前言：

方法出奇简单，效果出奇的好。

不使用任何一张训练集图片的情况下，就可以和预训练的ResNet-50达到相同的效果，ImageNet的分类。

从自然语言模型中获取监督信号，对视觉领域的图片进行分类。

How？

输入的是一张图片和一小段文字的pair，图片通过图片编码器，文本通过文本编码器，得到各自特征。Clip对这些特征进行对比学习。矩阵中的对角线上的就是正样本，N个正样本，N^2 - N个负样本。对比学习训练起来，无需手工标注。但需要大量数据。

如何做zero shoot处理？

Prompt template，可以在模板类中自己加入各种单词，单词转化为句子，通过编码器。作为template。

输入时，输入一张图片，提取特征，将图片特征和文本特征对比，计算相似度，找出最合适的描述。将视觉语义和文本语义联系到一起，迁移效果特别好。

还能基于CLip生成图像。

用于输入一段文字，输出一张图片，开放语义特征，还能识别出预定义之外的类别。

从视频中找出有关键语义信息的图片特征。

**题目：**

利用**自然语言的监督信号**，学习一个**迁移性能好**的视觉模型。目的：迁移性。学泛化性特别好的特征，对各种数据集都能有特别好的效果。

**摘要：**

现有的计算机视觉系统模型来说，先有固定、提前定义好的物体的集合，比如ImageNet等。减小了误识别的可能，但也对于要识别新物体类别的时候，就很难。于是有了利用文本信息特征去辅助识别的方法。找了400亿的数据集，自然语言引导视觉模型进行分类、并扩展到新的类别。超过30个不同的数据集上做了测试。不需要任何训练的情况下，和有监督的ResNet-50打成平手。很惊人。

**引言：**

直接从原始的文本数据里去预训练一个模型，过去几年中，在NLP领域取得巨大成果。比如BERT，GPT等，无论自回归预测还掩码是完形填空方式，都是自监督的训练模式，目标函数和下游任务无关，只是想通过预训练得到一个泛化特征特别好的模型。

随着计算能力增强，数据量的增多，模型能力稳步提升。但这样的系统只是“text-to-text”（文本进文本出）的模式，就比如GPT-3就是如此，对于大多数下游任务，并不需要具体的训练数据集。不过在视觉领域，大多还是用标注好的数据集，那能否将NLP领域的框架用在视觉里呢？Zero-shot迁移性的学习，和Li et al（2017）的工作特别相似。17年并无Transformer和大量的数据集。VirTex， ICMLM和ConVIRT等都在Transformer问世后尝试这种迁移学习的方法，都很接近或类似CLIP，但具体做法有存在些区别。VirTex采用自回归的预测方式做预训练，ICMLM用完型填空方式做预训练，ConVIRT和CLIP很相似，但只在医疗图像上做了测试。为什么视觉领域这种方法这么少？没大量的数据集，没好的自监督模型，因此精度很低，并没有很多人关注。

用标注好的数据是有很大局限性的，文本带来的弱监督信号，帮助有监督的模型进行更好的分类，虽然取得了很好的结果，但很难识别出新类别。对于zero-shoot的能力还有待提升。此前用于训练的数据是亿级别的，只是规模还不够，并不是方法不好。作者首先构建了一个新的4亿的数据集，和之前的工作持平，模型尝试上最大用了VIT-Large。大模型+大数据。

精度和模型规模正相关，可大概估算。

**方法**

**模型：**

方法的核心就是利用自然语言的监督信号，思路其实并不新，但之前的方法对于用语有些不妥。例如Zhang和Gomez和Joulin都是用到了文本配对图片的方式，但他们却将这种方法描述为无监督的、自监督的、弱监督的。作者的工作无非就是总结前人的这些经验、方法，并加大模型和数据集的规模。核心就是将文本作为训练的信号。在Transformer之前，NLP的模型其实并不是很好学，随着上下文具有语义环境的学习方式的发展（比如BERT），使得在自监督的范式之下，文本方面的监督信号拥有巨大的资源。所以说NLP训练出来的模型逐渐变得又大又好，简单、泛化的又好。很适合多模态的学习。

**Why使用NLP做视觉？**

1. 不用标注。只需要下载文本和图片的配对，方便得到数据集。
2. 文字和视觉特征绑定到了一起。学习到的特征也得到了多个维度，多模态。方便做zero-shoot的迁移。

