 862

Lecture 15

Pre-Training Data Requirement

- Pre-train on increasing size datasets (ImageNet, ImageNet-21K, JFT-300M)
- Fine tune on target dataset
- Observe peformance

Scaling



Linear 5-shot ImageNet Top1 [%]
70
60
50
40
30
10 M
30 M
100 M
300 M
Number of JFT pre-training samples
ViT-L/16
ViT-B/32
ResNet50x1 (BiT)
ViT-L/32
ViT-b/32
ResNet152x2 (BiT)

Scaling

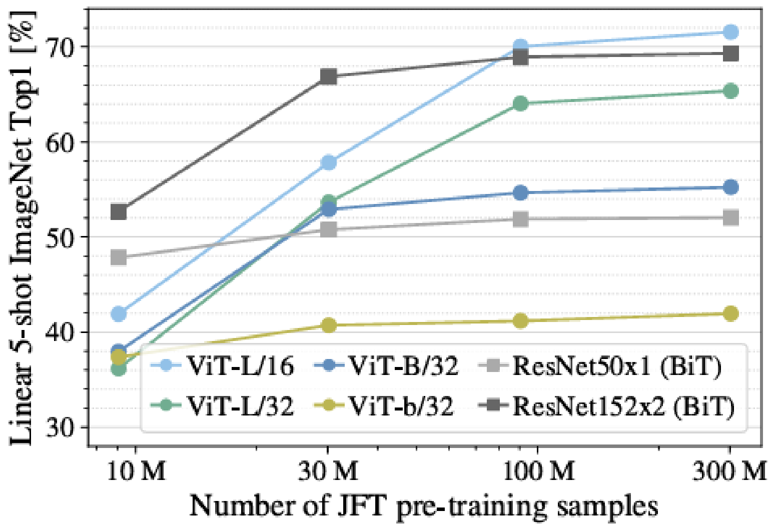


Figure 4: Linear few-shot evaluation on ImageNet versus pre-training size. ResNets perform better with smaller pre-training datasets but plateau sooner than ViT, which performs better with larger pre-training. ViT-b is ViT-B with all hidden dimensions halved.

Pre-Training Data Requirement

Large ViT models perform worse than ResNets when pre-trained on small datasets; however, they excel when pre-trained on larger datasets. Similarly, larger ViT variants surpass smaller ones as the dataset size increases.

Inspecting ViT

- Linear embedding visualization
- Learned position embedding similarity
- How far a patch is paying attention

Some Performances

		ViT-B/16	ViT-B/32	ViT-L/16	ViT-L/32
ImageNet	CIFAR-10	98.13	97.77	97.86	97.94
	CIFAR-100	87.13	86.31	86.35	87.07
	ImageNet	77.91	73.38	76.53	71.16
	ImageNet ReaL	83.57	79.56	82.19	77.83
	Oxford Flowers-102	89.49	85.43	89.66	86.36
	Oxford-IIIT-Pets	93.81	92.04	93.64	91.35
ImageNet-21k	CIFAR-10	98.95	98.79	99.16	99.13
	CIFAR-100	91.67	91.97	93.44	93.04
	ImageNet	83.97	81.28	85.15	80.99
	ImageNet ReaL	88.35	86.63	88.40	85.65
	Oxford Flowers-102	99.38	99.11	99.61	99.19
	Oxford-IIIT-Pets	94.43	93.02	94.73	93.09
JFT-300M	CIFAR-10	99.00	98.61	99.38	99.19
	CIFAR-100	91.87	90.49	94.04	92.52
	ImageNet	84.15	80.73	87.12	84.37
	ImageNet ReaL	88.85	86.27	89.99	88.28
	Oxford Flowers-102	99.56	99.27	99.56	99.45
	Oxford-IIIT-Pets	95.80	93.40	97.11	95.83

