# PJ presentation

Group 9 戴琪润，封启骏，王乐祺，王嗣超，吴欣怡

School of Computer Science, Fudan University

2023 年 6 月 2 日

# 模型的选择

- EffiecientNet: 纯卷积模型
- ViT: Vision Transformer-Deit
  - Deit
  - Deit distill
- ViT: Vision Transformer-Swin
- GCViT:Hybrid Model

# EffiecientNet
## 代码实现

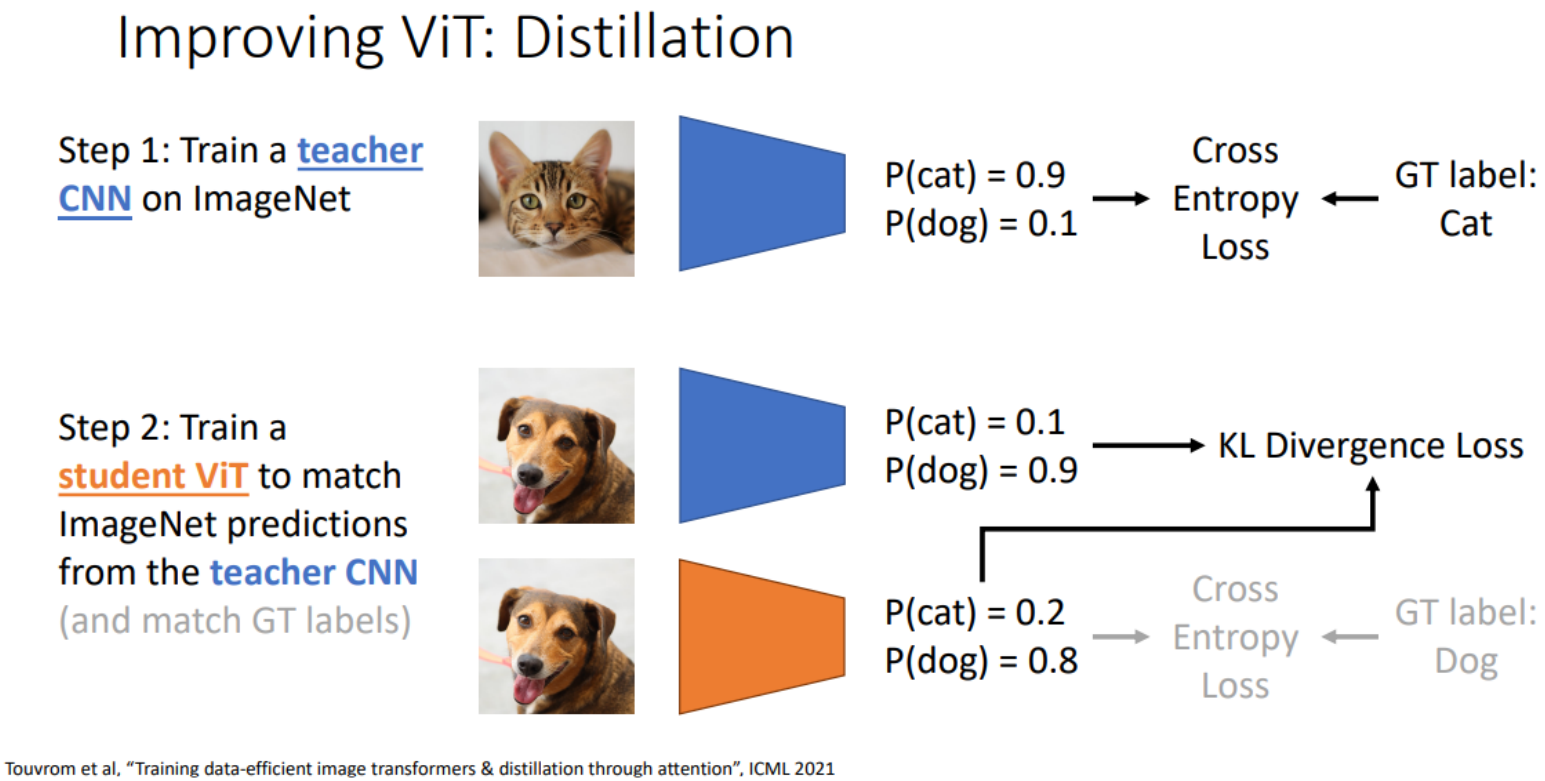
```python
# 4. EfficientNet/Hybrid: MaxVit forward_features 输出形状为 (N, C, H, W);
elif 'efficientnet' in self.model_name or self.model_name in {
    'maxvit_tiny_rw_224', 'gcvit_tiny', 'gcvit_xtiny', 'gcvit_small'
}:
    outputs = self.channel_pool(feature)
    outputs = self.head(outputs)
    return outputs
```

图 1: EfficientNet

# Vision Transformer-Deit

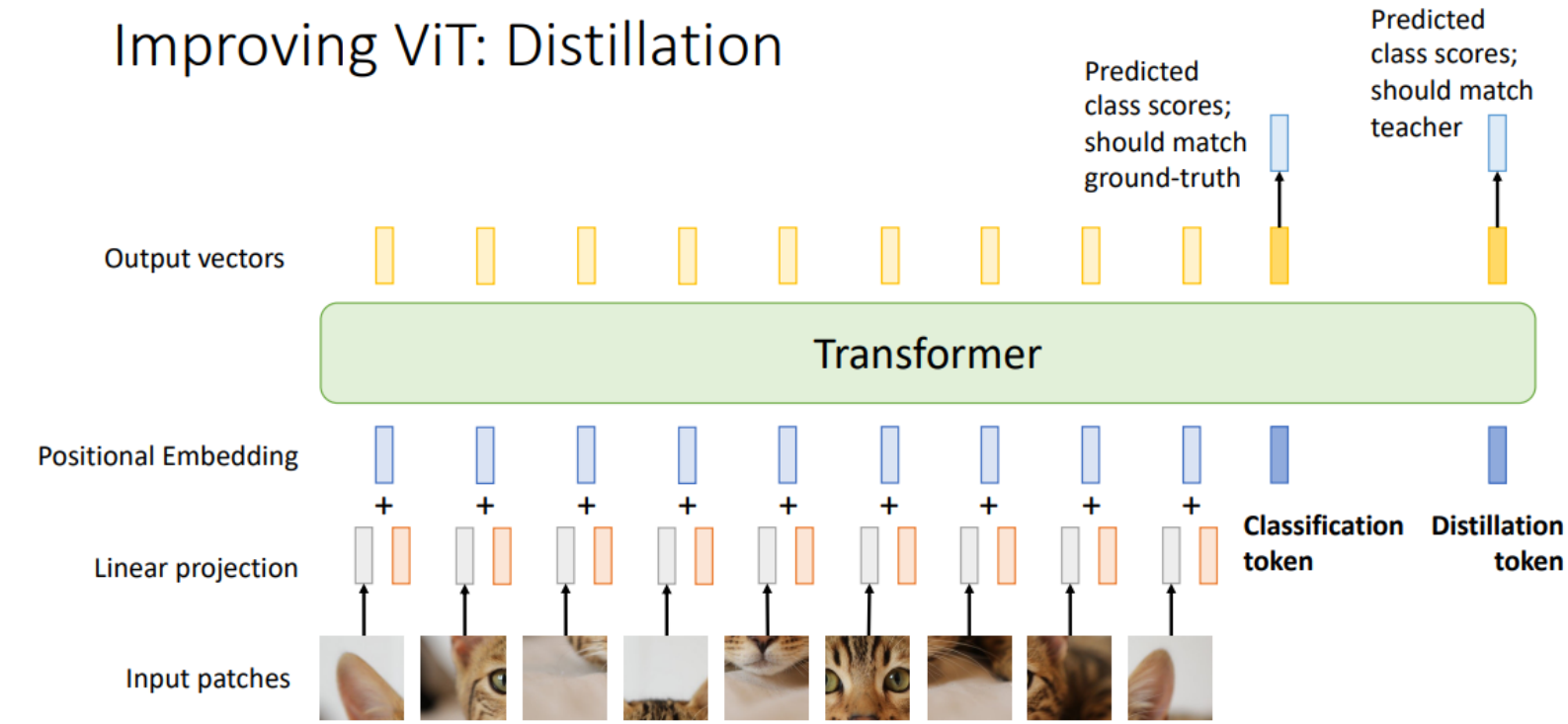
## 简介

- 高效的 training



# Vision Transformer-Deit

## 简介




Touvrom et al, "Training data-efficient image transformers & distillation through attention", ICML 2021

