Learning Transferable Visual Models From Natural Language Supervision笔记

> hhhhh，模电缓考终于结束辣！！！接下来就好好跟师兄混，顺便准备下GT和继续套磁辣！！！芜湖，真开心！

今天这篇CLIP给予了我莫大的震撼。一个是CLIP真NB，这就是多模态嘛，尤其是后面的video understing任务里表现实在太惊艳了；另外一个就是我终于领教了什么是OpenAI风格，简单来说就是财大气粗，力大飞砖，他们怎么那么有钱哇qwq，属于是新时代的贝尔实验室了。

Abstract

CLIP是一种结构和思想都非常简单的多模态模型，它的迁移能力非常强，并且他在zero-shot（零样本学习）上效果非常好。

在往常的视觉任务中，人们习惯于固定分类标签，比如说image-net就固定1000个类，cifar-10就固定1个类。这样做固然可以简化问题，但也带来了代价：模型的泛化能力不够强。简单来说，imgae-net训练好的模型，他就只会对这1000个分

类，一旦出现新的类，他就开始瞎猜了。

而CLIP则克服了这一缺点。利用自然语言的文本作为监督信号，同时将多段文本和一张图片作为一组输入，接着让图片通过对比学习和其中一段文本配对。

CLIP的结果非常惊艳，即使CLIP没有用image-net的任何一张图片训练过，它也可以打败image-net上预训练的Res50。

Introduction

CLIP是一种完全和下游任务无关的模型，它使用一种“text-to-text”架构（输入的是多段文本，输出是其中一段文本），这样就不用针对下游任务做特定分类头，迁移能力很强。

Related work

- Virtex

Transformer架构，使用自回归模型

- ICMLM

Transformer架构，使用cbow（完形填空）

- ConVIRT

基本完全相似，但是只在医疗图像上做，数据集太小没好效果。

还有一系列的工作，处于“实用主义的中间地带”，他们也知道使用固定分类是有限制的，但是他们计算资源不够无法得到更好结果，只能退而求其次使用类似弱监督信号。

于是OpenAI提出，不是架构玩不起，而是规模上不去。他们用了many GPU（甚至写过一篇How to trian big model in many GPUs）来训练模型，数据集样本足足有4亿个（这个数据集还是自己找人打标做的），同时mini batch也达到了几w的水平（重新定义mini）。

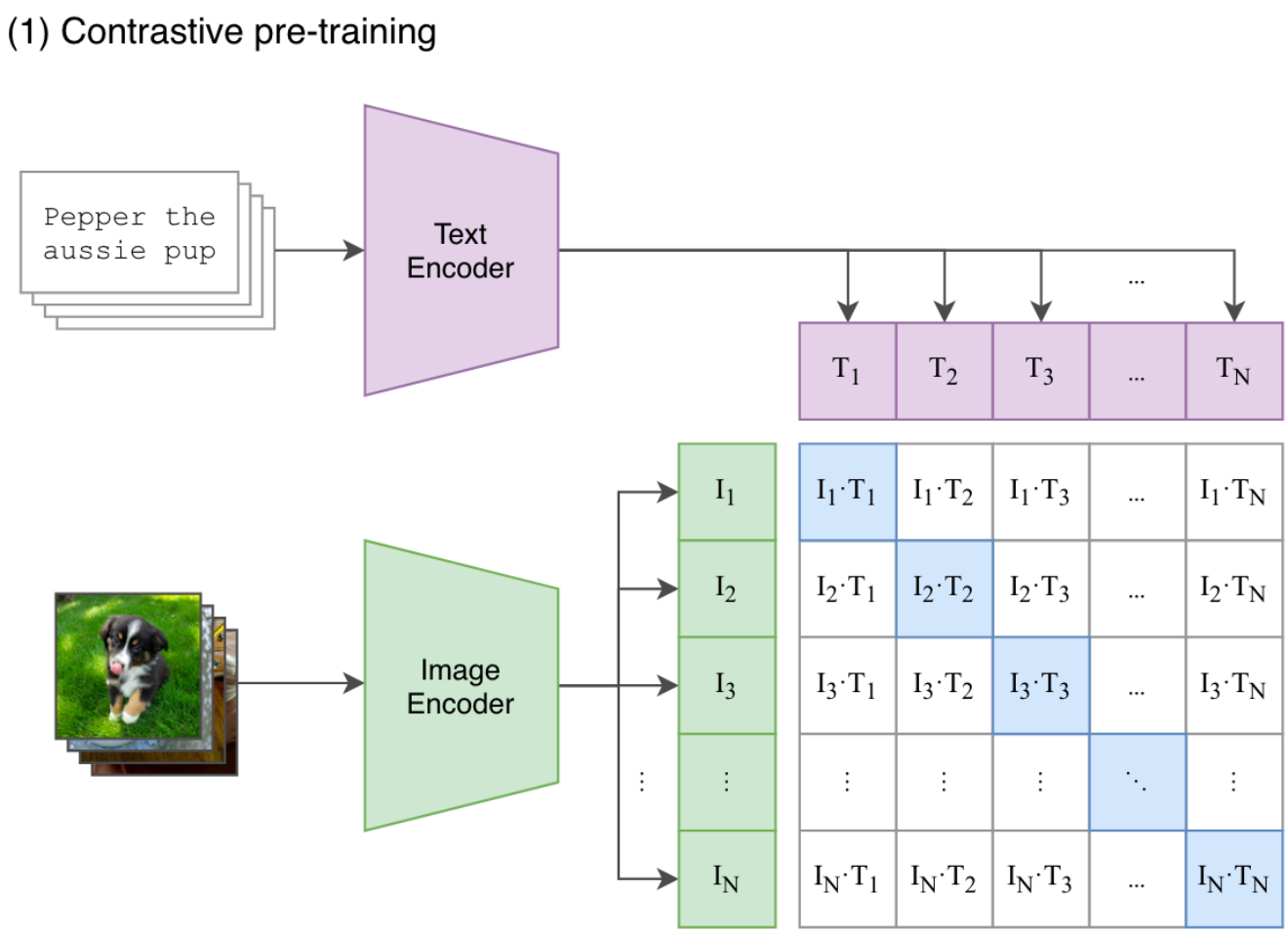
为了证明CLIP模型学习到特征的有效性和迁移性，CLIP没有使用image-net进行微调，而是将backbone冻住（linear probe），只使用网络抽取特征，训练最后一层的分类头。结果发现它的泛化性和稳健型都很强，即使在没见过的数据集和对抗性任务上都可以击败有Res-50的微调模型。