Transformer Shape

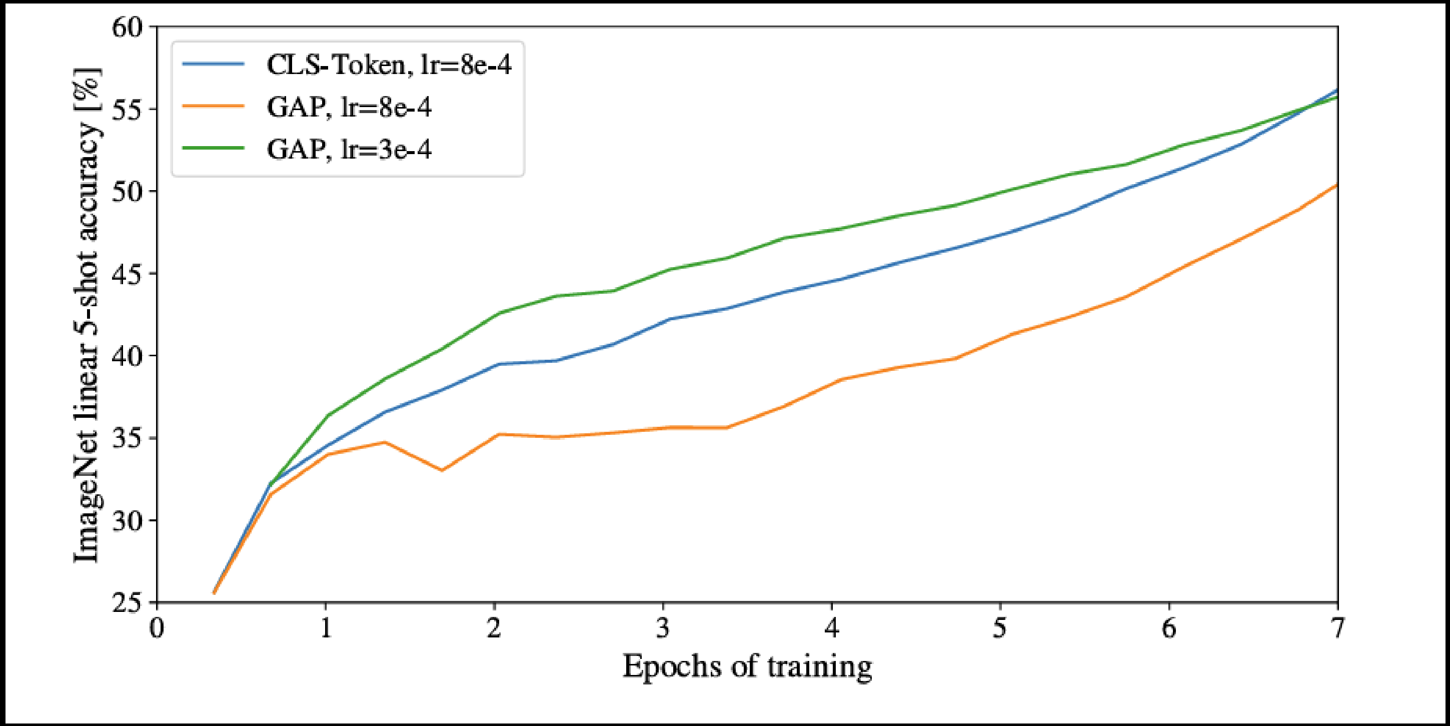
Decreasing the patch size and thus increasing the effective sequence length shows robust improvements without introducing parameters

Look Back at the Performances

		ViT-B/16	ViT-B/32	ViT-L/16	ViT-L/32
ImageNet	CIFAR-10	98.13	97.77	97.86	97.94
	CIFAR-100	87.13	86.31	86.35	87.07
	ImageNet	77.91	73.38	76.53	71.16
	ImageNet ReaL	83.57	79.56	82.19	77.83
	Oxford Flowers-102	89.49	85.43	89.66	86.36
	Oxford-IIIT-Pets	93.81	92.04	93.64	91.35
ImageNet-21k	CIFAR-10	98.95	98.79	99.16	99.13
	CIFAR-100	91.67	91.97	93.44	93.04
	ImageNet	83.97	81.28	85.15	80.99
	ImageNet ReaL	88.35	86.63	88.40	85.65
	Oxford Flowers-102	99.38	99.11	99.61	99.19
	Oxford-IIIT-Pets	94.43	93.02	94.73	93.09
JFT-300M	CIFAR-10	99.00	98.61	99.38	99.19
	CIFAR-100	91.87	90.49	94.04	92.52
	ImageNet	84.15	80.73	87.12	84.37
	ImageNet ReaL	88.85	86.27	89.99	88.28
	Oxford Flowers-102	99.56	99.27	99.56	99.45
	Oxford-IIIT-Pets	95.80	93.40	97.11	95.83

