第二周 工作汇报

王宇

目 录

CONTENTS

01 本周工作进展

本　　周　　工　　作　　进　　展

根据上周讨论情况，本周预计的工作计划以及完成情况

- ✓ 学习t-SNE算法原理以及代码实现。尝试复现原论文中插图
- ✓ 落实原论文对比学习中，如何处理neg-positive imbalance 问题的方法
- ✓ 阅读Grad-CAM原论文，以及相关代码实现
- ☐ 阅读Semi-Supervised Semantic Segmentation with Cross Pseudo Supervision (CVPR2021)
- ✓ 学习对比学习相关内容
- ✓ 统计学习方法(刚开始，还没有找到很好的学习方法)

本　周　工　作　进　展

✓ t-SNE 原理和代码实现

```
output_l = (Tensor: (4, 21, 320, 320)) tensor([[[[ 9.6662e+00,  9.7369e+0
```

```
tensor([[[[ 9.6662e+00,  9.7369e+00,  9.8075e+00,  ...,  9.1976e+00,
            8.9831e+00,  8.7685e+00],
          [ 1.0194e+01,  1.0260e+01,  1.0327e+01,  ...,  9.7720e+00,
            9.5306e+00,  9.2892e+00],
          [ 1.0721e+01,  1.0784e+01,  1.0846e+01,  ...,  1.0346e+01,
            1.0078e+01,  9.8098e+00],
          ...,
          [ 6.8022e+00,  7.1702e+00,  7.5381e+00,  ...,  9.6948e+00,
            9.3631e+00,  9.0313e+00],
          [ 6.5027e+00,  6.8879e+00,  7.2731e+00,  ...,  9.2898e+00,
            8.9087e+00,  8.5277e+00],
          [ 6.2032e+00,  6.6056e+00,  7.0081e+00,  ...,  8.8847e+00,
            8.4544e+00,  8.0240e+00]],

         [[-1.6227e+00, -1.6195e+00, -1.6162e+00,  ..., -5.0579e-01,
           -4.8892e-01, -4.7205e-01],
          [-1.7263e+00, -1.6190e+00, -1.5117e+00,  ..., -7.0250e-01,
           -6.4751e-01, -5.9252e-01],
          [-1.8298e+00, -1.6185e+00, -1.4071e+00,  ..., -8.9922e-01,
           -8.0610e-01, -7.1299e-01],
          ...,
          [-1.8357e+00, -1.9028e+00, -1.9699e+00,  ..., -1.5047e+00,
           -1.5140e+00, -1.5233e+00],
          [-1.8626e+00, -1.9558e+00, -2.0490e+00,  ..., -1.6917e+00,
```

Figure 3. Visual comparison between *contextual augmentation* (I and II) and *low-level augmentation* (I and III) using t-SNE visualization for features of the overlapping region (shown in yellow box). **Top**: input crops from the same image, where II and III apply the *contextual* and *low-level augmentation* respectively. **Middle**: t-SNE results of the model trained with labeled data only. Note that the three visualizations are in the same t-SNE space, and the dots with the same color represent the features of the same class. **Bottom**: t-SNE results of our method.

本　周　工　作　进　展

✓ t-SNE 原理和代码实现

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import manifold
'''
    tSNE对数据进行t-sne可视化
    @para:
    X: shape（num,dim) num为数据点的个数，dim是每个数据点的特征的维度  eg:（100，3）表示一共有100个数据点，每个数据点是3维向量
    label: shape(num,1) 对应每个数据点的类别标签
    colorlist: dict:{"red":red}每个类别对应的颜色
    n_components:数据点降维之后的维度，默认是2
    init:和库函数的含义一样，可以选择"pca: "random"
    perplexity:建议5-50，数据量越大，该值应该设置的越大
'''
def tSNE(X,label,colorlist,n_components=2,init="pca", perplexity=10):
    #降维
    tsne = manifold.TSNE(n_components=n_components,init=init,random_state=0, perplexity=10)
    X_tsne = tsne.fit_transform(X)#(187500, 2)
    #可视化
    for color in colorlist.keys():
        plt.scatter(X_tsne[colorlist[color][:,0]][:,0], X_tsne[colorlist[color][:,0]][:,1],s=0.01, c=color)
    plt.show()
```

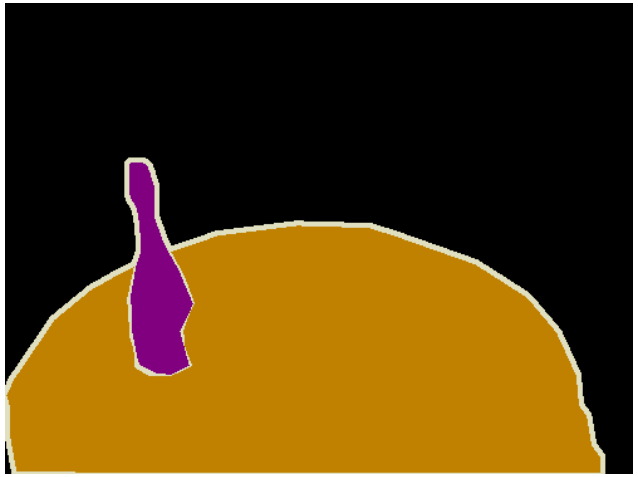
本　周　工　作　进　展

- ✓ t-SNE 原理和代码实现
  函数调用

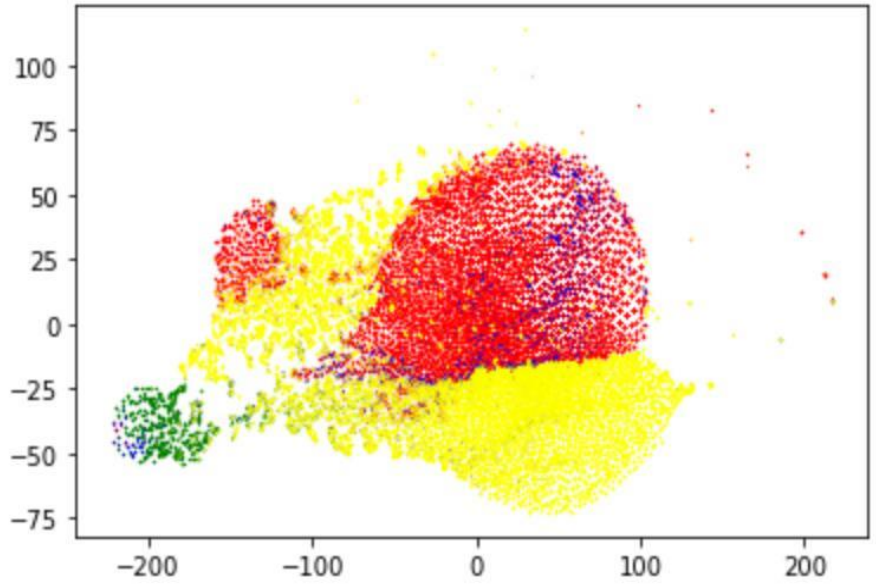
```
from PIL import Image
import numpy as np
#加载图片和标签
X= np.asarray(Image.open("D:/Users/July/Desktop/image1.jpg")) #(366, 500,3)
label = np.asarray(Image.open("D:/Users/July/Desktop/label1.png")) #(366,500)

#调整尺寸
label = label.reshape(-1,1)
X=X.reshape(-1,3)

#计算colorlist
red= label==0
green= label ==5
blue = label ==255
yellow = label ==11
colorlist={}
colorlist['red']=red
colorlist['green']=green
colorlist['blue']=blue
colorlist['yellow']=yellow
#调用函数
tSNE(X,label,colorlist)
```

本　周　工　作　进　展

✓ t-SNE 原理和代码实现