# Vision Transformer-Deit
## 代码实现

```python
# 1. deit(no distillation): 只有 cls_token
if self.model_name in {'deit_small_patch16_224', 'deit_base_patch16_224'}:
    outputs = self.channel_bn(torch.squeeze(feature[:, 0, :], dim=1))
    outputs = self.head(outputs)
    return outputs
```

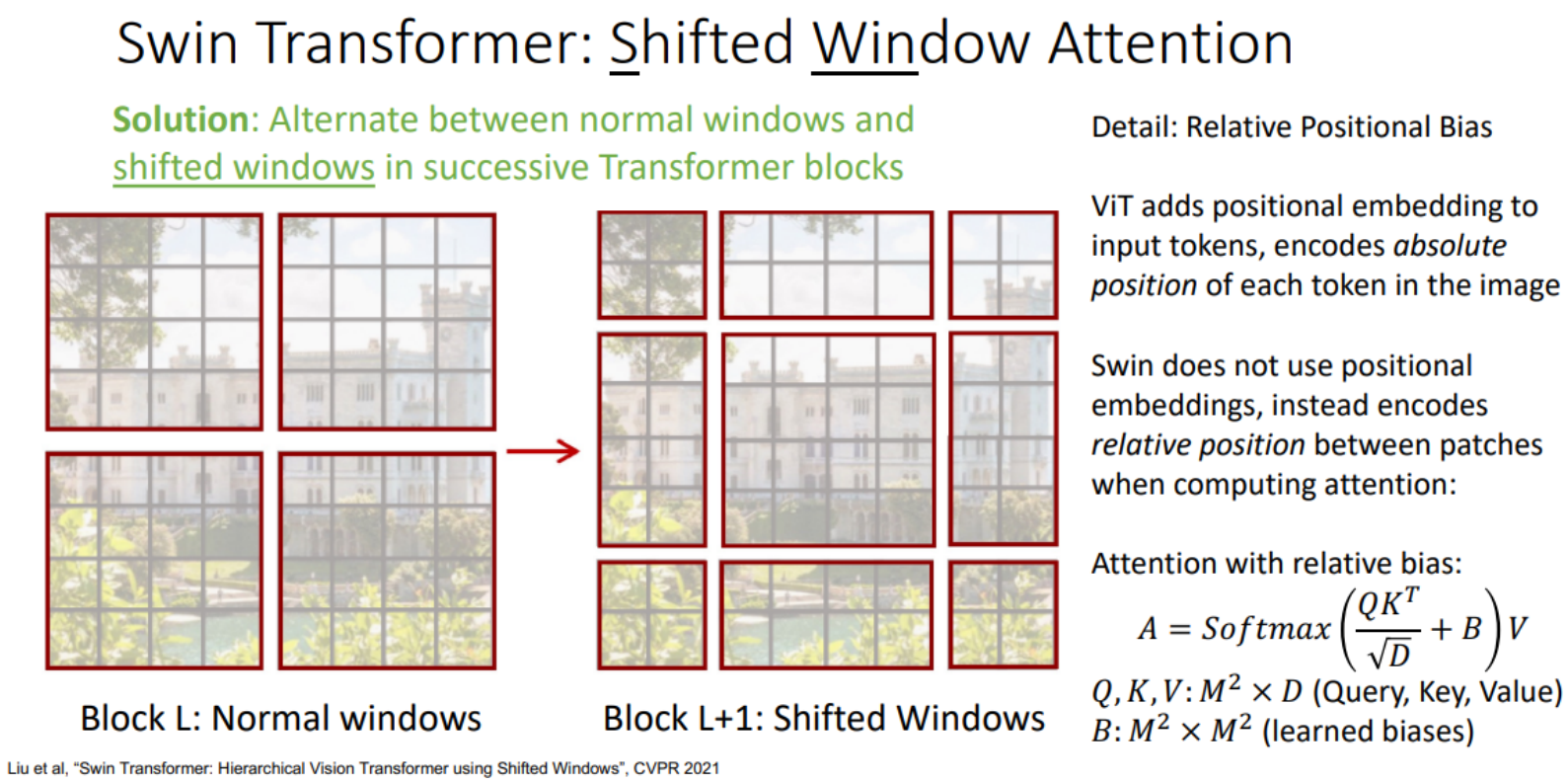
图 2: Deit without distillation

```python
# 2. deit(distilled): cls_token + dist_token
elif self.model_name in {'deit_small_distilled_patch16_224', 'deit_base_distilled_patch16_224'
    cls, dist = feature[:, 0], feature[:, 1]
    cls = self.head(cls)
    dist = self.head_dist(dist)
    return (cls + dist) / 2
```

图 3: Deit with distillation

# Vision Transformer-Swin
## 简介

- 高效的模型架构



# Vision Transformer-Swin
## 代码实现

```python
# 3. swin-transformer: (N,L,C) + mean(dim=1)
elif self.model_name in {'swin_tiny_patch4_window7_224', 'swin_small_patch4_window7_224'}:
    feature = feature.mean(dim=1)
    feature = self.head(feature)
    return feature
```

图 4: Swin Transformer

# GCViT

## 简介

- 卷积的 inductive-bias 优势
- vit 的长序建模优势

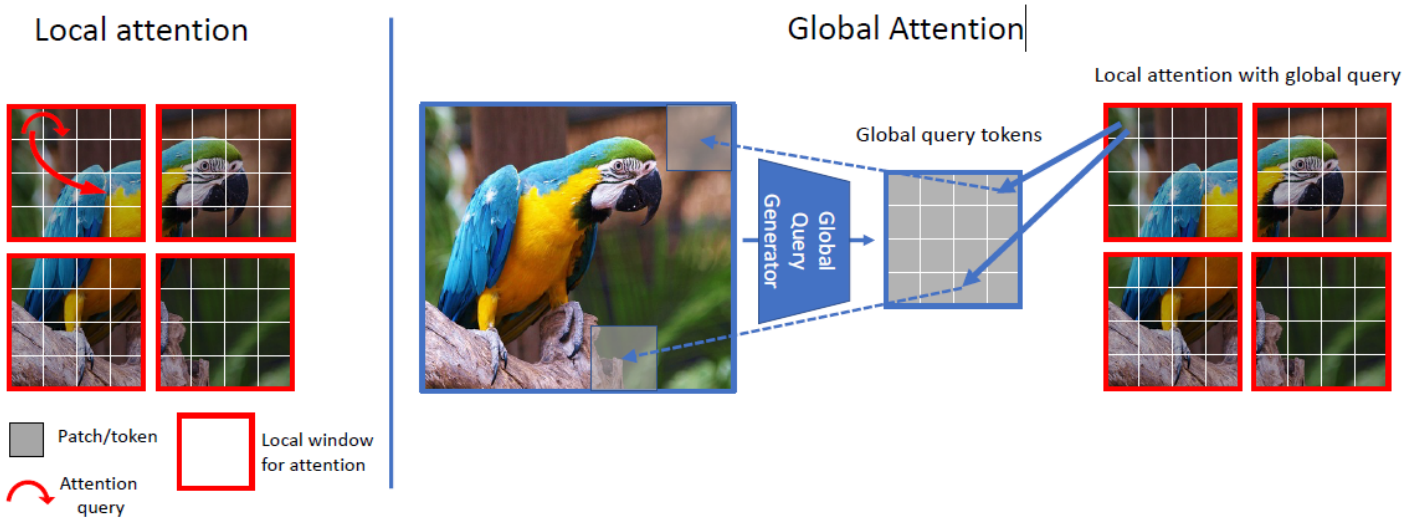



**Figure 4** – Attention formulation. Local attention is computed on feature patches within local window only (left). On the other hand, the global features are extracted from the entire input features and then repeated to form global query tokens. The global query is interacted with local key and value tokens, hence allowing to capture long-range information via cross-region interaction. Best viewed in color.