If not using class token

If not using class token



Positional embedding a bit more

A few options

- No embedding

A few options

- 1-dimensional positional embedding

Assigns a unique embedding to each patch based on its position in a flattened sequence

A few options

- 2-dimensional positional embedding

Pos. Emb.	Default/Stem
No Pos. Emb.	0.61382
1-D Pos. Emb.	0.64206
2-D Pos. Emb.	0.64001
Rel. Pos. Emb.	0.64032

Different size/resolution

The Vision Transformer can handle arbitrary sequence, however, the pre-trained position embeddings may no longer be meaningful. May need to perform interpolation of the pre-trained position embeddings, according to their location in the original image.

Overlapping patches

Windowed attentions

Introducing Convolutions to ViTs

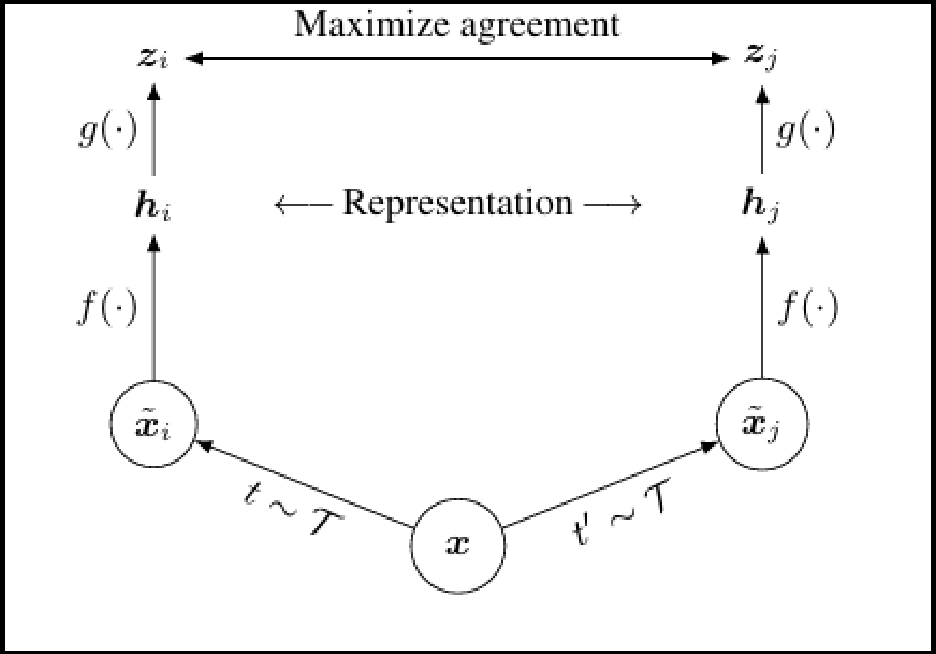
CvT : Introducing Convolutions to Vision Transformers