```
In [14]: tSNE(X,label,colorlist)
```



本　周　工　作　进　展

✓ t-SNE 原理和代码实现

output_l = (Tensor: (4, 21, 320, 320)) tensor([[[[9.6662e+00, 9.7369e+00, 9.807

```
tensor([[[[ 9.6662e+00,  9.7369e+00,  9.8075e+00,  ...,  9.1976e+00,
            8.9831e+00,  8.7685e+00],
          [ 1.0194e+01,  1.0260e+01,  1.0327e+01,  ...,  9.7720e+00,
            9.5306e+00,  9.2892e+00],
          [ 1.0721e+01,  1.0784e+01,  1.0846e+01,  ...,  1.0346e+01,
            1.0078e+01,  9.8098e+00],
          ...,
          [ 6.8022e+00,  7.1702e+00,  7.5381e+00,  ...,  9.6948e+00,
            9.3631e+00,  9.0313e+00],
          [ 6.5027e+00,  6.8879e+00,  7.2731e+00,  ...,  9.2898e+00,
            8.9087e+00,  8.5277e+00],
          [ 6.2032e+00,  6.6056e+00,  7.0081e+00,  ...,  8.8847e+00,
            8.4544e+00,  8.0240e+00]],

         [[-1.6227e+00, -1.6195e+00, -1.6162e+00,  ..., -5.0579e-01,
           -4.8892e-01, -4.7205e-01],
          [-1.7263e+00, -1.6190e+00, -1.5117e+00,  ..., -7.0250e-01,
           -6.4751e-01, -5.9252e-01],
          [-1.8298e+00, -1.6185e+00, -1.4071e+00,  ..., -8.9922e-01,
           -8.0610e-01, -7.1299e-01],
          ...,
          [-1.8357e+00, -1.9028e+00, -1.9699e+00,  ..., -1.5047e+00,
           -1.5140e+00, -1.5233e+00],
          [-1.8626e+00, -1.9558e+00, -2.0490e+00,  ..., -1.6917e+00,
```

I. Original　　II. Contextual aug　　III. Low-level aug

Figure 3. Visual comparison between *contextual augmentation* (I and II) and *low-level augmentation* (I and III) using t-SNE visualization for features of the overlapping region (shown in yellow box). **Top**: input crops from the same image, where II and III apply the *contextual* and *low-level augmentation* respectively. **Middle**: t-SNE results of the model trained with labeled data only. Note that the three visualizations are in the same t-SNE space, and the dots with the same color represent the features of the same class. **Bottom**: t-SNE results of our method.

本　周　工　作　进　展

- ✓ 如何处理neg-positive imbalance 问题的方法

  通过超参数设置positive pair 和 negative pair的个数

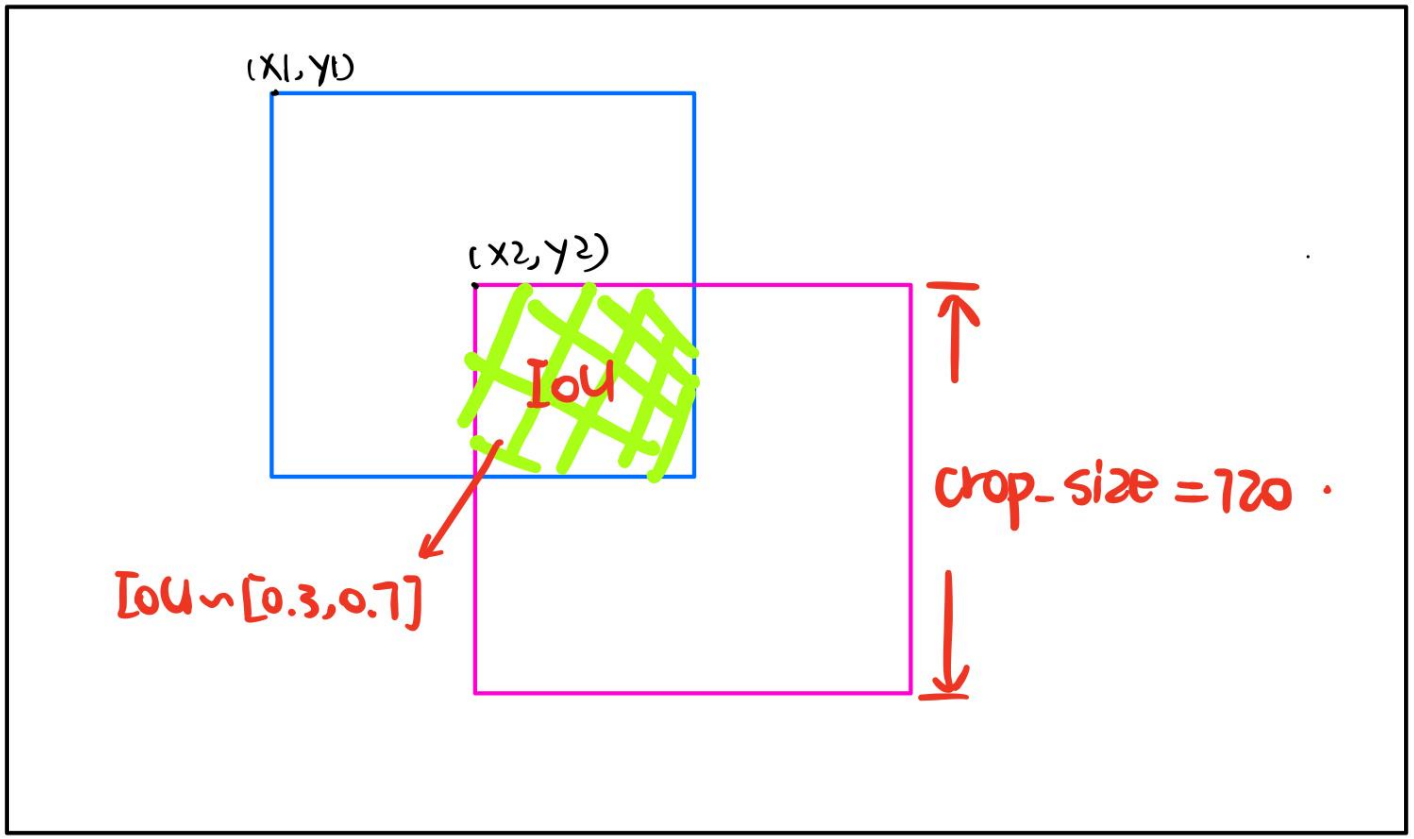
selected_num：规定了negative pair的个数

```
#  "selected_num": 6400
selected_num = self.selected_num
#permute ：将维度进行换位调整，permute之后tensor在内存中不再是连续存储的，而view要求tesor是连续存储的，所以需要contiguous来返回一个contiguous cop
#output_ul1是pojector的输出
output_ul1_flatten = output_ul1.permute(0, 2, 3, 1).contiguous().view(b*h*w, c) #(6400,128)
output_ul2_flatten = output_ul2.permute(0, 2, 3, 1).contiguous().view(b*h*w, c)

#随机选择6400个像素点
selected_idx1 = np.random.choice(range(b*h*w), selected_num, replace=False) #6400
selected_idx2 = np.random.choice(range(b*h*w), selected_num, replace=False)
output_ul1_flatten_selected = output_ul1_flatten[selected_idx1]  #(6400,128)
output_ul2_flatten_selected = output_ul2_flatten[selected_idx2]

#将随机选择的两个子图之间的像素点的feature_map进行配对
output_ul_flatten_selected = torch.cat([output_ul1_flatten_selected, output_ul2_flatten_selected], 0) #[2*kk, c]  (12800,128)
output_ul_all = self.concat_all_gather(output_ul_flatten_selected) #[2*N, c]   (25600,128)