# GCViT
## 代码实现

```python
# 4. EfficientNet/Hybrid: MaxVit forward_features 输出形状为 (N, C, H, W);
elif 'efficientnet' in self.model_name or self.model_name in {
    'maxvit_tiny_rw_224', 'gcvit_tiny', 'gcvit_xtiny', 'gcvit_small'
}:
    outputs = self.channel_pool(feature)
    outputs = self.head(outputs)
    return outputs
```

图 5: EfficientNet

# 数据增强

```python
def build_transform(is_train, args, img_size=224,
                        mean=IMAGENET_DEFAULT_MEAN, std=IMAGENET_DEFAULT_STD):
    # TODO: does any other data augmentation work better?
    # 只有在 train 时才做 augmentation; val 与 test 时都不做 augmentation.
    if is_train:
        t = []
        t.append(transforms.Resize(img_size))
        t.append(transforms.CenterCrop(img_size))
        # if args.flip:   # no flip by default
        #     t.append(transforms.RandomVerticalFlip(p = args.flip))
        #     t.append(transforms.RandomHorizontalFlip(p = args.flip))
        # if args.rotation:   # no rotation by default
        #     t.append(transforms.RandomRotation(args.rotation))
        if args.rand_aug:
            t.append(transforms.RandAugment(num_ops=2, magnitude=8))
        t.append(transforms.ToTensor())
        t.append(transforms.Normalize(mean, std))
        return transforms.Compose(t)

    t = []
    t.append(transforms.Resize(img_size))
    t.append(transforms.CenterCrop(img_size))
    t.append(transforms.ToTensor())
    t.append(transforms.Normalize(mean, std))
    return transforms.Compose(t)
```

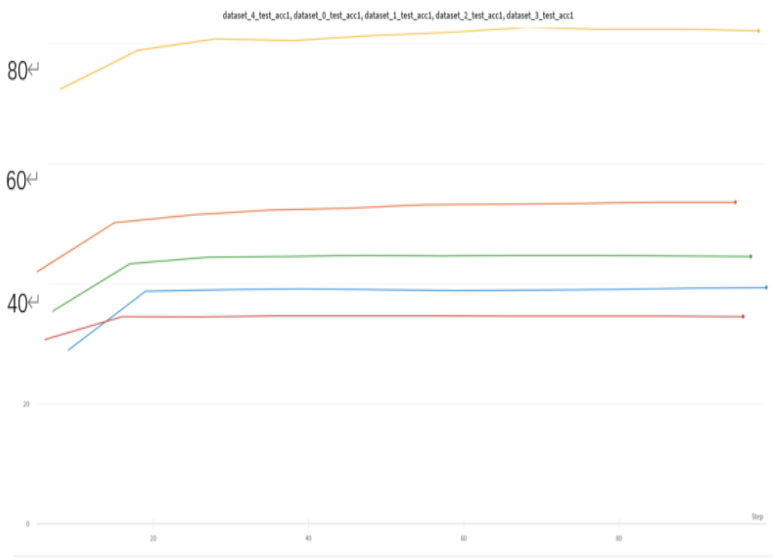
图 6: build_transform

# 数据增强

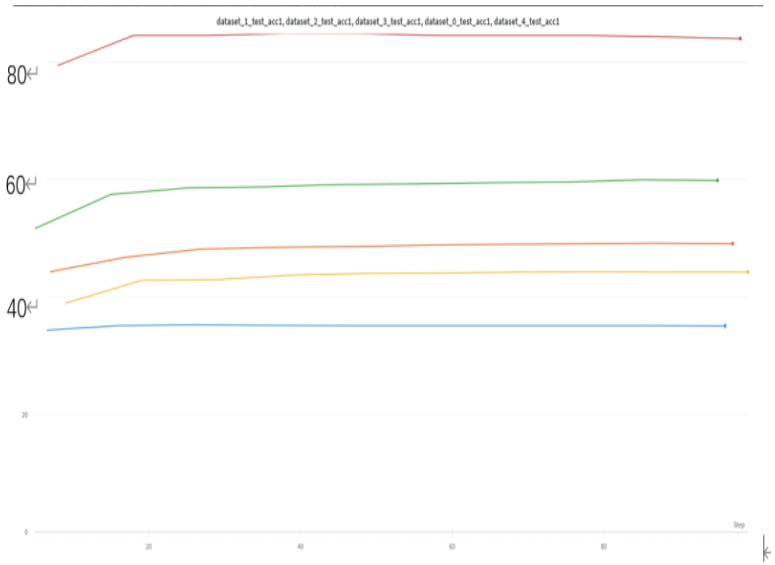
```python
def __getitem__(self, index) :
    """
    Returns:
        sample: the tensor of the input image
        target: a int tensor of class id
        dataset_id: a int number indicating the dataset id
    """
    path, target, dataset_id = self.samples[index]
    sample = self.loader(path)

    if self.transform is not None:
        # 1/3 的概率下, 不做 augmentation, 返回原图;
        if torch.rand(1).item() < 0.333:
            sample = self.std_transform(sample)
        else:
            sample = self.transform(sample) # 数据集的 transform 在 MultiImageFolder 取数据的同时才会做.

    # 返回的数据类型 (tensor, int tensor, int number)
    return sample, target, dataset_id
```

图 7: ___get_item___

# 实验结果




(a) deit_small_distilled




(b) deit_base_distilled

# 实验结果

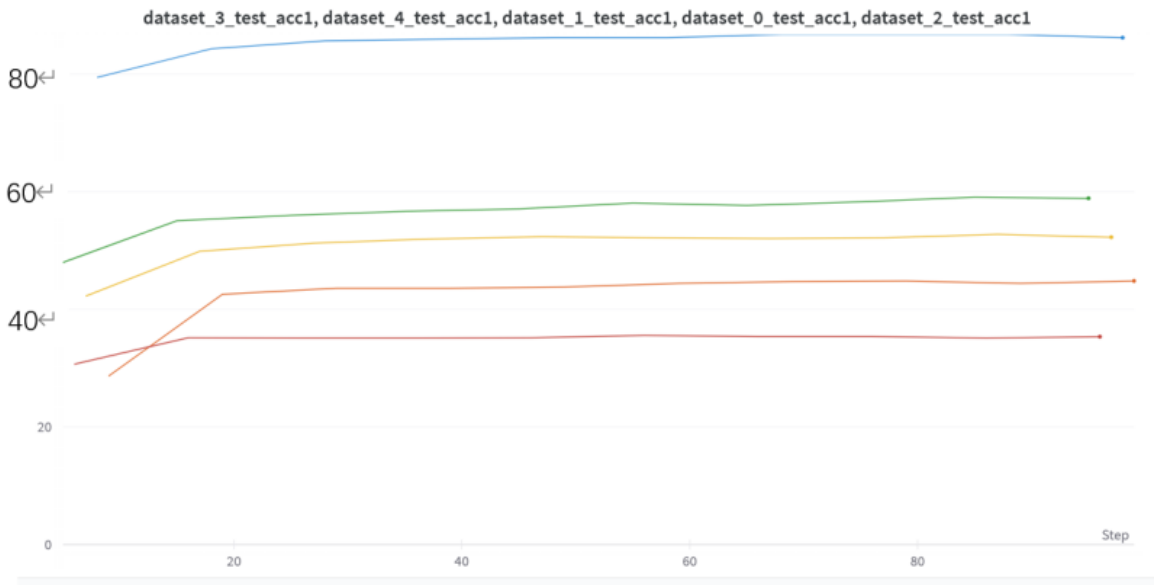



图 8: swin_transformer

# 实验结果

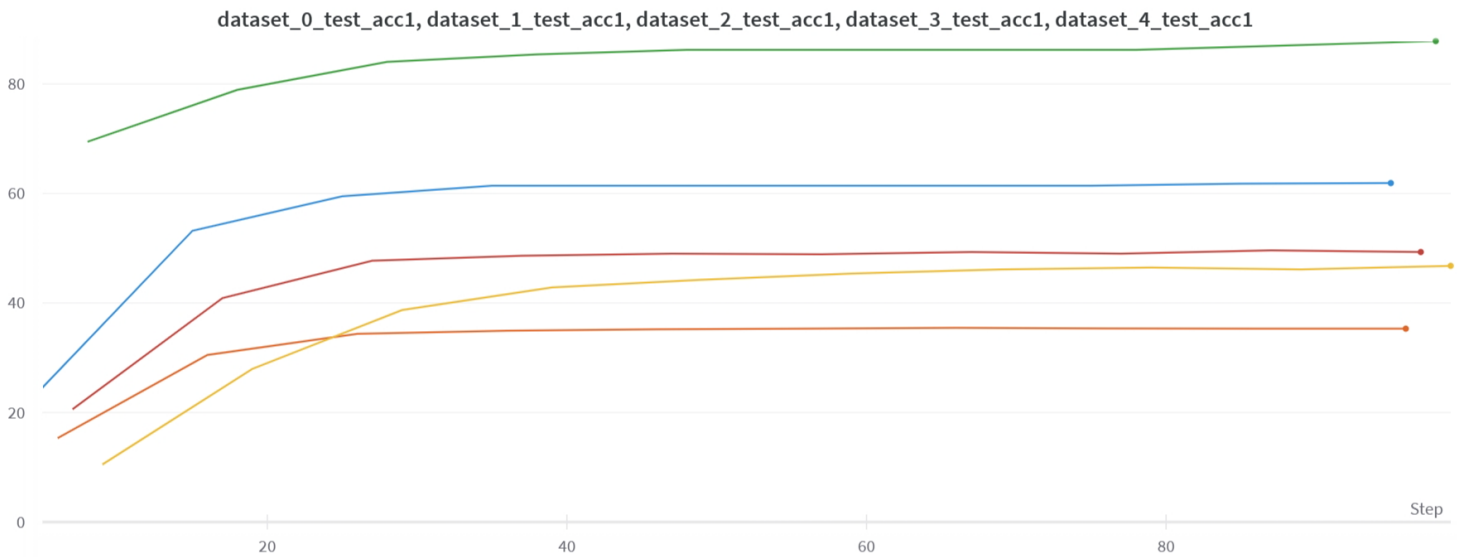



图 9: gcvit

# 结论

对于模型，在控制模型规模（参数量）大小相似的情况下:

- 在 few-shot-learning 的任务中，纯卷积 <vit< 混合 vit
- 混合 vit
  - 卷积的 inductive bias 优势
  - 更强的特征提取能力
  - 不一定要在大数据集上才能 perform well

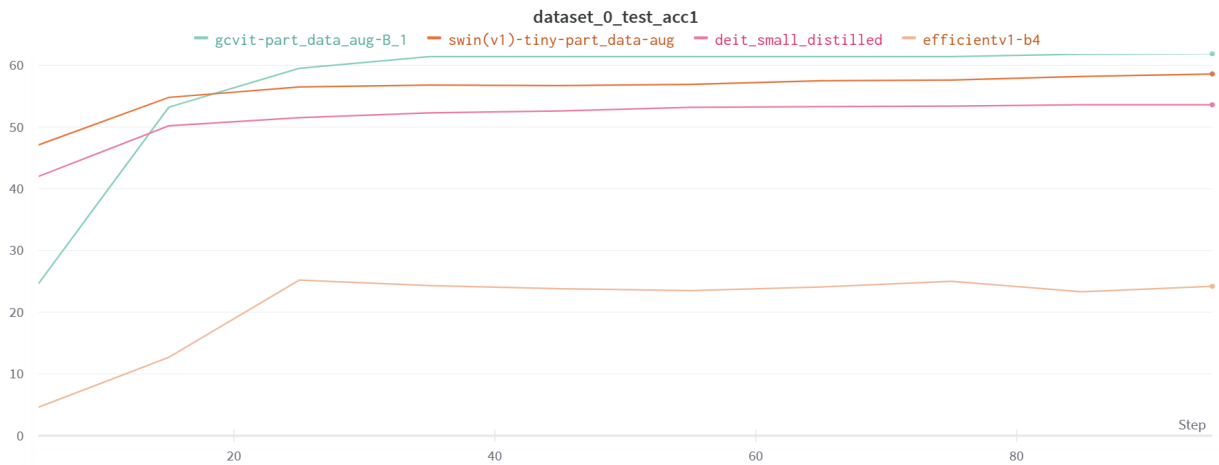



图 10: dataset0 test accuracy

# 结论
## 控制变量

1. 参数规模
   - EfficientNet b5:30M
   - Deit_small_distilled:22M
   - swin_tiny:29M
   - gcvit_tiny:28M

**Table 1** – Image classification benchmarks on **ImageNet-1K** dataset (Deng et al., 2009).

| Method | Param (M) | FLOPs (G) | Image Size | Top-1 (%) |
|---|---|---|---|---|
| **EfficientNet-B5** | **83.6%** | **96.7%** | **30M** | **1x** | **9.9B** | **1x** |
| DeiT-Small/16 (Touvron et al., 2021b) | 22 | 4.6 | $224^2$ | 79.9 |
| Swin-T (Liu et al., 2021) | 29 | 4.5 | $224^2$ | 81.3 |
| **GC ViT-T** | 28 | 4.7 | $224^2$ | **83.4** |

2. 数据增强方式:randaug*2
3. lr_schedule:cosine
4. batch_size:64

# 如何利用大量的 unlabel data?

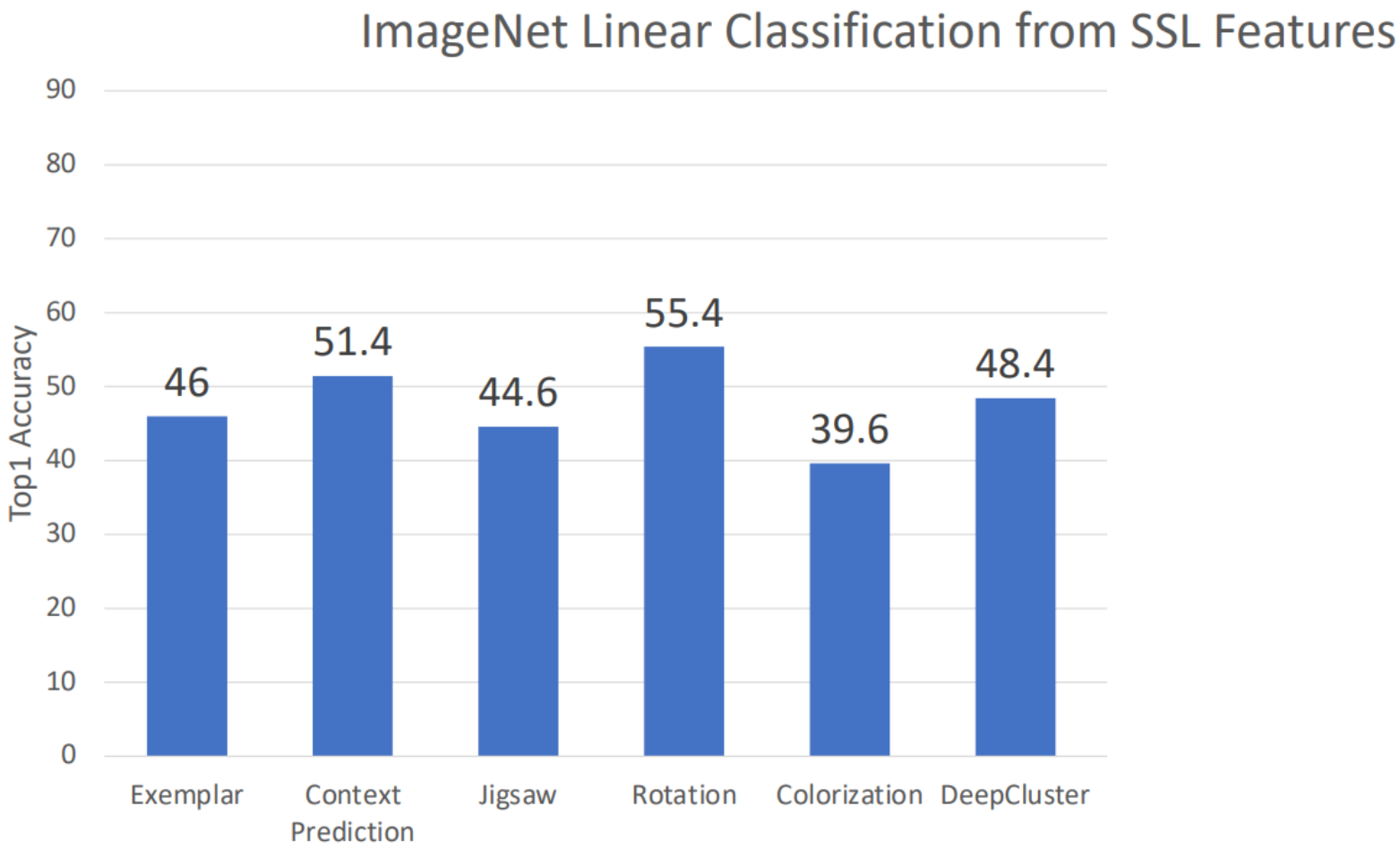
## RotNet: Predict Rotation

4-way classification task: How much was each image rotated? (0, 90, 180, or 270 degrees)

90 270 180 0 270

Gidaris et al, "Unsupervised representation learning by predicting image rotations", ICLR 2018