Self-supervision

- Not explored in details in the original ViT paper

A bit recap from the SSL lectures

SimCLR



A Simple Framework for Contrastive Learning of Visual Representations, 2020

SimCLR

Learning algorithm

Composition of data augmentation

Helps

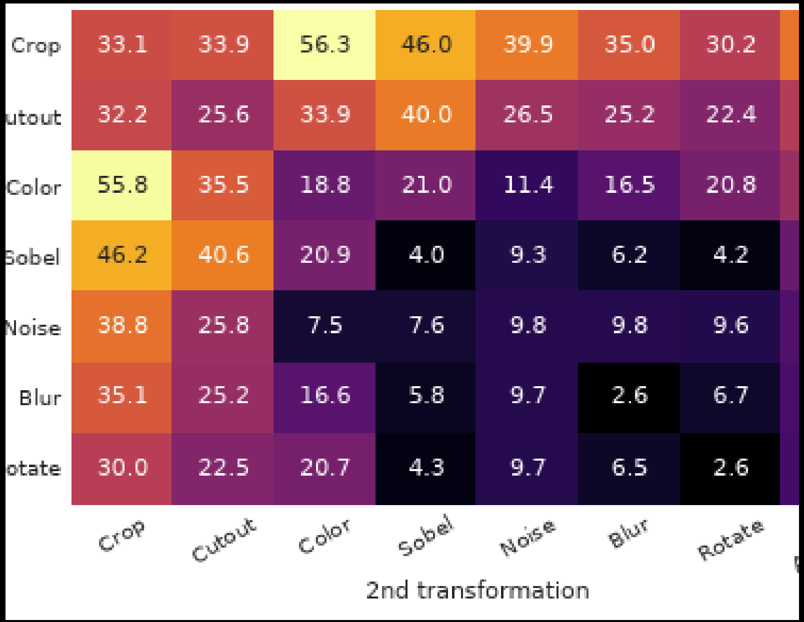
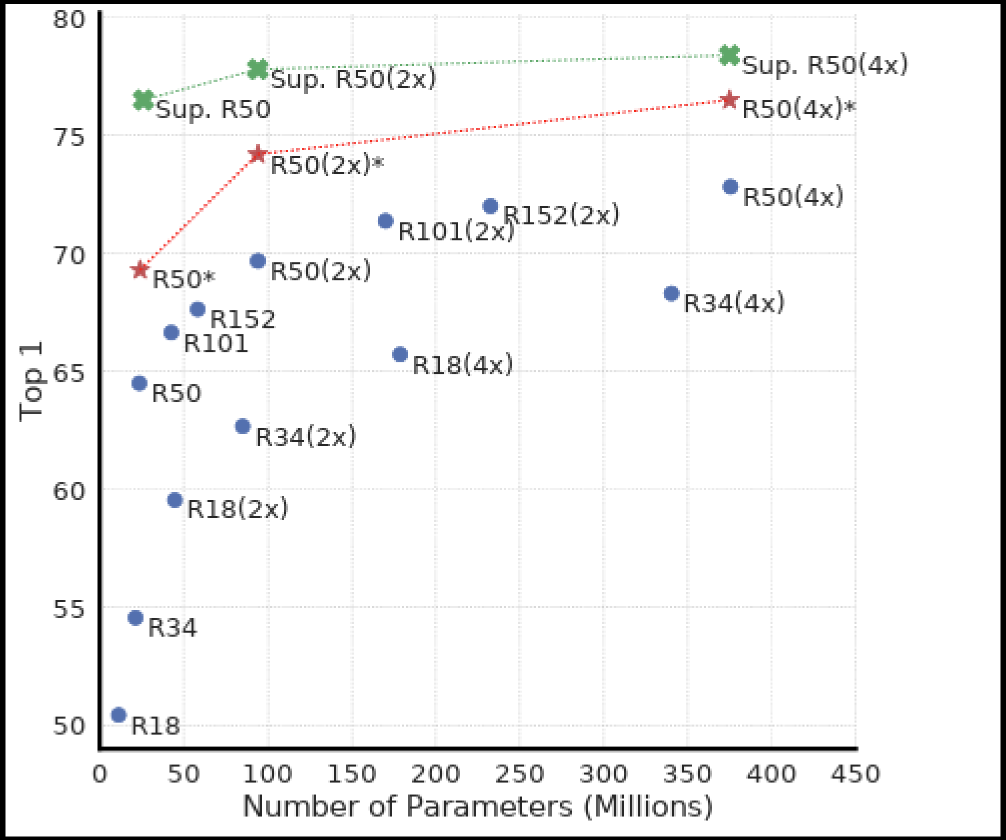