#将像素点对应的label进行配对
pseudo_label1_flatten_selected = pseudo_label1.view(-1)[selected_idx1]  #(6400,)
pseudo_label2_flatten_selected = pseudo_label2.view(-1)[selected_idx2]
pseudo_label_flatten_selected = torch.cat([pseudo_label1_flatten_selected, pseudo_label2_flatten_selected], 0) #[2*kk]
pseudo_label_all = self.concat_all_gather(pseudo_label_flatten_selected) #[2*N]
```

本　周　工　作　进　展

- ✓ 如何处理neg-positive imbalance 问题的方法

  通过超参数设置positive pair 和 negative pair的个数



本　　周　　工　　作　　进　　展

- ✓ 如何处理neg-positive imbalance 问题的方法

  这里 IoU_bound, crop_size , selected_num 都是可以超参数,通过配置文件设置

最后negative pair的个数为:
IoU_bound[0]*crop_size*crop_size *selected_num*2*batch_size

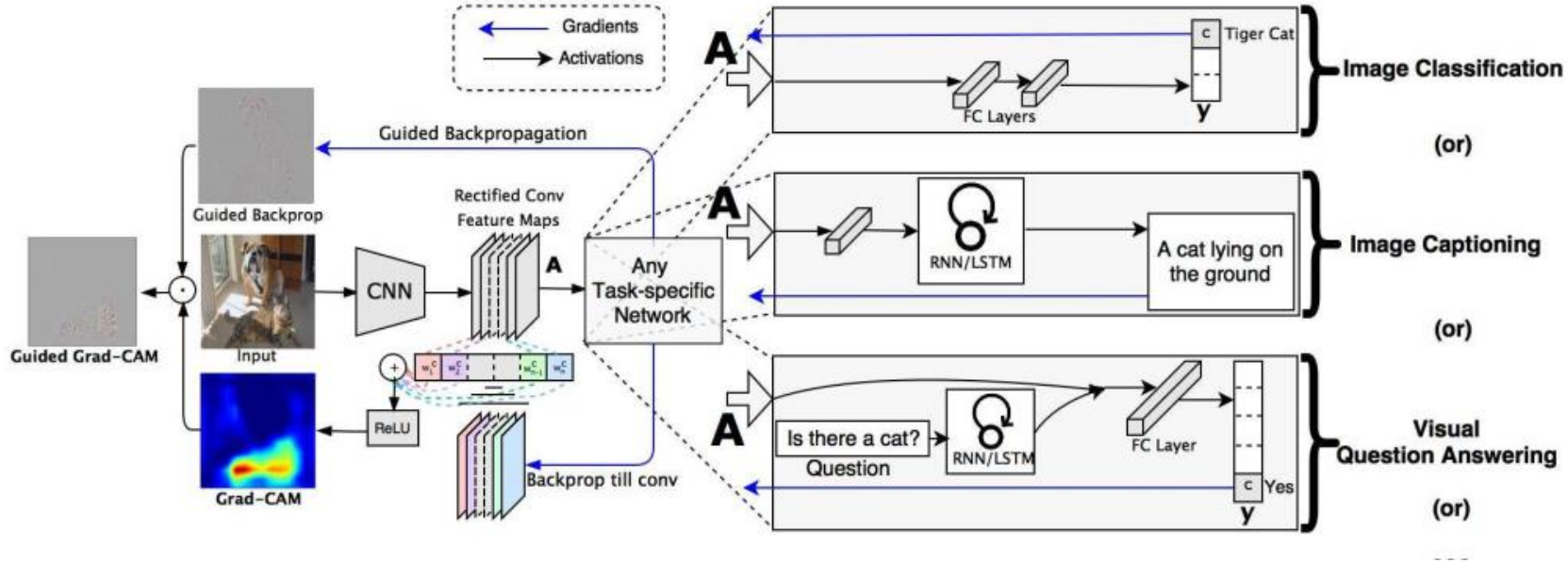
Positive pair的个数为重合区域所包含的像素点:
[IoU_bound[0]*crop_size*crop_size , IoU_bound[1]*crop_size*crop_size]

在原论文中:　　positive pair的个数为 [155520,362880]
negative pair与positive pair的比例为 6400*2*8= 102400

(加上mask计算，可能远小于这个)

本　周　工　作　进　展

✓ Grad-CAM论文阅读和代码实现



$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

$$L^c_{\text{Grad-CAM}} = ReLU\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

本　周　工　作　进　展

✓ Grad-CAM论文阅读和代码实现

CNN可视化方法 https://github.com/jacobgil/pytorch-grad-cam

Method	What it does
GradCAM	Weight the 2D activations by the average gradient
GradCAM++	Like GradCAM but uses second order gradients
XGradCAM	Like GradCAM but scale the gradients by the normalized activations
AblationCAM	Zero out activations and measure how the output drops (this repository includes a fast batched implementation)
ScoreCAM	Perbutate the image by the scaled activations and measure how the output drops
EigenCAM	Takes the first principle component of the 2D Activations (no class discrimination, but seems to give great results)
EigenGradCAM	Like EigenCAM but with class discrimination: First principle component of Activations*Grad. Looks like GradCAM, but cleaner
LayerCAM	Spatially weight the activations by positive gradients. Works better especially in lower layers

使用方法:
pip install grad-cam

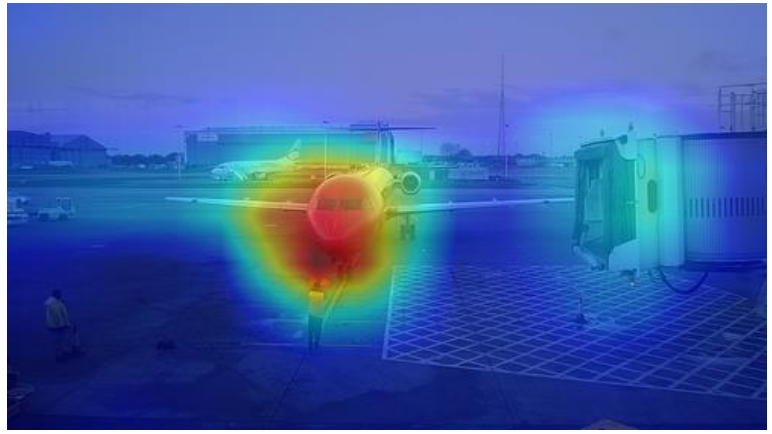
环境要求: (pytorch)
pip install tqdm
pip install ttach

本　周　工　作　进　展

✓ Grad-CAM论文阅读和代码实现

```python
from pytorch_grad_cam import GradCAM, ScoreCAM, GradCAMPlusPlus, AblationCAM, XGradCAM, EigenCAM
from pytorch_grad_cam.utils.image import show_cam_on_image
from torchvision.models import resnet50
import torchvision.transforms as transforms
import cv2 as cv
import numpy as np
```

```python
#加载函数模型
model = resnet50(pretrained=True)
#设置要测量那个网络层的效果
target_layer = model.layer4[-1]

#网络输入，转换成 Tensor格式，要符合网络的输入 (batch_size,c,w,h)
image = cv.imread("C:/Users/doris/Desktop/image.jpg").astype(np.float32)/255
transf= transforms.ToTensor()
input_tensor = transf(image).unsqueeze(0)

#加载GradCAM
cam = GradCAM(model=model, target_layer=target_layer, use_cuda=False)

#设置要关注的类别，如果不设置，默认返回得分最高的类别
# will be used for every image in the batch.
# target_category can also be an integer, or a list of different integers
# for every image in the batch.
target_category = 1

#计算网络可视化结果
# You can also pass aug_smooth=True and eigen_smooth=True, to apply smoothing.
grayscale_cam = cam(input_tensor=input_tensor, target_category=target_category)

# In this example grayscale_cam has only one image in the batch:
grayscale_cam = grayscale_cam[0, :]

#注意这里image 和 grayscale_cam都是narray.float32格式
visualization = show_cam_on_image(image, grayscale_cam)

