dw 1012

今日要做的事有：

1. npu论文大概干了什么怎么做的
2. clip论文
3. gatys代码研读，搞懂每个模块干了什么
4. 写好自辩课演讲稿并且模拟练习
5. 完成本周周报
6. 完成leetcode每日一题，有空的话一天三题

4 完成了一半 目前写完了。中午去练习一下。

2 看李沐的clip讲解论文：**Learning Transferable Visual Models From Natural Language Supervision**（openai的作品）

clip的迁移学习能力很强，且zeroshot

直接从原始文本中学习关于图像的知识是一种很有前途的选择，它利用了更广泛的监督来源。我们证明了简单的训练前任务，预测哪个标题与哪个图像是一种有效的和可伸缩的方法，从互联网收集的4亿（图像，文本）数据集学习SOTA图像表示。该模型非平凡地转移到大多数任务，并且通常与完全监督的基线竞争，而不需要任何特定数据集的训练

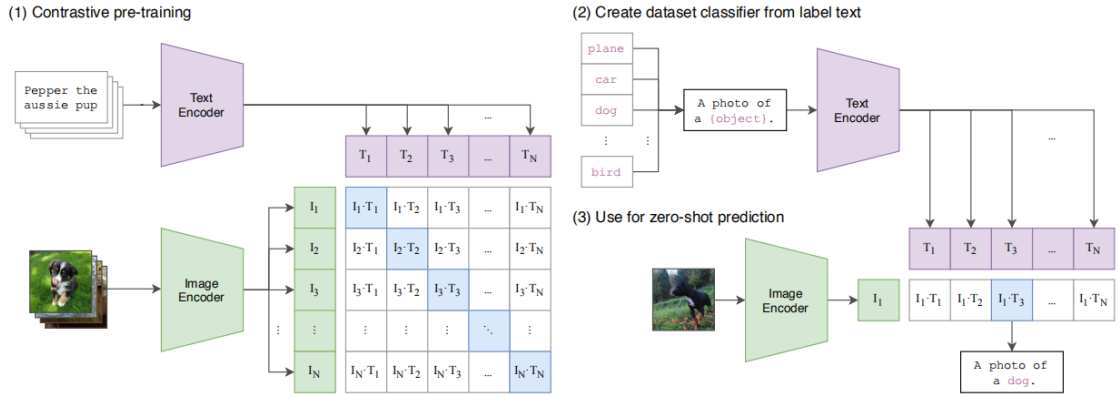


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

clip的预训练：模型输入是图片-文字对

然后endocer后进行对比学习、无监督预训练、大概可以通过流程图看得懂流程。

相关工作 style-clip 就是stylegan+clip的工作。； ClipDraw文本生成简笔画

clip还可以用于目标检测和图像分割；或者用来做视频检索（检索一个视频内有没有出现某个东西）

文本进入--文本出的自监督预训练模型。

这篇文章 李沐的视频讲的很好~

Contrastive Language_Image Pretrainning

clip泛化性很好，想法简单 性能高效

对比任务：只要判断这个图片和这个文本是不是一对就行，不用进行预测生成

```
# 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
```

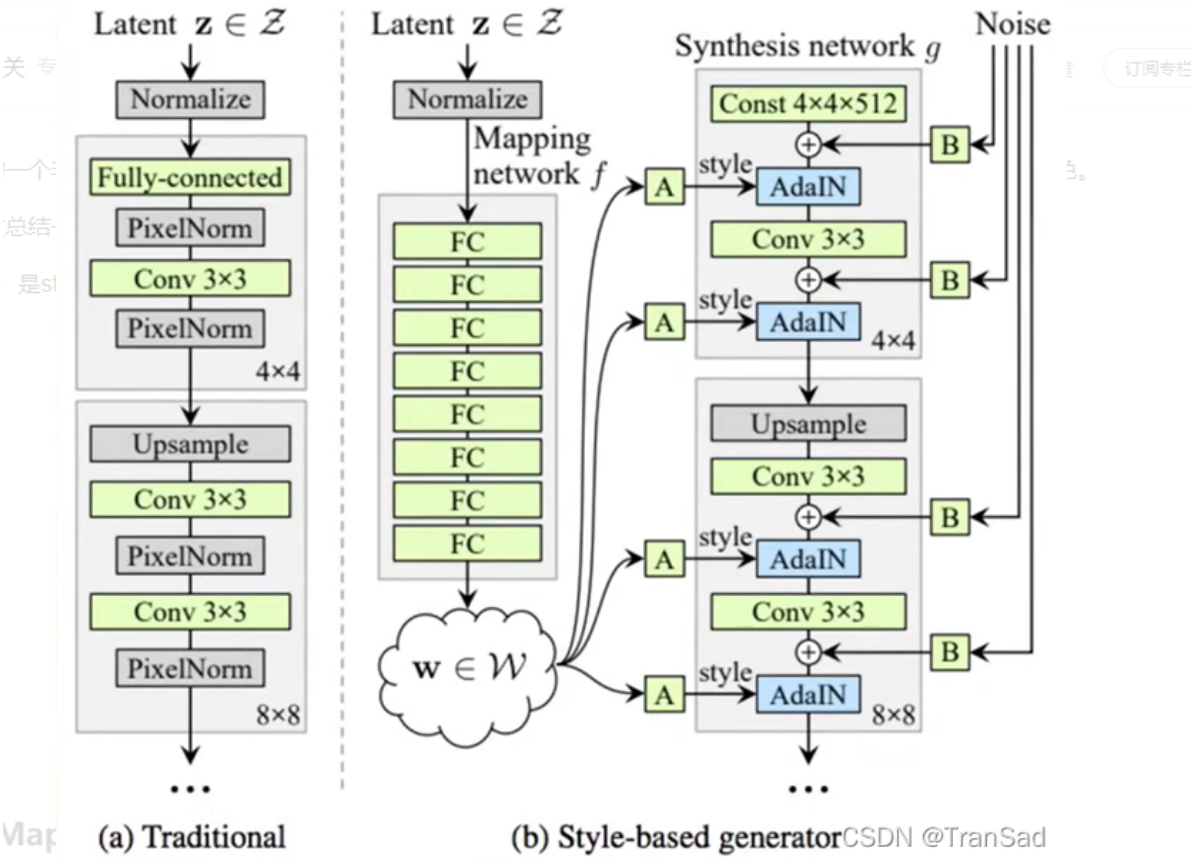
Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