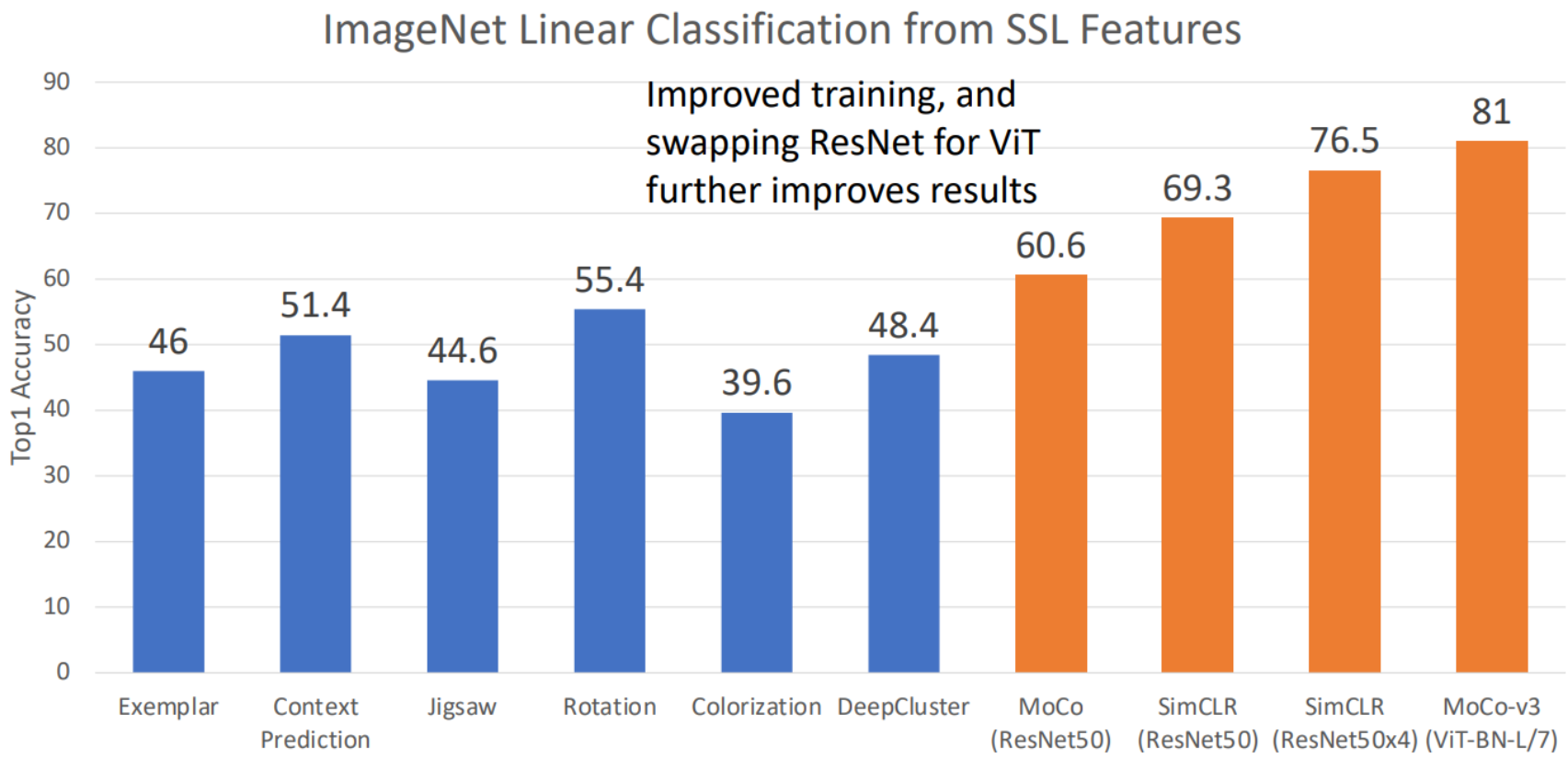
# 如何利用大量的 unlabel data?




He et al, "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020
Chen et al, "A Simple Framework for Contrastive Learning of Visual Representations", ICML 2020
Chen et al, "An Empirical Study of Training Self-Supervised Vision Transformers", ICCV 2021

(Lots of caveats here ... different architect

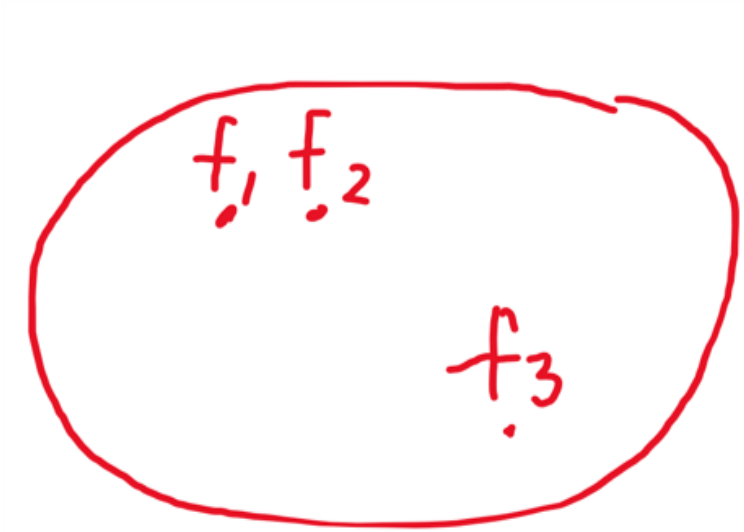
# 如何利用大量的 unlabel data?



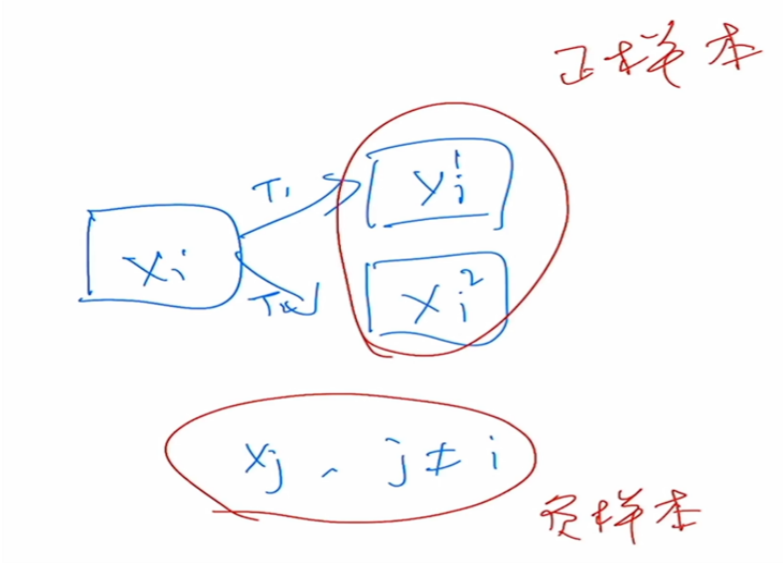

He et al, "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020
Chen et al, "A Simple Framework for Contrastive Learning of Visual Representations", ICML 2020
Chen et al, "An Empirical Study of Training Self-Supervised Vision Transformers", ICCV 2021

(Lots of caveats here ... different architectures, etc)