#将结果写入到本土图片中
cv.imwrite("C:/Users/doris/Desktop/result.jpg",visualization)
```



本　周　工　作　进　展

✓ Grad-CAM论文阅读和代码实现

这里使用的是resnet50 avgpool的前一个卷积层
原文中应该是self.classfier[-1]

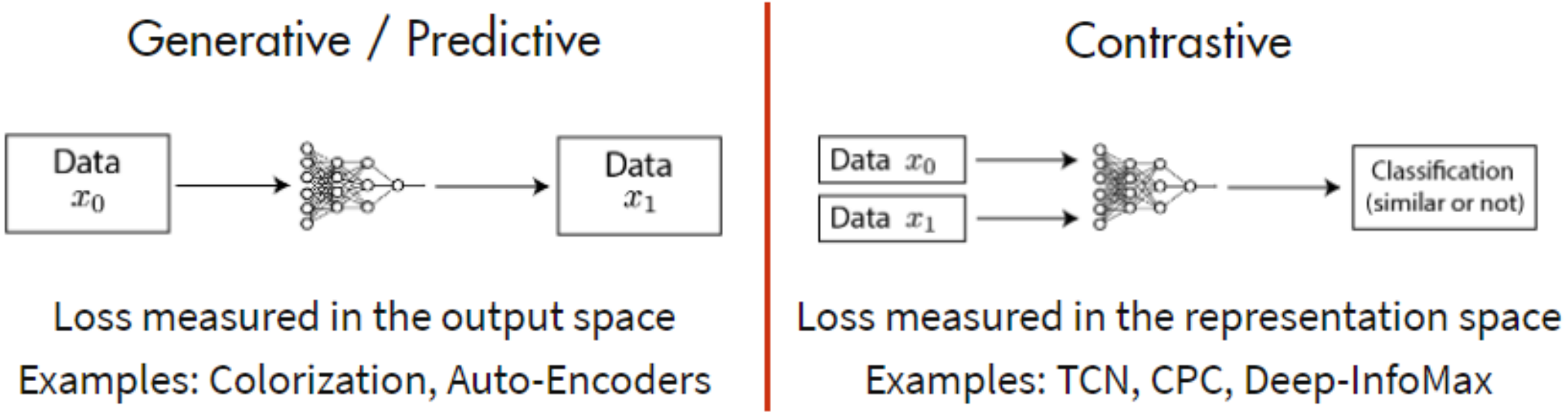
```
if self.backbone == 'deeplab_v3+':
    self.encoder = DeepLab_v3p(backbone='resnet{}'.format(self.layers))
    self.classifier = nn.Sequential(nn.Dropout(0.1), nn.Conv2d(256, num_classes, kernel_size=1, stride=1))
    for m in self.classifier.modules():
        if isinstance(m, nn.Conv2d):
            torch.nn.init.kaiming_normal_(m.weight)
        elif isinstance(m, nn.BatchNorm2d):
            m.weight.data.fill_(1)
            m.bias.data.zero_()
        elif isinstance(m, nn.SyncBatchNorm):
            m.weight.data.fill_(1)
            m.bias.data.zero_()
elif self.backbone == 'psp':
    self.encoder = Encoder(pretrained=pretrained)
    self.classifier = nn.Conv2d(self.out_dim, num_classes, kernel_size=1, stride=1)
else:
    raise ValueError("No such backbone {}".format(self.backbone))

if self.mode == 'semi':
    self.project = nn.Sequential(
        nn.Conv2d(self.out_dim, self.out_dim, kernel_size=1, stride=1),
        nn.ReLU(inplace=True),
        nn.Conv2d(self.out_dim, self.proj_final_dim, kernel_size=1, stride=1)
    )