Figure 5. Linear evaluation (ImageNet top-1 accuracy) under individual or composition of data augmentations, applied only to one branch. For all columns but the last, diagonal entries correspond to single transformation, and off-diagonals correspond to composition of two transformations (applied sequentially). The

Benefits from bigger models



Batch size

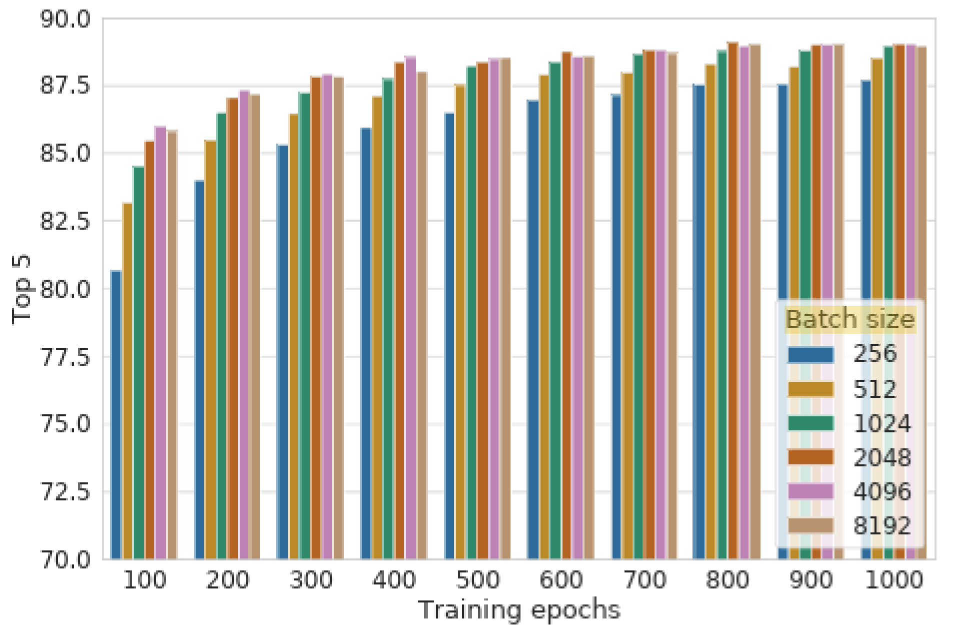


Figure B.1. Linear evaluation (top-5) of ResNet-50 trained with different batch sizes and epochs. Each bar is a single run from scratch. See Figure 9 for top-1 accuracy.

Default batch size - 4096

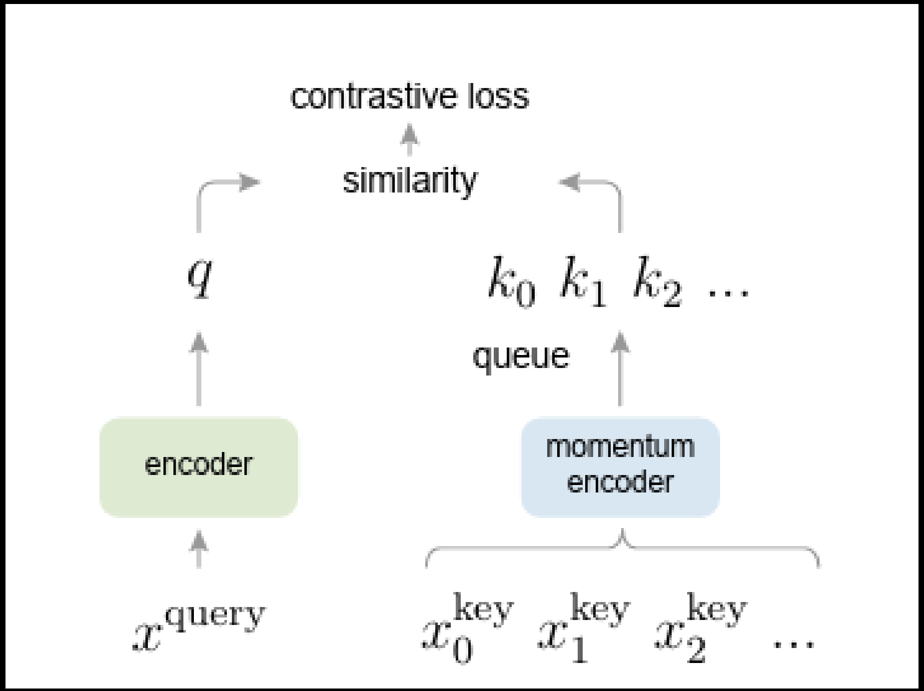
Pre-text invariant representation learning