# Background of Contrastive Learning

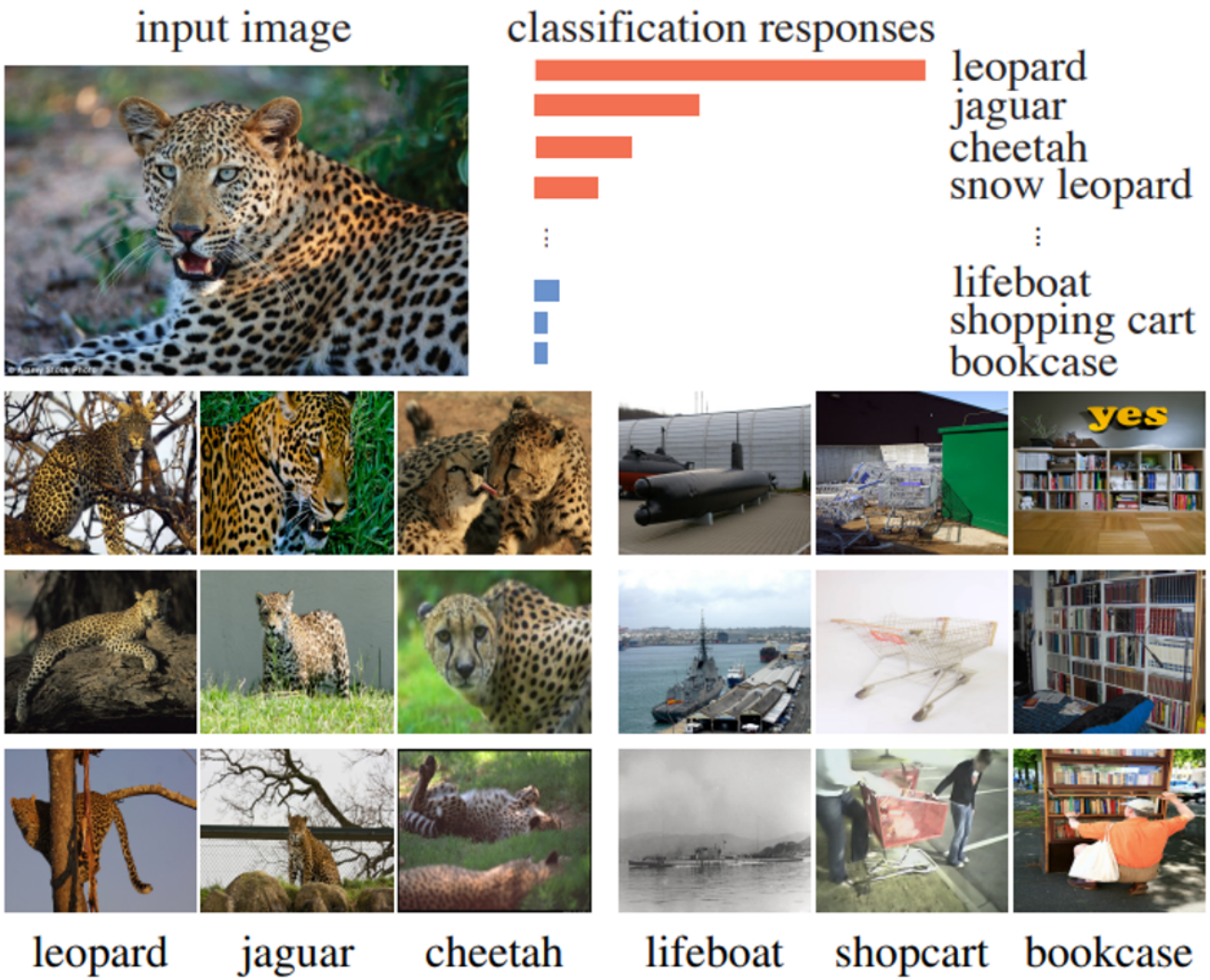



(a) What's contrast learning?




(b) How to get "labels"?

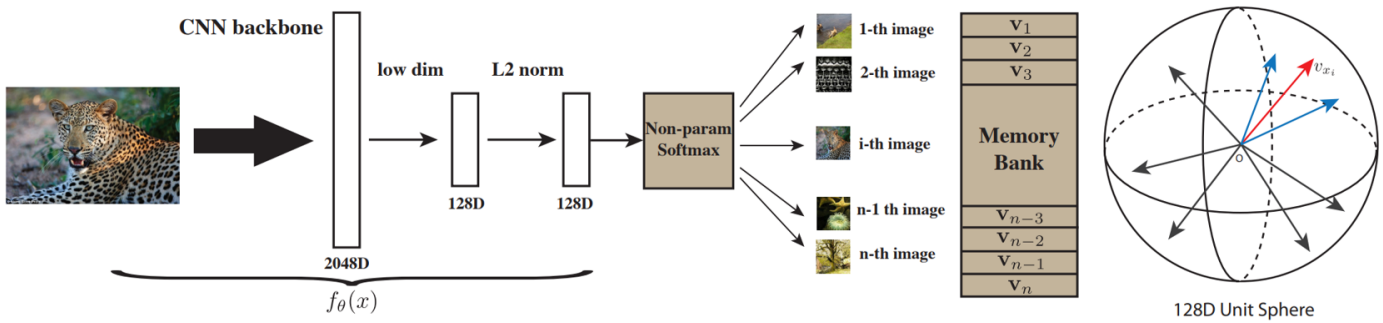
# Background of Constrastive Learning



# Background of Contrastive Learning

## instance-discrimination

- 每一个图片 (instance) 本身就是一个" 类". 正样本就是每张图片本身, 负样本就是所有其他图片 (均存放于 memory bank 中, 每次随机抽取一些负样本来和正样本一起训练). 每张图片所采用的特征维度为 128 维
- 在 NCELoss 的基础上去算目标函数; 每一个不同的 iteration, 都用新得到的每张图片的特征 tensor 来更新存放于 memory bank 中的" 负样本".



# Background of Contrastive Learning

## invariant spreading

- 同一个 instance 在不同 data augmentation 之下生成的图像, features 应该尽可能相似; 不同的 instance, features 应该尽可能不同.
- 在同一个 minibatch 中生成负样本, 这样就能只用 1 个编码器实现 end-to-end 训练.

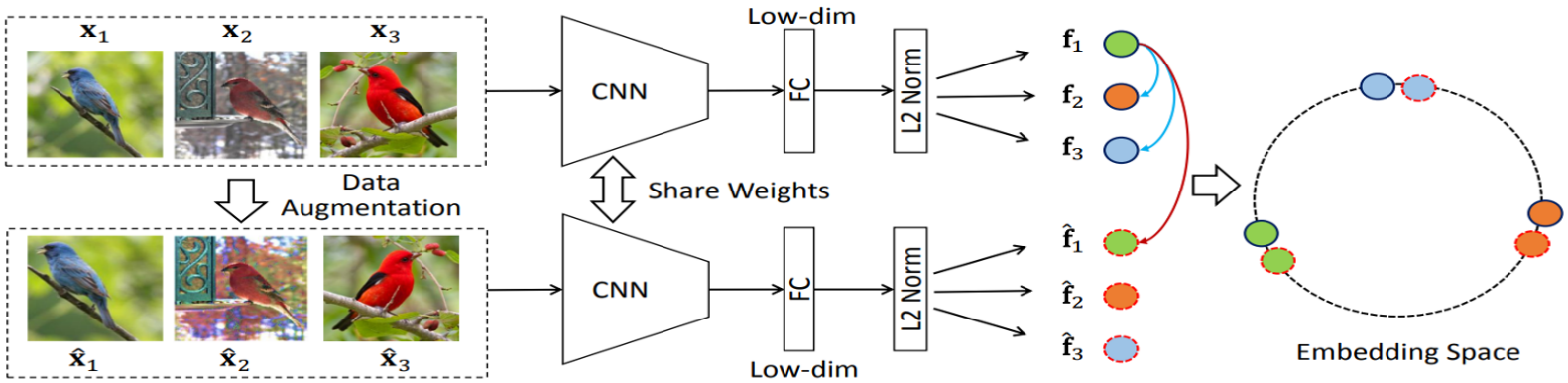



Figure 2: The framework of the proposed unsupervised learning method with Siamese network. The input images are projected into low-dimensional normalized embedding features with the CNN backbone. Image features of the same image instance with different data augmentations are invariant, while embedding features of different image instances are spread-out.

# Momentum Contrast for Unsupervised Visual Representation Learning

## Momentum Contrast for Unsupervised Visual Representation Learning

Kaiming He Haoqi Fan Yuxin Wu Saining Xie Ross Girshick

Facebook AI Research (FAIR)

Code: https://github.com/facebookresearch/moco

We present *Momentum Contrast* (MoCo) as a way of building large and consistent dictionaries for unsupervised learning with a contrastive loss (Figure 1). We maintain the dictionary as a *queue* of data samples: the encoded representations of the current mini-batch are enqueued, and the oldest are dequeued. The queue decouples the dictionary size from the mini-batch size, allowing it to be large. Moreover, as the dictionary keys come from the preceding several mini-batches, a *slowly progressing* key encoder, implemented as a momentum-based moving average of the query encoder, is proposed to maintain consistency.

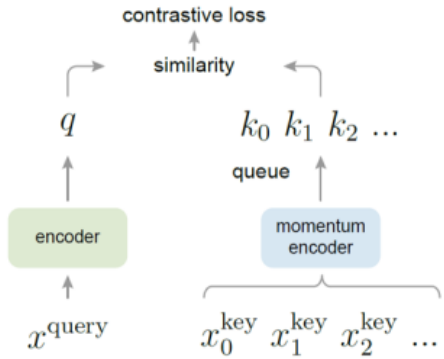



Figure 1. Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query $q$ to a dictionary of encoded keys using a contrastive loss. The dictionary keys $\{k_0, k_1, k_2, ...\}$ are defined on-the-fly by a set of data samples.