    self.weight_unsup = conf['weight_unsup']
    self.temp = conf['temp']
    self.epoch_start_unsup = conf['epoch_start_unsup']
    self.selected_num = conf['selected_num']
    self.step_save = conf['step_save']
    self.step_count = 0
```

本　周　工　作　进　展

✓ 对比学习相关论文阅读

深度学习的本质：表示学习和归纳偏好学习

表示学习可以分为两类：Generative 和 Contrastive



本　周　工　作　进　展

✓ 对比学习相关论文阅读

对比学习的核心思想:
表示学习算法不一定要关注到样本的每个细节，只要学习到的特征能够使它和其它样本区分开就可以。

More formally, for any data point x, contrastive methods aim to learn an encoder f such that:

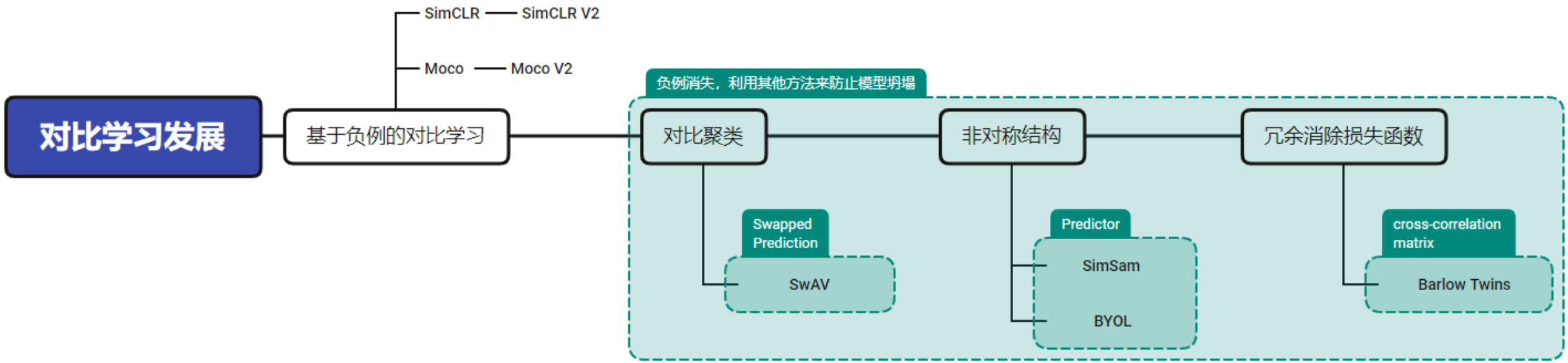
$$\text{score}(f(x), f(x^+)) >> \text{score}(f(x), f(x^-))$$

- here x^+ is data point similar or congruent to x, referred to as a *positive* sample.
- x^- is a data point dissimilar to x, referred to as a *negative* sample.
- the **score** function is a metric that measures the similarity between two features.

$$\mathcal{L}_N = -\mathbb{E}_X \left[\log \frac{\exp\left(f(x)^T f\left(x^+\right)\right)}{\exp\left(f(x)^T f\left(x^+\right)\right) + \sum_{j=1}^{N-1} \exp\left(f(x)^T f\left(x_j\right)\right)} \right]$$

本　周　工　作　进　展

✓ 对比学习相关论文阅读



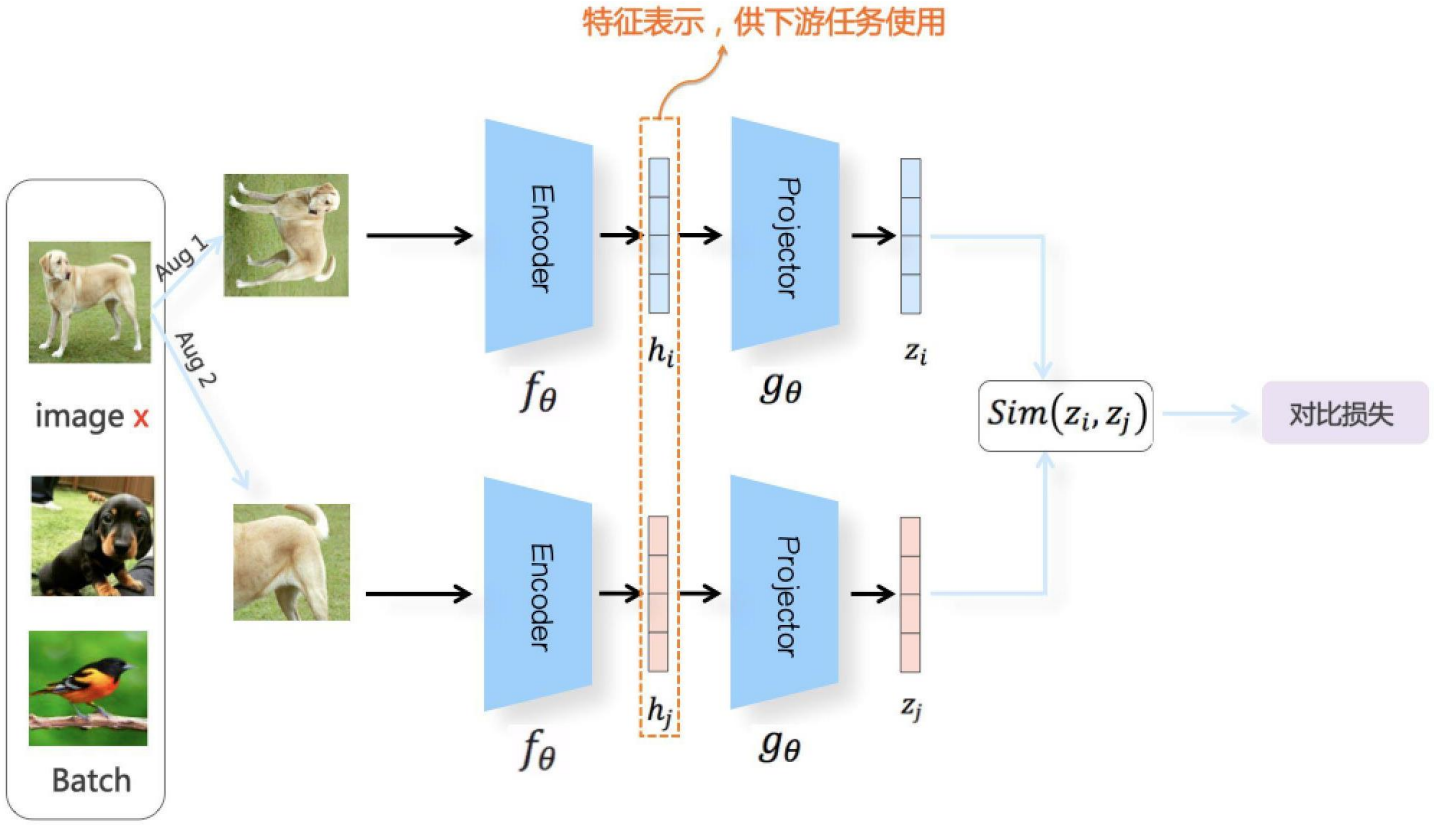
参考文献：

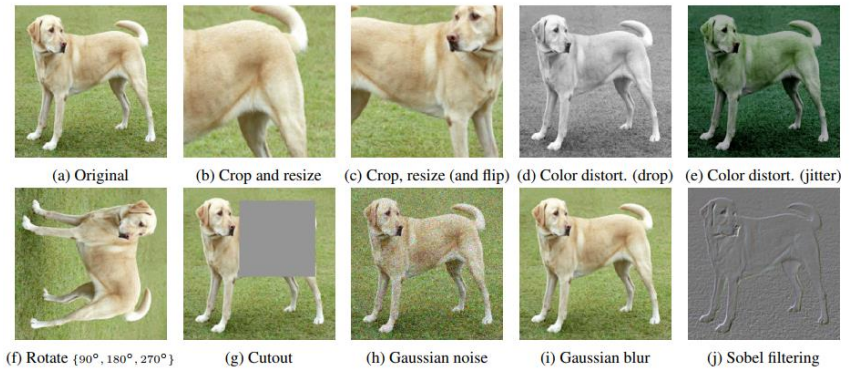