Method

网络架构

CLIP使用对比学习进行预训练，简单来说，假如我们有以下图片，并将它编码成I。

我们会一并输入以下文本集合{“正在打篮球的人”，“正在打羽毛球的人”，“正在撸猫的人”，“正在打看电视的人”}，再把他们添加进句子A photo of a {object}，通过编码器将他们编码成embedding向量T1，T2，T3。则I应该和T1最接近，loss最小；同理，I和T2、T3构成负样本，loss应该尽可能大。类似的如果我们输入I1、I2、I3和T1、T2、T3（相同序号是一对正样本），则他们之间的loss计算可以写成这么一个矩阵

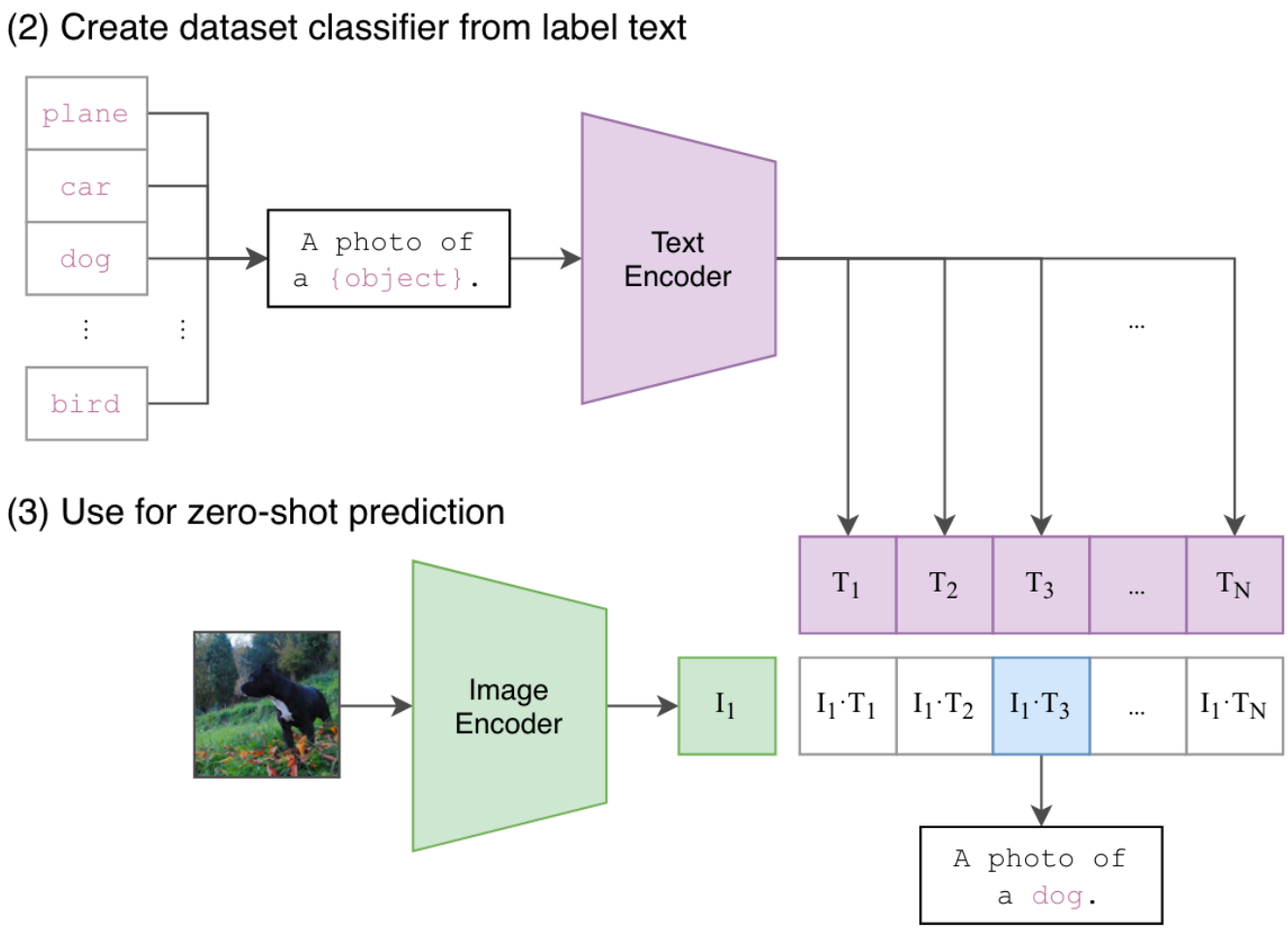


接着我们使用代理任务。用I的特征来预测T的特征，反过来再用T的特征预测I的特征，两个loss平均下就是最终特征。

```
# 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.

在预训练完毕后，我们就可以随便找张模型没见过的图片，比如一支布偶猫，通过embedding得到向量I；再输入一堆文本，比如{"cat","dog","pig"}，通过embedding得到向量T1，T2，T3。根据余弦相似性，I和T1得到的结果应该是最像。



提示工程

估计大家也发现了，为什么输入文本后不直接embedding成向量，还要再把他们添加进句子A photo of a {object}呢?

这是因为我们在预训练时输入的是完整的句子，我们想要在形式上达到统一（毕竟句子和单词还是非常不一样的），这就是prompt engineering（提示工程）。

同时我们也注意到，其实一张图片给不同的人来打标签，结果是非常不一样的。拿上面的坤坤来说，你可以说他是“一个在打篮球的人”，也可以说“一个kunkun”，亦可以说他是“一个穿背带裤的人”。因此，我们只需要让模型学习文本和哪个图片配对就行，不用知道它和哪个特定特征配对。