# Momentum Contrast for Unsupervised Visual Representation Learning

Contrast learning as **dictionary look-up**

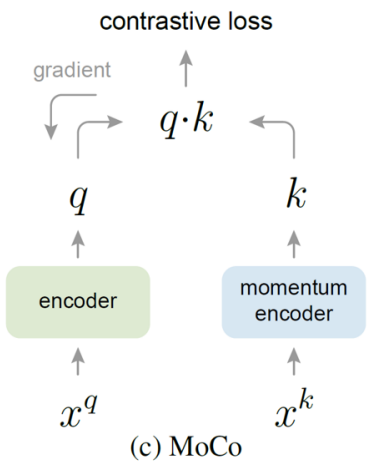
- An encoded query $q$ and a set of samples $\{k_0, k_1, k_2, ...\}$
- A contrastive loss is a function whose value is low when q is similar to its positive key $k_+$ and dissimilar to all other keys (negative keys).

$$InfoNCE : \mathcal{L}_q = -log\frac{exp(q \cdot k_+/\tau)}{\sum_{i=0}^{K} exp(q \cdot k_i/\tau)}$$

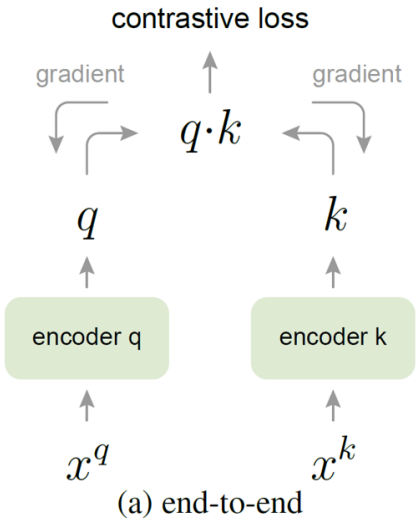
# Strength 1: Dictionary as a queue

优于 end-to-end

1. 负样本数量与 batch_size 解耦合，显著提高每一次比较量的同时，又不会增大 batch_size 负担，减少显存消耗（当时硬件所能支撑的最大 batch-size=1024）；
2. batch_size 过大会导致 optimizer 更难 train，得用 adam 甚至更高级的 optimizer




(c)

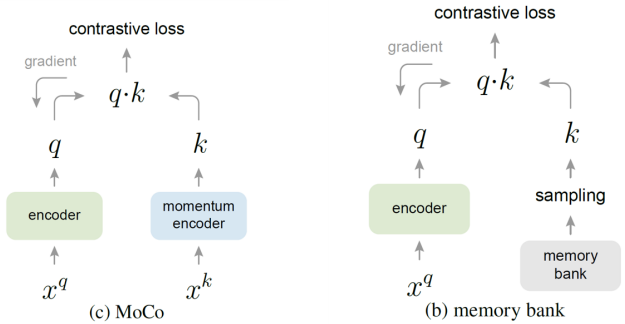



(d)

# Strength 2: Momentum update

优于 memory bank

1. momentum updated key encoder 使得当前处在字典中的 keys(负样本) 都是最新的: 是与当前要比较的 query（正样本）之间 consistency 更加高的;
2. Why not BP? Too large dictionary! Gradient should propagate to all samples in the queue
3. Why not copy the key encoder $f_k$ from the query encoder $f_q$? Poor experiment results! Rapidly changing encoder reduces the key representations' consistency.




(e) (f)

# 实验结果-Ablations

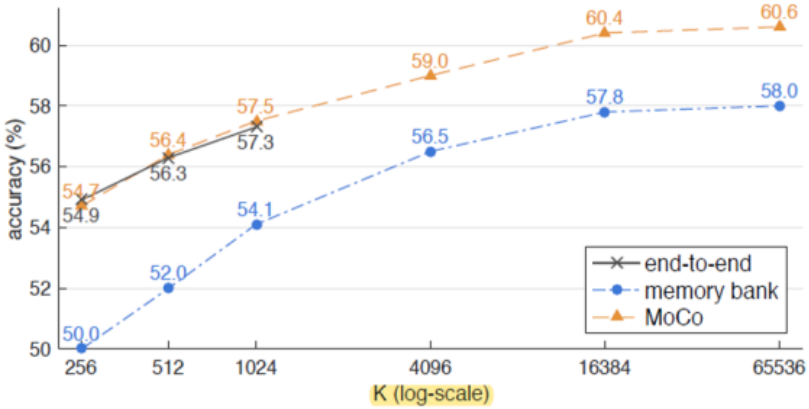


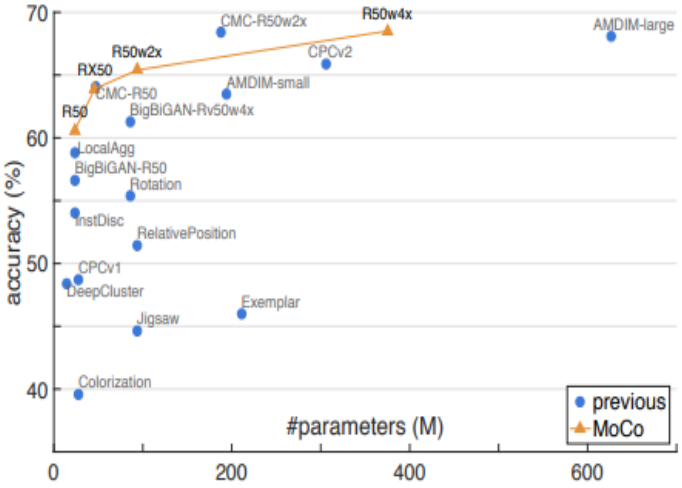

Figure 3. **Comparison of three contrastive loss mechanisms** under the ImageNet linear classification protocol. We adopt the same pretext task (Sec. 3.3) and only vary the contrastive loss mechanism (Figure 2). The number of negatives is $K$ in memory bank and MoCo, and is $K-1$ in end-to-end (offset by one because the positive key is in the same mini-batch). The network is ResNet-50.

**Ablation: momentum.** The table below shows ResNet-50 accuracy with different MoCo momentum values ($m$ in Eqn.(2)) used in pre-training ($K = 4096$ here) :

| momentum $m$ | 0 | 0.9 | 0.99 | 0.999 | 0.9999 |
|---|---|---|---|---|---|
| accuracy (%) | *fail* | 55.2 | 57.8 | 59.0 | 58.9 |

It performs reasonably well when $m$ is in 0.99 ~ 0.9999, showing that a slowly progressing (*i.e.*, relatively large momentum) key encoder is beneficial. When $m$ is too small (*e.g.*, 0.9), the accuracy drops considerably; at the extreme of *no momentum* ($m$ is 0), the training loss oscillates and fails to converge. These results support our motivation of building a consistent dictionary.

# 实验结果-Classification




(g)

| method | architecture | #params (M) | accuracy (%) |
|---|---|---|---|
| Exemplar [17] | R50w3× | 211 | 46.0 [38] |
| RelativePosition [13] | R50w2× | 94 | 51.4 [38] |
| Jigsaw [45] | R50w2× | 94 | 44.6 [38] |
| Rotation [19] | Rv50w4× | 86 | 55.4 [38] |
| Colorization [64] | R101* | 28 | 39.6 [14] |
| DeepCluster [3] | VGG [53] | 15 | 48.4 [4] |
| BigBiGAN [16] | R50 | 24 | 56.6 |
| | Rv50w4× | 86 | 61.3 |
| *methods based on contrastive learning follow:* | | | |
| InstDisc [61] | R50 | 24 | 54.0 |
| LocalAgg [66] | R50 | 24 | 58.8 |
| CPC v1 [46] | R101* | 28 | 48.7 |
| CPC v2 [35] | $R170^{*}_{wider}$ | 303 | 65.9 |
| CMC [56] | $R50_{L+ab}$ | 47 | 64.1† |
| | $R50w2\times_{L+ab}$ | 188 | 68.4† |
| AMDIM [2] | $AMDIM_{small}$ | 194 | 63.5† |
| | $AMDIM_{large}$ | 626 | 68.1† |
| **MoCo** | R50 | 24 | 60.6 |
| | RX50 | 46 | 63.9 |
| | R50w2× | 94 | 65.4 |
| | R50w4× | 375 | **68.6** |

(h)