**需要足够大的数据集：**

对于已有的数据集来说，大规模的数据集存在文不对图的情况，清洗过后数据集规模又会大规模下降。最终合适的大小都和ImageNet差不多。但也认为不够大。

So，造了个大数据集。收集了400million的图片文本对。什么概念，比视觉领域google-GFT还多一个亿，和NLP领域的GPT-2差不多接近。

**选择高效的预训练方法：**

视觉方面的模型都非常大，训练起来非常贵。还只是预测1000各类。对于开放视觉分类来说，训练这样的系统即使是OpenAI这样拥有大量计算资源的团队，都认为这是一个惊人的训练量。

尝试：1. 图像CNN， 文本Transformer。通过图片预测文本。

2．对比学习。预测型去训练，对于一个场景会有很多种可能的解释，训练就会非常慢。

只需要判断图片和文本是不是一个配对。预测型换成对比型，效率直接提升四倍。

**模型伪代码：**

两个输入：一个图像，一个文本。编码。ResNet or vision transformer， CBOW or Text Transformer

归一化。得到多模态、用来对比的特征。

算相似度。用于分类。

做交叉熵。算loss

**细节：**

数据集太大，训练不太会有过拟合的问题。投射层的选择，没用非线性；对比学习中，非线性往往会比线性投射效果高10个百分点；多模态线性和非线性没什么关系，非线性只是用来适配图片单模态的学习。

耗时，不好调参。设置为可学习的标量。

**实验：**

**什么是zero-shoot transfer？**

Xxxxxx

**如何只训练一个模型，之后就不再训练、不再微调了呢？**

这就是作者研究zero-shoot迁移的研究动机。

之前那种自监督或者无监督的方法，主要研究的是特征学习的能力，他们的目标是去学一种泛化性比较好的特征。即使学到了很好的特征，应用到下一个任务时，仍然需要有标签的数据集去做微调，就会牵扯各种各样的问题：下游任务不好去收集数据。

一旦借助文本训练了一个又大又好的模型之后，就可以用这个文本作为引导，去灵活地做这种zero-shoot的迁移。至少在分类上效果都非常好。

**怎么用CLIP去做这种zero-shoot的迁移呢？**

**F**igure 1 看推理。

预训练好之后，就会有两个编码器。

照片通过图片编码器，就会得到一个图片的特征。

而文本方面的输入，则是你感兴趣的，想识别到的一些类别。通过prompt engineering，这些文本会编成一个句子。汽车，变为“这是一张汽车的照片”。四个单词就会变成四个句子。四个句子通过文本编码器，就会得到四个文本的特征。四个文本特征和输入单个图像的特征去算similarity，得到的相似度通过一层softmax，得到一个概率分布，哪个概率最大indicate相似度最高，那对应的句子大概率是描述这张照片。对于ImageNet有1000个类，就会生成1000个句子，相当于每输入一个图片，都会用这1000个句子去问它，看和哪个文本最接近就是哪类。并不是顺次进行，而是批次进行的，所以推理还是很高效的。

**和Visual N-Grams跑分对比：**

Table 1 ， visual N-Grams在ImageNet上只有11.5%的准确率，而CLIP可以达到76.2%。

在完全没用1.28million张训练图片的前提下，和原版的ResNet-50得到相同的效果。

但这个对比并不完全公平，采用的数据集比visual N-Grams大了10倍，视觉上的模型比它大100倍的计算。所以在训练上用了超过1000倍的资源去训练。架构上也用了2017年visual N-Grams发表时没用提出的transformer。

**Prompt Engineering and Ensembling**

推理和微调采用的一种方法，而不是在预训练阶段，所以不需要很多的计算资源。

提示（引导）作用。

为什么要做这个？

1. 一个单词有很多含义，只用一个单词对应就会歧义。比如ImageNets中construction cranes是建筑中的工具，而单单一个crane是丹顶鹤。这样计算出的相似度就会有问题。
2. 在预训练时，匹配的文本都是一个句子，很少是一个单词，所以通过prompt就可以使输入时仅是一个单词的情况，转换为句子，使输入的分布差（distribution gap）得到匹配，使抽到的特征更好。

简单的方法：a photo of a {label}. 使这个label一定是一个名词。粗暴，但好用，提升1.3%。

提示的句子还可以根据数据集的不同进行修改，达到缩小选择空间的目的。比如已知是在宠物的数据集上测试，那就可以改为“a photo of a {label}, a type of pet”。