如果你非要提高模型的能力，让他知道所有特征，，那你可以一个句子里有多个分句，比如“一个打篮球的人，一个真正的man，一个穿背带裤的人”。这就是prompt ensembling（提示合奏）

深层次讲，这两个prompt也是非常重要的。

1）一个词语往往会有多义性，经典的crane会有“丹顶鹤”和“起重机”两个意思，你没有上下文就不知道他到底是什么意思；

2）数据分布偏差（distribution gap）。如果在只输入单词，通常数据的分布会和预训练时数据分布（预训练输入的是句子）不同，导致模型效果很差。所以作者采用了prompt template结构，套个模版性能突飞猛进。

3）更加精确定位。使用prompt ensemble，比如一只躲在角落的猫猫，你第一句是“This is a photo of anmimal”，不太明确，你再加一个“This is a photo of cat”，最后再加一个“This is a photo of one hard to see”这样可以尽可能描述一个物体的状态。

数据集

作者在做大数据集像image-net这些时，CLIP都是爆杀其他模型，但是在做手写数字数据集MNIST时，发现效果居然还不如MLP。后面作者发现，这是因为自己给CLIP用的数据集中一张相似的图片都没有。所以说CLIP一个巨大的缺点是对数据集要求还是蛮高的，要求尽可能大和全。

实验

我觉得特别好玩的是有个视频检测工作，场景是个十字路口。你输入“A photo of BWM”，还真的可以找出那一帧图片。但是这也引起了人们的担忧，他们看到了CLIP的潜力，也知道这玩意可以用来干非法的事情。所以说OpenAI一直注重安全问题也是有理由的吧。

其他的先不写了，未来有机会再补充。