图 11: Comparison under the linear classification protocol on ImageNet

# 实验结果-MoCov2

| case | unsup. pre-train MLP | aug+ | cos | epochs | batch | ImageNet acc. |
|---|---|---|---|---|---|---|
| MoCo v1 [6] | | | | 200 | 256 | 60.6 |
| SimCLR [2] | ✓ | ✓ | ✓ | 200 | 256 | 61.9 |
| SimCLR [2] | ✓ | ✓ | ✓ | 200 | 8192 | 66.6 |
| **MoCo v2** | ✓ | ✓ | ✓ | 200 | 256 | **67.5** |
| *results of **longer** unsupervised training follow:* | | | | | | |
| SimCLR [2] | ✓ | ✓ | ✓ | 1000 | 4096 | 69.3 |
| **MoCo v2** | ✓ | ✓ | ✓ | 800 | 256 | **71.1** |

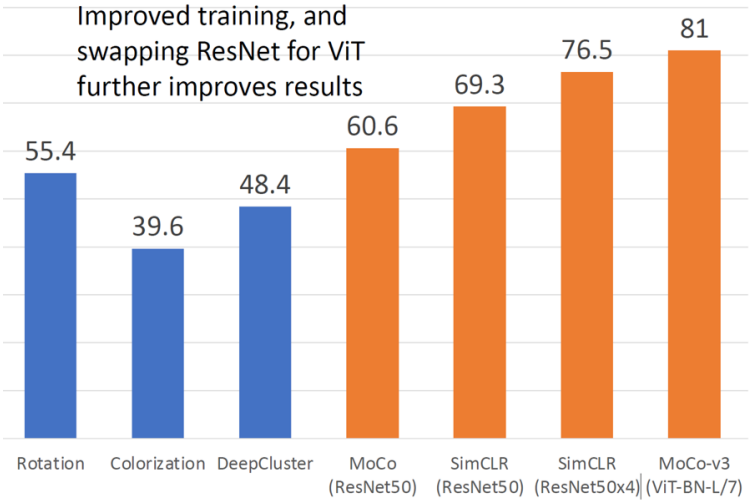
Table 2. **MoCo *vs.* SimCLR**: ImageNet linear classifier accuracy (**ResNet-50, 1-crop 224×224**), trained on features from unsuper-

| mechanism | batch | memory / GPU | time / 200-ep. |
|---|---|---|---|
| MoCo | 256 | **5.0G** | **53 hrs** |
| end-to-end | 256 | 7.4G | 65 hrs |
| end-to-end | 4096 | 93.0G† | n/a |

Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. †: based on our estimation.

# 总结与讨论

- A general model for contrastive learning!
- Moco V3-just replace the backbone with ViT




(a)

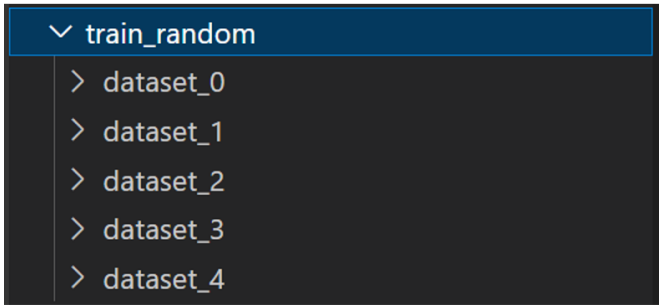
This paper's focus is on a mechanism for general contrastive learning; we do not explore orthogonal factors (such as specific pretext tasks) that may further improve accuracy. As an example, "MoCo v2" [8], an extension of a preliminary version of this manuscript, achieves 71.1% accuracy with R50 (up from 60.6%), given small changes on the data augmentation and output projection head [7]. We believe that this additional result shows the generality and robustness of the MoCo framework.

(b)

# 我们自己的 Moco-V2 实现

1、Unlabeled Dataset

- 训练速度缘故，故根据各数据集类数比例，sample 了 10000 + 21100 +10100 + 3310 + 6184 张图片，共约 50000 张；
- 考虑到 10-shot 数据集不像 imagenet 各类分布均匀，所以这样 sample，或许能有缓解数据长尾分布的作用；



# 我们自己的 Moco-V2 实现

## 2、Backbone

- 采用和 moco v2 一样的 resnet50(25M 参数量，与上面所实验的几个 vit 相似)，并且对于 backbone 也导入了 imagenet-pretrained，以提高模型初始能力，希望能学到更好的 features;

```
34
35          self.encoder_q = base_encoder(weights=ResNet50_Weights.IMAGENET1K_V1)
36          self.encoder_k = base_encoder(weights=ResNet50_Weights.IMAGENET1K_V1)
37          dim_mlp = self.encoder_q.fc.weight.shape[1]
38
39          if mlp:  # hack: brute-force replacement
40              self.encoder_q.fc = nn.Sequential(
41                  nn.Linear(dim_mlp, dim_mlp), nn.ReLU(), nn.Linear(dim_mlp, dim)
42              )
43              self.encoder_k.fc = nn.Sequential(
44                  nn.Linear(dim_mlp, dim_mlp), nn.ReLU(), nn.Linear(dim_mlp, dim)
45              )
46
```

# 我们自己的 Moco-V2 实现

3、Hyperparameters

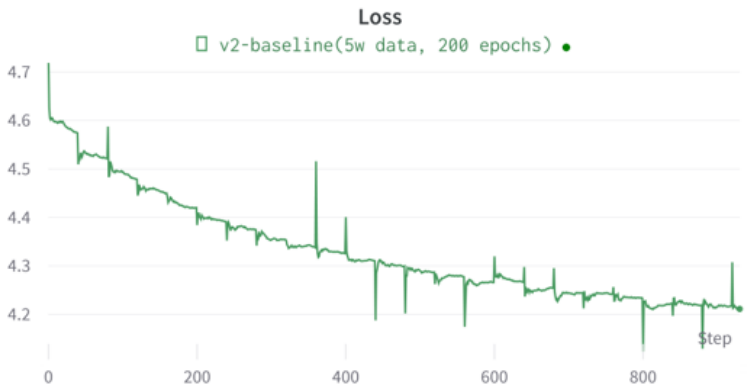
- 原论文显存消耗: 8*5G; 256 batch_size; lr=0.03; 我们计算资源有限, 因此 batch_size=128, num_keys=4096; lr=0.015; 比例合适; 其余均与 mocov2 的配置一样

AICourse-Few-Shot-Learning > moco > $ ddp.sh

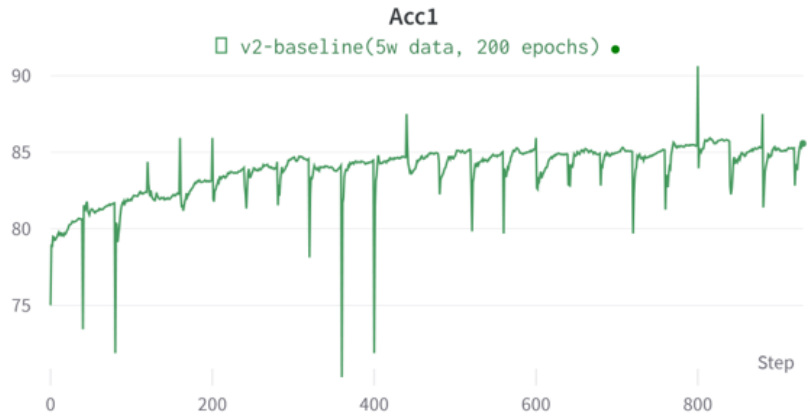
You, 3 hours ago | 1 author (You)

```
python main_moco.py \
  -a resnet50 \
  --lr 0.015 --epochs 200 \
  --batch-size 128 --moco-k 4096 \
  --dist-url 'tcp://localhost:10001' --multiprocessing-distributed --world-size 1 --rank 0 \
  --resume 'checkpoint_0004.pth.tar' \
  --mlp --moco-t 0.2 --aug-plus --cos \
```

# 训练结果




(c) Loss




(d) Acc

[1] xinlei Chen. “Improved baselines with momentum contrastive learning”. In: *arXiv preprint arXiv:2003.04297* (2020).

[2] Kaiming He et al. “Momentum Contrast for Unsupervised Visual Representation Learning”. In: *arXiv preprint arXiv:1911.05722* (2019).

[3] Zhirong Wu. “Unsupervised feature learning via non-parametric instance discrimination”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018).

[4] Mang Ye. “Unsupervised embedding learning via invariant and spreading instance feature”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019).

# Thanks!