预训练伪代码

clip使用了vit进行图像段的encoder

附加： StyleGan： Stylegan的一个非常实用的人脸生成模型，它可以通过控制参数来控制人脸生成的样式，包括五官、发型乃至人种肤色。

这篇文章大致总结一下Stylegan的一些原理和要点。



从上图可以看到，传统的生成器的输入就是直接用高斯分布下512维的随机向量。而Stylegan中使用了8层全连接网络Mapping network来进一步处理输入，将它映射到另一个512维，并未改变其维度。此外，我们可以发现传统的生成器输入数据只送到了第一层；而在StyleGAN中，通过Mapping Network处理之后的W'会陆续地影响很多层，这是用来控制不同层次下的样式的。这一点在下面的AdaIN部分再细讲。

使用Mapping Network的好处在于：

传统的输入肯定是高斯分布，它有可能和实际生成器使用的真实数据输入有区别；而通过一个全连接网络，就可以消除高斯分布的影响，出来的新512维的数据可以是任意的分布，它可以更好匹配我们的真实数据。

（第2点在看完下面的AdaIN部分后再看会清楚一些） 如果直接用输入向量来送入生成网络的各个层去控制不同层次下的信息，会有"特征纠缠"的现象。这个现象简单来说会导致：当我们想控制低层次下的粗略信息时，也会影响到高层次下的细微信息，会有"牵一发而动全身"的效果——显然我们不想这样做，我们就想不同层次各自控制各自的。而通过这么一个Mapping Network我们就可以让处理后的输入数据不会有这个现象。

关于生成网络（图中的Synthesis network）部分，我们看起来好像是4*4的网络之后连接上一个8*8网络，不同大小的网络层似乎是串联起来的——但实际上，**生成网络当中只有一个结构，并在训练的过程中不断地变化**。也就是一开始是4*4，后来变成8*8……最终变成1024*1024。

这源自于Stylegan的前身ProGAN，在ProGAN模型中，我们发现生成器如果采用这样变化的策略，可以首先学习到低分辨率图像的特征。有了低分辨率图像特征做基础，随着分辨率的提高和训练，慢慢增加网络复杂度的同时，就学习到越来越多的细节，这是一个非常自然、快速而水到渠成的训练过程。

那这里的生成模型是如何实现网络的动态变化呢？这里比较粗略地提一下：比如在图中我们发现，从4*4变化到8*8需要进行一个Upsample，我们的变化就从这个Upsample展开：我们知道，如果初始为4*4，如果不经过Upsample，那么输出依然是4*4的没有变化；如果经过了Upsample则会变成8*8——显然我们有两条路：经过Upsample与不经过Upsample。

我们给经过Upsample路径一个权重α，不经过就是（1-α），那么一开始我们网络结构不变化的时候，α=0；然后我们逐渐增大α，网络结构开始平滑地发生变化，此时对于一个batch的样本，就有α的比率走进了8*8的网络，（1-α）继续保持4*4；当α=1时，则结构正式变为8*8。（从8*8变化至16*16也是类似。）

这篇就简单总结了一下生成模型stylegan的一些要点部分，包括Mapping Network，AdaIN等。除了以上的几个主要的要点之外，StyleGAN还有很多其他细节方面的操作，比如加入噪音来控制多样化、微调超参数等，这些可以在论文中继续研究。