Main working principle similar to the SimCLR, however uses memory bank

PIRL memory bank

- The memory bank contains feature representation of each original image (without transformation) in the dataset.
- Memory bank allows us to replace negative terms in the loss function with their memory bank representation, without increasing training batch size.

MoCo



Momentum Contrast for Unsupervised Visual Representation Learning

Negative samples - difficulty

- Challenging to obtain
- Geographically “farther-closer” does not always ensure that we indeed got negative samples
- Quality of negative samples may hinder contrastive learning

Bootstrap your own latent

- No pseudo-label

- Unlike previous contrastive methods, no negative sample

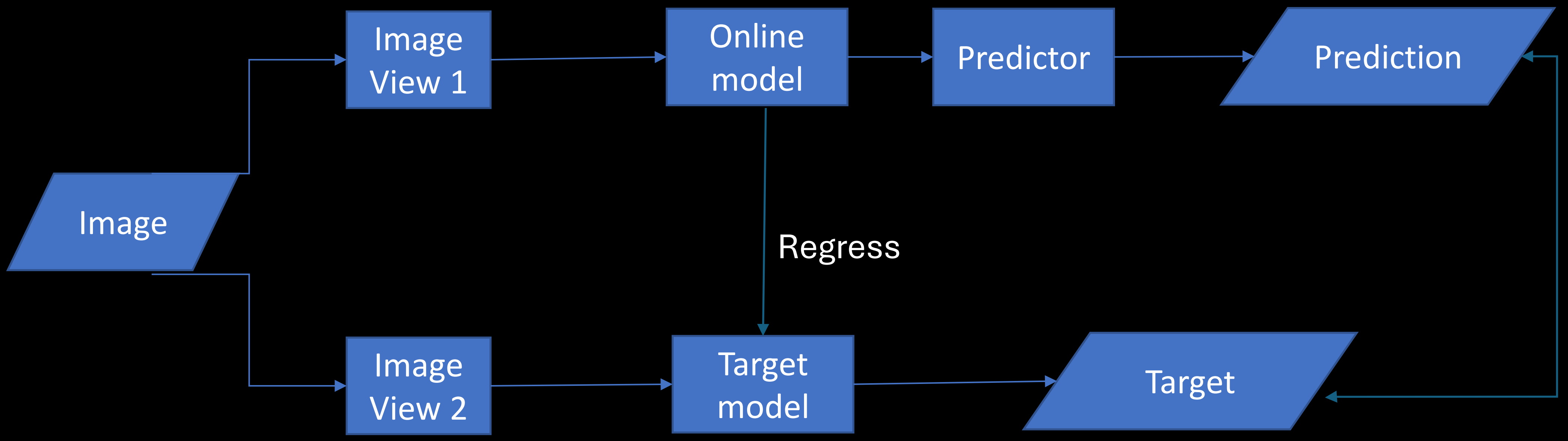
❑ Negative samples are expensive

- Training procedure is simple

BYOL

- Image: generate two views
- Two different networks: online network and target network
- One input is processed through the online network and the other through the target network

BYOL - mechanism



Algorithm 1: BYOL: **B**ootstrap **Y**our **O**wn **L**atent

Inputs :