https://zhuanlan.zhihu.com/p/367290573;
https://ankeshanand.com/blog/2020/01/26/contrative-self-supervised-learning.html;
https://zhuanlan.zhihu.com/p/367290573;

本　周　工　作　进　展

✓ 对比学习相关论文阅读

从SimCIR开始(最早提出Project结构)





A Simple Framework for Contrastive Learning of Visual Representations

本　周　工　作　进　展

✓ 对比学习相关论文阅读

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}, t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

对比学习到底如何设计才能有效呢?

为什么在进行相似度计算的时候用到L2范式

Projector结构的作用

loss function的选择(X)

loss function中T的影响

ℓ_2 norm?	τ	Entropy	Contrastive acc.	Top 1
Yes	0.05	1.0	90.5	59.7
	0.1	4.5	87.8	64.4
	0.5	8.2	68.2	60.7
	1	8.3	59.1	58.0
No	10	0.5	91.7	57.2
	100	0.5	92.1	57.0

Table 5. Linear evaluation for models trained with different choices of ℓ_2 norm and temperature τ for NT-Xent loss. The contrastive distribution is over 4096 examples.

A Simple Framework for Contrastive Learning of Visual Representations

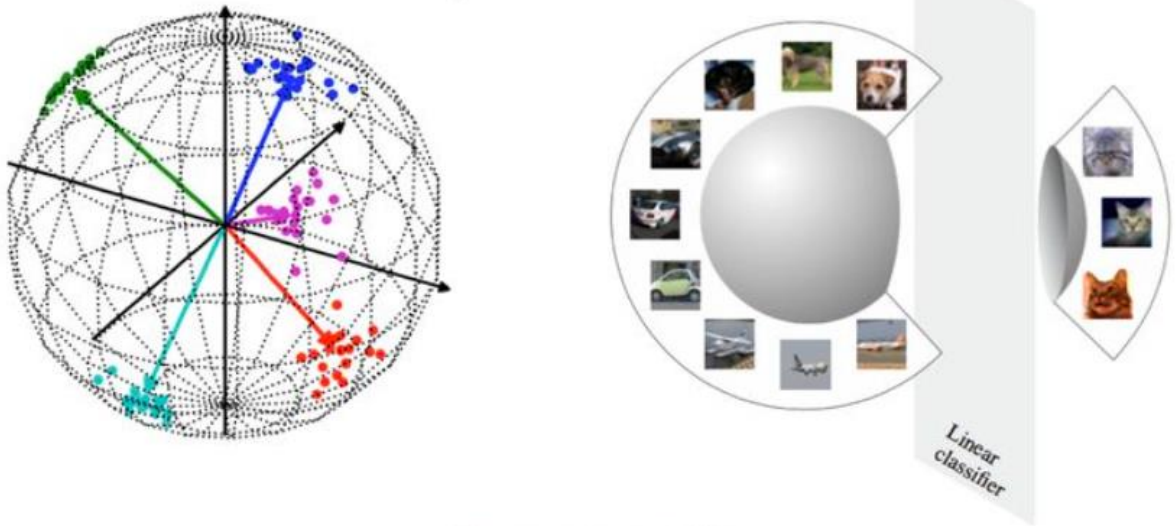
本　　周　　工　　作　　进　　展

✓ 对比学习相关论文阅读

- 特征表示的相似性计算

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

- 很多相似性计算：对向量做L2正则，再点积运算 或者 直接采用Cosine相似性
- 即 先把向量映射到一个超球面上，之后用两个向量的夹角的大小 表示 两个向量之间的相似性。



单位超球面及线性可分