作者还采用多个（开源代码中是80个）提示模板进行分类后，将结果进行综合判定。

**实验结果（27个数据集）**

大多数结果超过有监督的ResNet-50. 对于普通的对物体的分类的数据集，效果比较好。

但 对于更难的数据集（对纹理、对图中物体计数等）就效果并不好。

对特别难的任务，zero-shoot的迁移太过苛刻，对于人来说，没有先验知识都很难做。

20个结果合并的图6，对比了在各个shoot下的表现。甚至在低样本下，还不如zero-shoot。

Zero-shoot, few-shoot, 所有数据都衡量了模型的好坏。

**模型泛化性：**

数据有分布偏移时，普通模型掉点就非常严重，而CLIP则十分稳健。

**和人对比：**

赛前不看各类型图片、看一张、看两张。但代表性并不够，只是做了个测试。体现CLIP强大。对于人和CLIP而言，准确度低的二者都低，高的都高。还是有联系的。

**模型局限性**

给后续研究留下更多发展空间。

1. CLIP性能强，但并没有达到不可一世的地步，本文对比的只是一个基线的模型，即ResNet-50，才和它再ImageNet上打成平手，但相较于最新最大的Vision Transformer和noisy student，还是有十几个点的差距。继续加大模型和数据集，CLIP模型的性能还会上升，但对比能达到88%的模型，想要弥补十几个点的差距，要再扩大1000倍，这显然需要新的方法。
2. 有些数据集上，zero-shoot效果也并不好低于基线的ResNet-50，细分类、抽象的数据集。因此，再很多领域，CLIP的性能和瞎猜是一样的。
3. CLIP的泛化做得很好，对自然图像的分布偏移，模型还是相对稳健的，但如果在推理时，数据偏移的非常远，泛化效果同样也很差。比如在MNIST数据集上，zero-shoot的CLIP仅有88%，一般的模型都能轻易达到99%。排查原因发现，预训练CLIP时，尽管有四亿个样本，但就是没有一个和MNIST长得像的，由于MNIST是合成数据集，并没有一些自然的特征。对于CLIP而言，就out of distribution了。
4. 虽然CLIP可以做zero-shoot的分类任务，但还是在给定的、期望的输出类别中选择，但更为直接的是直接生成输出。有个自然的想法是说，将生成式的目标函数和对比学习的目标函数合在一起，就有可能结合两种方法的优势。
5. 对数据的利用并不是很高效。32个epochs，每个epoch四亿个数据。
6. 测试时不断在已有的数据上测试，调超参数，其实已经偏离了zero-shoot的初衷，希望是能创建一个专门的数据集用来做zero-shoot迁移能力的测试，就会帮助解决很多问题。
7. CLIP所使用的文本、图像对是从网上爬取的，并没有经过严格的清洗和审查。因此训练出的模型很可能带有社会上的偏见，如性别、肤色、宗教等。
8. 很复杂的概念有些即使连语言都没法描述。同时在测试中还看到在few-shoot如one-shoot和two-shoot时，性能还不如zero-shoot，这是很耐人寻味且不符合一般人类规律的。因此后期研究还将聚焦于如何使CLIP能在zero-shoot和few-shoot上都取得高效的表现。

**对未来的影响：**

**结论：**

NLP与下游任务无关。复制到其他领域，cv。确实效果也不错。预训练（对比学习），大规模是聚集和大模型加持下，效果不错，但提升空间很大。

打破了固定种类标签的范式，收集数据集和训练模型时，不用再预定义各种固定的类了。

新颖度：打破了视觉领域固定标签的做法，彻底放飞了视觉模型的训练过程。

有效性：做得数据集多，效果好，泛化性好，某种数据集下比人的性能还好。

问题大小：一个模型解决大部分的分类任务。

Code

|  |
| --- |
| import numpy as np  import torch  import clip  from PIL import Image  # 选择加载模型，可选项为：ViT-B/32, ViT-B/16, RN50, RN50x16等  device = "cuda" if torch.cuda.is\_available() else "cpu"  model, preprocess = clip.load("ViT-B/16", device=device)  # 创建编码器和希望得到的图片类别  image = preprocess(Image.open("plane.jpeg")).unsqueeze(0).to(device)  text = clip.tokenize([ "a man","tesla", "robot", "Boston", "plane", "pig", "cartoon"]).to(device)  with torch.no\_grad():  # 提取特征  image\_features = model.encode\_image(image)  text\_features = model.encode\_text(text)    logits\_per\_image, logits\_per\_text = model(image, text)  probs = logits\_per\_image.softmax(dim=-1).cpu().numpy()  # 输出每个类别的概率，概率高的相似度高  print("Label probs:", probs) |