$\mathcal{D}$, $\mathcal{T}$, and $\mathcal{T}'$	set of images and distributions of transformations
θ, f_θ, g_θ, and q_θ	initial online parameters, encoder, projector, and predictor
ξ, f_ξ, g_ξ	initial target parameters, target encoder, and target projector
optimizer	optimizer, updates online parameters using the loss gradient
K and N	total number of optimization steps and batch size
$\{\tau_k\}_{k=1}^K$ and $\{\eta_k\}_{k=1}^K$	target network update schedule and learning rate schedule

for $k = 1$ **to** K **do**
- $\mathcal{B} \leftarrow \{x_i \sim \mathcal{D}\}_{i=1}^N$ `// sample a batch of` N `images`
- **for** $x_i \in \mathcal{B}$ **do**
  - $t \sim \mathcal{T}$ and $t' \sim \mathcal{T}'$ `// sample image transformations`
  - $z_1 \leftarrow g_\theta(f_\theta(t(x_i)))$ and $z_2 \leftarrow g_\theta(f_\theta(t'(x_i)))$ `// compute projections`
  - $z'_1 \leftarrow g_\xi(f_\xi(t'(x_i)))$ and $z'_2 \leftarrow g_\xi(f_\xi(t(x_i)))$ `// compute target projections`
  - $l_i \leftarrow -2 \cdot \left(\frac{\langle q_\theta(z_1), z'_1 \rangle}{\|q_\theta(z_1)\|_2 \cdot \|z'_1\|_2} + \frac{\langle q_\theta(z_2), z'_2 \rangle}{\|q_\theta(z_2)\|_2 \cdot \|z'_2\|_2} \right)$ `// compute the loss for` x_i
- **end**
- $\delta\theta \leftarrow \frac{1}{N} \sum_{i=1}^{N} \partial_\theta l_i$ `// compute the total loss gradient w.r.t.` θ
- $\theta \leftarrow \text{optimizer}(\theta, \delta\theta, \eta_k)$ `// update online parameters`
- $\xi \leftarrow \tau_k \xi + (1 - \tau_k)\theta$ `// update target parameters`

end

Output : encoder f_θ

Ack:
https://theaisummer.com/byol/

```python
class Augment:
    """
    A stochastic data augmentation module
    Transforms any given data example randomly
    resulting in two correlated views of the same example,
    denoted x ĩ and x j̃, which we consider as a positive pair.
    """
    def __init__(self, img_size, s=1):
        color_jitter = T.ColorJitter(
            0.8 * s, 0.8 * s, 0.8 * s, 0.2 * s
        )
        blur = T.GaussianBlur((3, 3), (0.1, 2.0))

        self.train_transform = T.Compose([
            T.ToTensor(),
            T.RandomResizedCrop(size=img_size),
            T.RandomHorizontalFlip(p=0.5),  # with 0.5 probability
            T.RandomApply([color_jitter], p=0.8),
            T.RandomApply([blur], p=0.5),
            T.RandomGrayscale(p=0.2),
            # imagenet stats
            T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
        ])
    def __call__(self, x):
        return self.train_transform(x), self.train_transform(x),
```

```python
        self.net = net
        self.student_model = AddProjHead(model=net, in_features=in_features,
                                         layer_name=layer_name,
                                         embedding_size=projection_size,
                                         hidden_size=projection_hidden_size,
                                         batch_norm_mlp=batch_norm_mlp)
        self.use_momentum = use_momentum
        self.teacher_model = self._get_teacher()
        self.target_ema_updater = EMA(moving_average_decay)
        self.student_predictor = MLP(projection_size, projection_size, projection_hidden_size)


    @torch.no_grad()
    def _get_teacher(self):
        return copy.deepcopy(self.student_model)
```

```python
# student projections: backbone + MLP projection
student_proj_one = self.student_model(image_one)
student_proj_two = self.student_model(image_two)

# additional student's MLP head called predictor
student_pred_one = self.student_predictor(student_proj_one)
student_pred_two = self.student_predictor(student_proj_two)

with torch.no_grad():
    # teacher processes the images and makes projections: backbone + MLP
    teacher_proj_one = self.teacher_model(image_one).detach_()
    teacher_proj_two = self.teacher_model(image_two).detach_()

loss_one = loss_fn(student_pred_one, teacher_proj_one)
loss_two = loss_fn(student_pred_two, teacher_proj_two)

return (loss_one + loss_two).mean()
```

```python
def loss_fn(x, y):
    # L2 normalization
    x = F.normalize(x, dim=-1, p=2)
    y = F.normalize(y, dim=-1, p=2)
    return 2 - 2 * (x * y).sum(dim=-1)
```

```python
class EMA():
    def __init__(self, alpha):
        super().__init__()
        self.alpha = alpha

    def update_average(self, old, new):
        if old is None:
            return new
        return old * self.alpha + (1 - self.alpha) * new
```