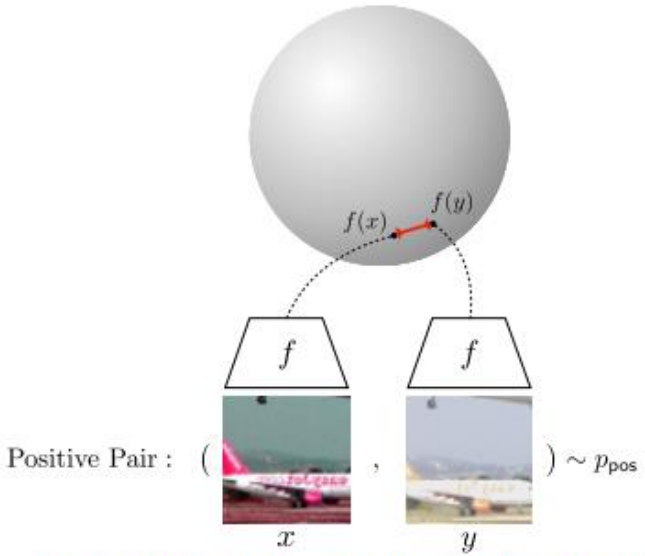
Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere

本　周　工　作　进　展

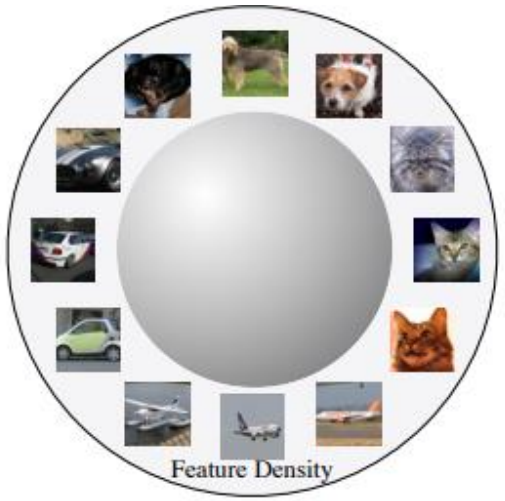
✓ 对比学习相关论文阅读

模型坍塌

线性可分制导原则可分进一步可以规范为：**Alignment 和 Uniformity**



Alignment: Similar samples have similar features.
(Figure inspired by Tian et al. (2019).)



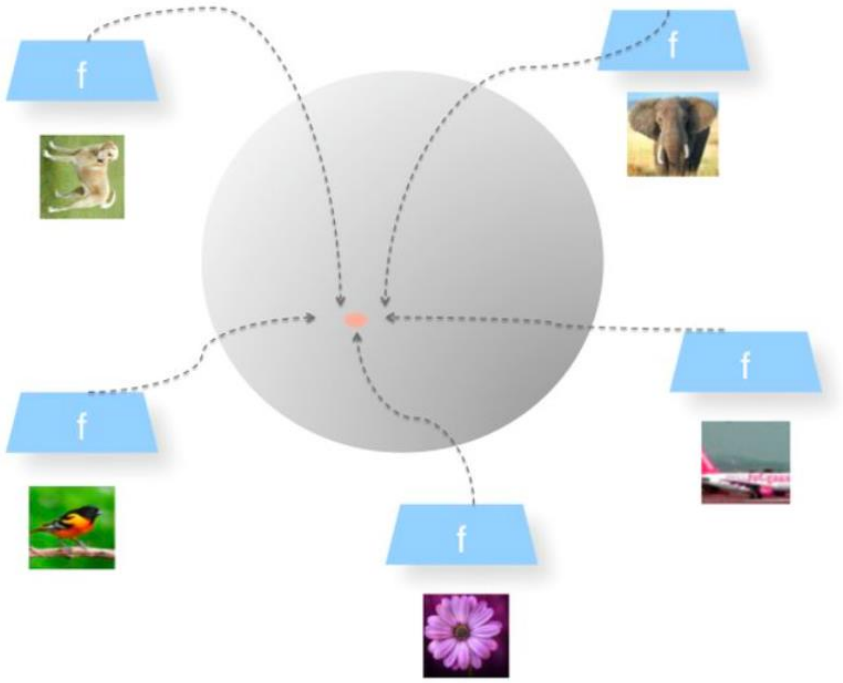
Uniformity: Preserve maximal information.

Figure 1: Illustration of alignment and uniformity of feature distributions on the output unit hypersphere. STL-10 (Coates et al., 2011) images are used for demonstration.

Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere

本　周　工　作　进　展

✓ 对比学习相关论文阅读



模型的坍塌

模型坍塌

$$\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$$

负例承担了Uniformity职责，防止模型坍塌

温度在这里起到soften作用。T越小，对困难样本的惩罚就越大

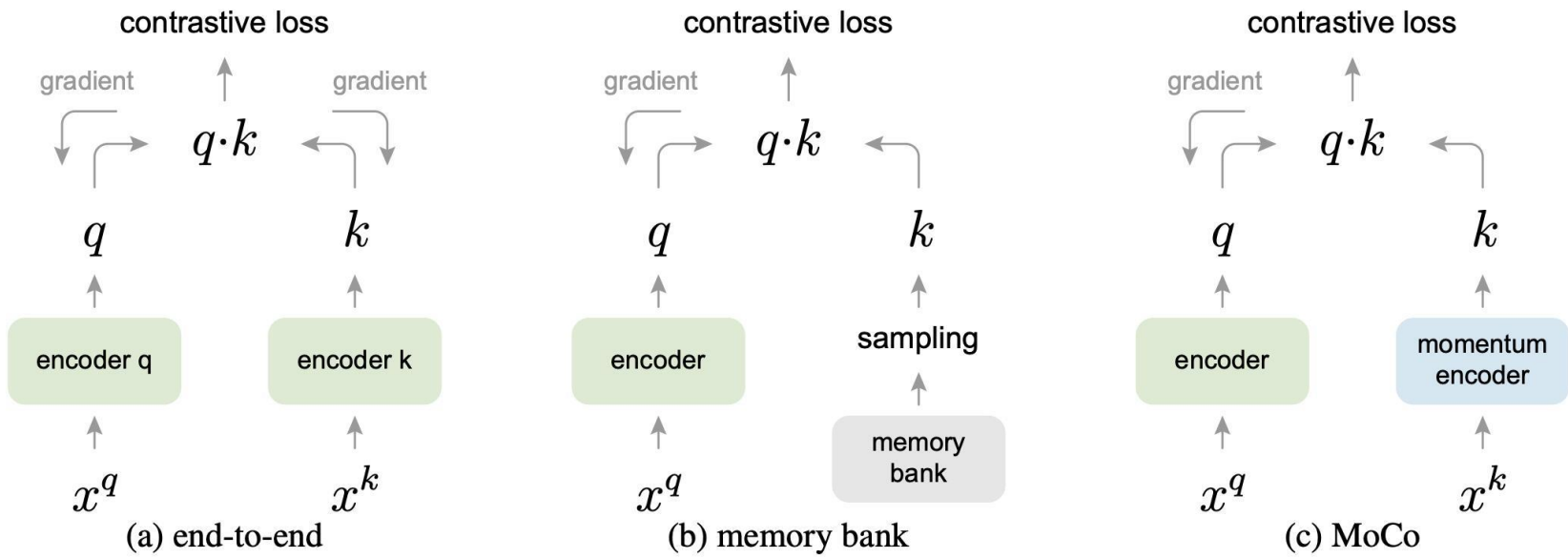
如果依靠负样本实现Uniformity，负样本的数量也很关键

Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere

本　周　工　作　进　展

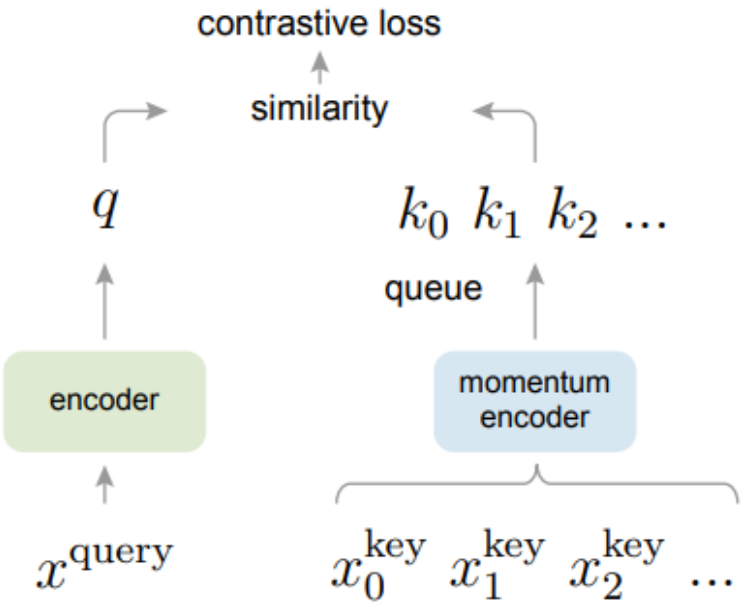
✓ 对比学习相关论文阅读

MoCo：利用一个队列来维护负例，并设计了一个momentum encoder进行缓慢的梯度更新，来维持负例中的相似性



本　周　工　作　进　展

✓ 对比学习相关论文阅读

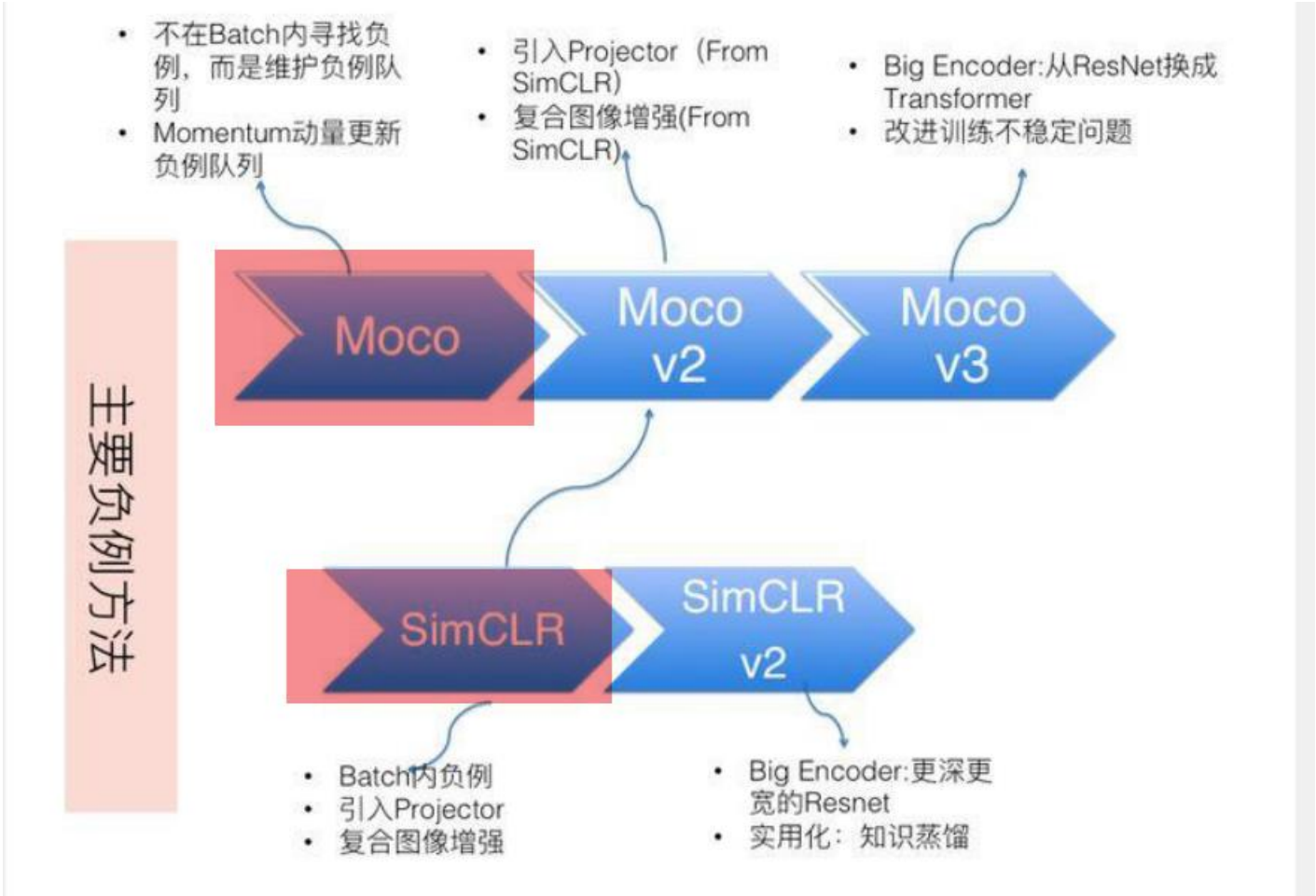


Algorithm 1 Pseudocode of MoCo in a PyTorch-like style.