Fine-tuning with small dataset

1% means fine-tuned with only 1% of ImageNet's training set

Method	Top-1		Top-5	
	1%	10%	1%	10%
Supervised [77]	25.4	56.4	48.4	80.4
InstDisc	-	-	39.2	77.4
PIRL [35]	-	-	57.2	83.8
SimCLR [8]	48.3	65.6	75.5	87.8
BYOL (ours)	**53.2**	**68.8**	**78.4**	**89.0**

(a) ResNet-50 encoder.

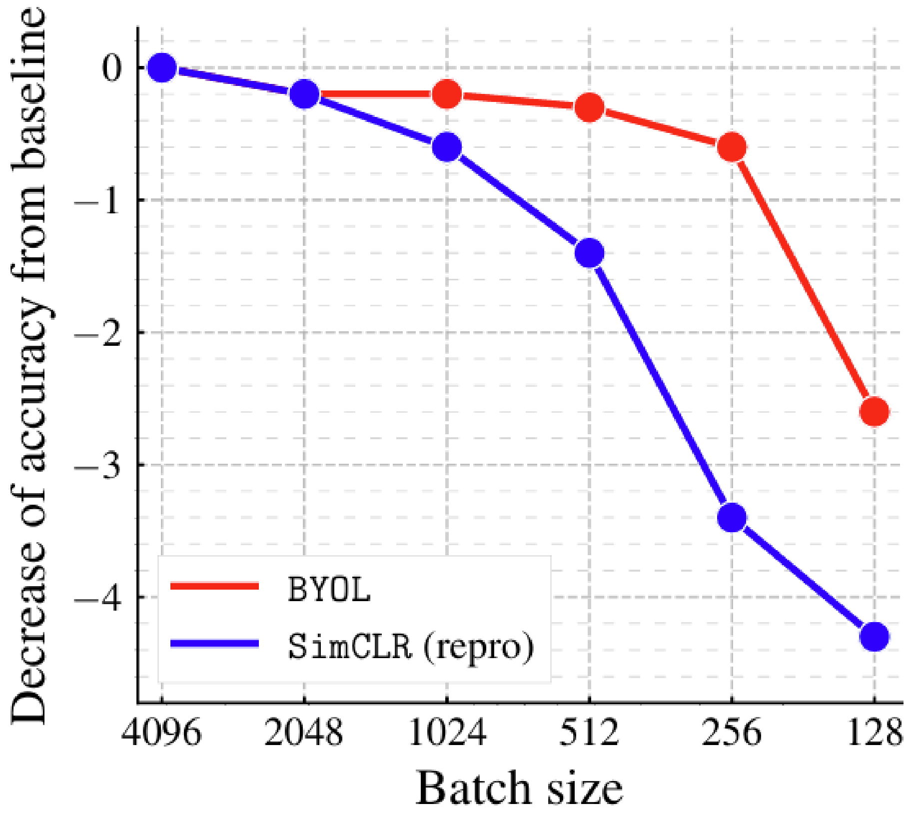
BYOL - sensitive

- To augmentation choice
- Projection dimension
- To batch size

Projector g_θ output dim	Top-1	Top-5
16	$69.9_{\pm 0.3}$	89.9
32	71.3	90.6
64	72.2	90.9
128	72.5	91.0
256	72.5	90.8
512	**72.6**	**91.0**

(b) Projection dimension.

Batch Size



(a) Impact of batch size

SSL evaluation

- Linear classification
- Data efficiency (e.g., fine-tune with 1% of actual training dataset)
- K-nearest neighbor