```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
   x_q = aug(x) # a randomly augmented version
   x_k = aug(x) # another randomly augmented version

   q = f_q.forward(x_q) # queries: NxC
   k = f_k.forward(x_k) # keys: NxC
   k = k.detach() # no gradient to keys

   # positive logits: Nx1
   l_pos = bmm(q.view(N,1,C), k.view(N,C,1))

   # negative logits: NxK
   l_neg = mm(q.view(N,C), queue.view(C,K))

   # logits: Nx(1+K)
   logits = cat([l_pos, l_neg], dim=1)

   # contrastive loss, Eqn.(1)
   labels = zeros(N) # positives are the 0-th
   loss = CrossEntropyLoss(logits/t, labels)

   # SGD update: query network
   loss.backward()
   update(f_q.params)

   # momentum update: key network
   f_k.params = m*f_k.params+(1-m)*f_q.params

   # update dictionary
   enqueue(queue, k) # enqueue the current minibatch
   dequeue(queue) # dequeue the earliest minibatch
```

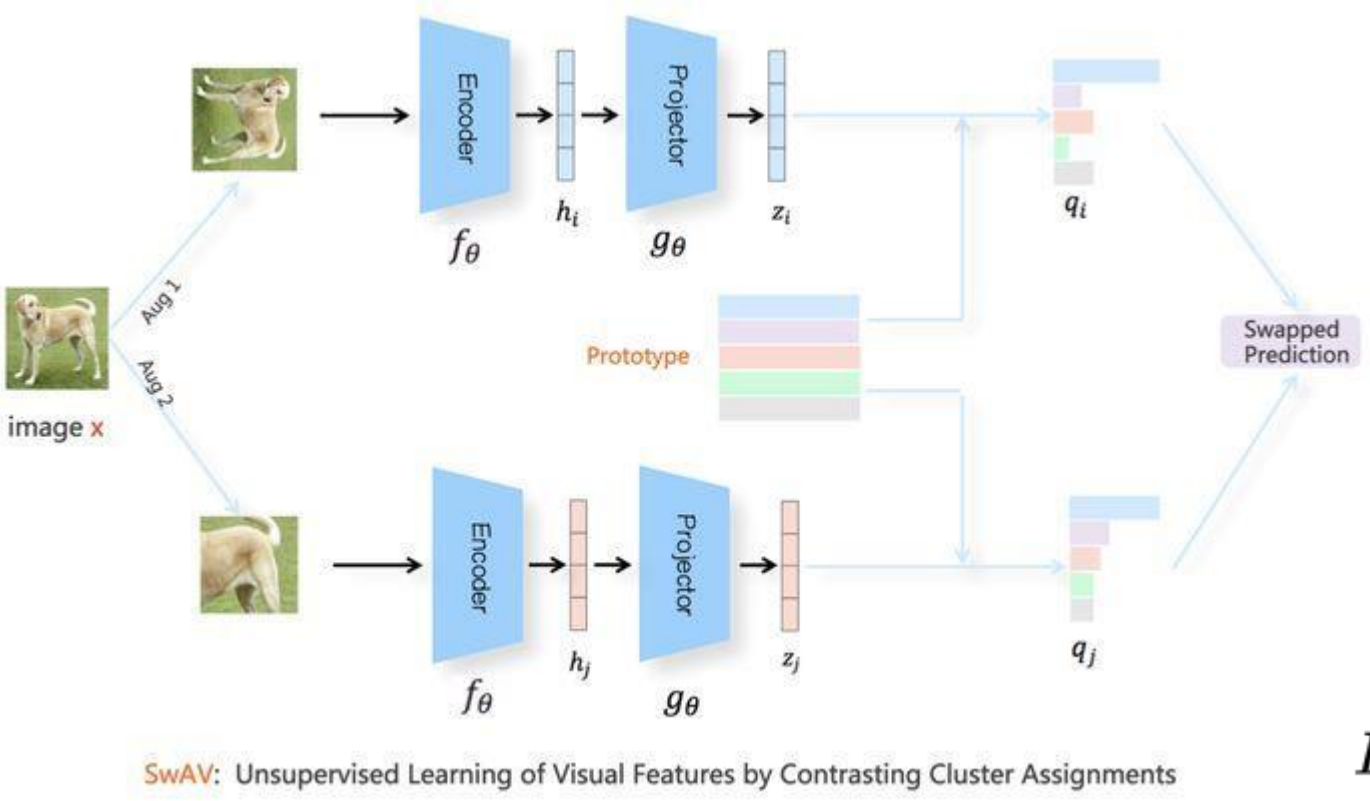
本　周　工　作　进　展

✓ 对比学习相关论文阅读



本　周　工　作　进　展

✓ 对比学习相关论文阅读



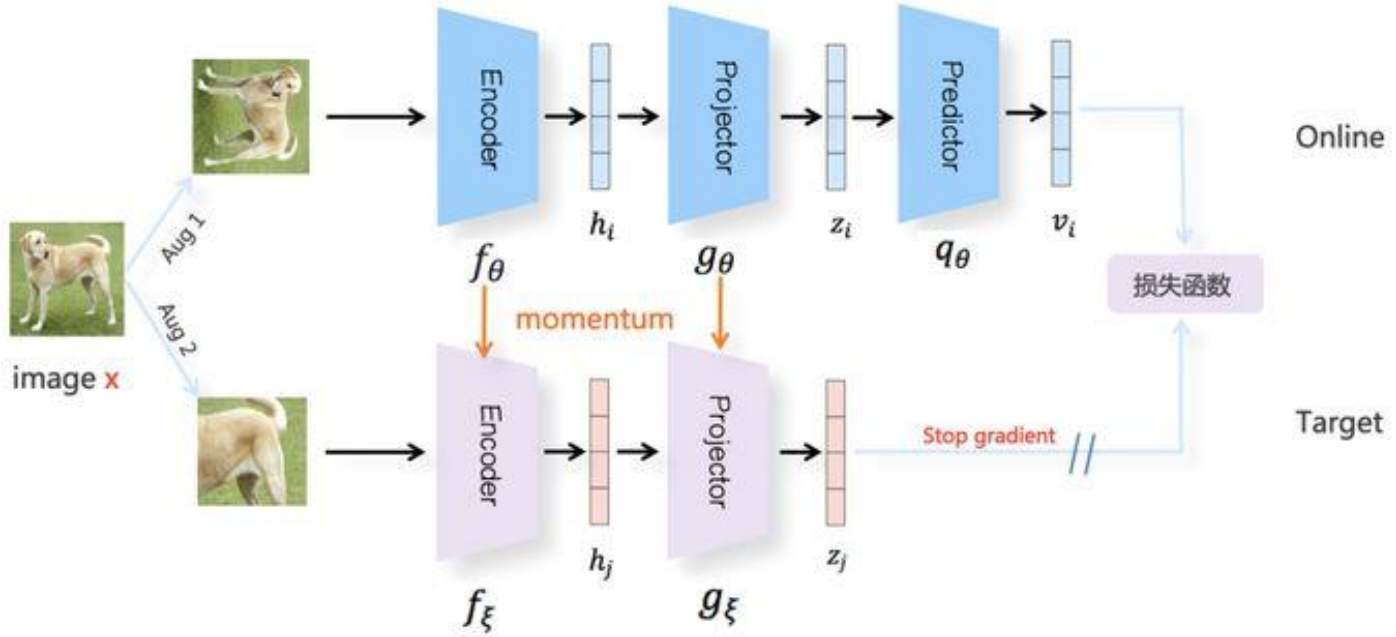
根据Sinkhorn-Knopp算法，在线对Batch内数据进行聚类

为了防止模型坍塌，SwAV对聚类增加了约束条件，要求Batch内实例比较均匀地聚类到不同的类

$$L_{SwAV} = L_{aug1}(z_i, q_j) + L_{aug2}(z_j, q_i)$$

本　周　工　作　进　展

✓ 对比学习相关论文阅读



BYOL:Bootstrap Your Own Latent A New Approach to Self-Supervised Learning

Predictor模块

本　　周　　工　　作　　进　　展

✓ 对比学习相关论文阅读

Summary:

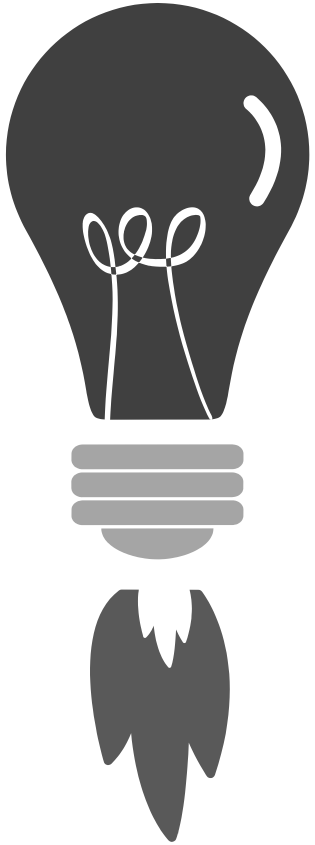
1. 无论是基于负例还是负例-free的方法，主要是围绕Alignment 和 uniformity 原则
2. 未来的研究方向:
   - 像素级学习能力
   - 构建正例和负例的方式
   - 训练数据偏移问题(迁移学习)

02

工 作 经 验 总 结

Life was like a box of chocolates, you never know what you’re go to get.

工　作　经　验　总　结



1. 准备着实总结一些比较有用的工具代码
2. 一些比较基础的知识还不是很扎实
3. 目前知识面还不够广

03 未来工作计划

下　周　工　作　计　划

上周遗留工作:

- Semi-Supervised Semantic Segmentation with Cross Pseudo Supervision阅读

论文阅读计划:

- CPC：Representation Learning with Contrastive Predictive Coding (需要补充学习一些NLP的一些知识)
- A SURVEY ON CONTRASTIVE SELF-SUPERVISED LEARNING（综述文章）
- 负例-free的一些方法
- CASTing Your Model:Learning to Localize Improves Self-Supervised Representations
- Demystifying Contrastive Self-Supervised Learning:Invariances, Augmentations and Dataset Biases(构造正例)
- 对比学习和半监督相关相关内容（目前还不能很好的联系起来）
- 生成式网络(目前还不是很了解)

代码:

- pytorch学习更深层一些，主要是梯度更新等一些底层的问题(官方文档和博客)
- MoCo源码(整理出一些东西)
- SimCLR源码

谢谢