# 扫盲贴

# 标签种类

1. 直接类别标签(classes laber)
2. 二进制编码
3. one-hot编码（长度等于类别总数）
4. multi-hot编码

# one-hot编码的意义

1. **唯一**：每个类别由一个唯一的二进制向量表示，其中一个元素设为 1，其余元素设为 0。这种表示确保了每个类别都有一个独立的、互不相干的表示。
2. **避免数值关联**：由于类别通常是非数值性质的（如单词、类别标签等），使用数值表示可能无意中引入数学上的关联或排序。例如，如果用 1、2、3 来编码三个类别，算法可能错误地假设 3 是 1 的三倍，或者 2 比 1 “更大”。One-hot 编码避免了这种问题，因为它不引入任何特定的数值关系。
3. **适用于分类模型**：在分类任务中，尤其是在使用神经网络时，one-hot 编码提供了一种清晰的方式来表示多类输出。神经网络的输出层通常有与类别数量相同的神经元，每个神经元的输出对应一个类别。使用 one-hot 编码，可以直接将网络的输出与目标编码比较，常用于计算损失（如交叉熵损失）。
4. **简化模型学习过程**：使用 one-hot 编码，模型可以更容易地学习到不同类别之间的区分。在未编码或使用其他编码方式的情况下，模型可能需要更多的数据和复杂度来学习到同样的区分。

# embedding

用于将大量复杂的数据（如文本或图像）转换为一组更小、更密集的固定大小的向量。

## 在自然语言处理中的应用：

1. **词嵌入（Word Embeddings）**：
   - 例如，将单词转换为数值向量。
   - 这些向量捕捉单词之间的语义关系，如相似性和上下文关联。
   - 常见的词嵌入模型包括Word2Vec、GloVe和FastText。
2. **句子/段落嵌入（Sentence/Paragraph Embeddings）**：
   - 类似于词嵌入，但用于更长的文本单位。
   - 例如，BERT和GPT系列就是生成这种嵌入的模型。

## 在其他领域的应用：

- **图像处理**:
  - 将图像转换为密集的向量表示，以用于图像识别、分类等任务。
- **推荐系统**:
  - 将用户和项目转换为嵌入，以更好地理解它们的特性和关系。

## embedding的优点：

1. **降维**:
   - 将高维数据（如自然语言中的单词）转换为低维、更易处理的形式。
2. **捕捉隐含特征**:
   - 嵌入能够捕捉数据之间的复杂关系和模式。
3. **提高计算效率**:
   - 用密集向量表示数据可以提高计算效率。

# zero-shot

https://blog.csdn.net/zcyzcyjava/article/details/127006287

- zero-shot :没样本

> 指的是我们之前没有这个类别的训练样本。但是我们可以学习到一个映射X->Y。如果这个映射足够好的话，我们就可以处理没有看到的类了，也就是我们可以看到不在训练label里面的classes

> 也就是利用过去的知识（马，老虎，熊猫和斑马的描述），在脑海中推理出新对象的具体形态，从而能对新对象进行辨认。（如图1所示）ZSL就是希望能够模仿人类的这个推理过程，使得计算机具有识别新事物的能力

- one-shot :只有一个样本也能学
- few-shot :一点点样本也能学
  类比一下few-shot就是给你看两三个你没见过的

# 软标签

^68107c
用一个概率分布来表示每个类别的可能性，而不是用一个固定的数字或符号（即硬标签）。例如，对于一个面部表情，可以用一个向量来表示它属于开心、悲伤、生气等类别的概率，而不是只用一个标签来表示它是哪一种表情。这种方法可以更好地处理注释模糊性的问题，因为它可以反映出数据的不确定性和多样性. 但是，这种方法也有一些缺点，例如，需要更多的计算资源，以及如何定义合适的软标签分布等。

# 余弦相似度

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \times \|\mathbf{B}\|}$$
将两个向量之间的相似度定义为它们之间的夹角余弦值, 实际上这个在前面已经思考过了卷积神经网络 > ^dfde17

# Benchmark

指一组测试套件、任务集、数据集、评判基准。用于衡量算法、模型或计算机系统性能。

# metric

Metric是指用于衡量模型性能的指标，例如准确率、召回率、F1分数等。而Benchmark是指用于比较不同模型性能的标准，例如在特定数据集上的最佳模型或人类表现。Benchmark可以包含多个Metric，但Metric不能替代Benchmark

# SOTA

"State of the Art"（技术最先进或技术最高水平）的缩写，通常指的是在特定领域或任务的benchmark测试中表现最好的已知模型、算法或方法。

***在一个benchmark中，我将SOTA作为我的baseline，我网络的backbone是我自己研发的，最终获得了更优越的结果***

# 端到端(end-to-end)

> 1. 在一个典型的NLP问题中，包括分词、词性标注、句法分析、语义分析等多个独立步骤，每个步骤是一个独立的任务，其结果的好坏会影响到下一步骤，从而影响整个训练的结果，这是非端到端的. 类似分治法.
> 2. 深度学习则为我们提供了另一种范式（Paradigm）即“端到端”学习方式，整个学习流程并不进行人为的子问题划分，而是完全交给深度学习模型直接学习从原始输入到期望输出的映射。相比分治策略，**“端到端”的学习方式具有协同增效的优势，有更大可能获得全局最优解。**

举例:

> BERT新技术简介 > BERT 是一种预训练的自然语言处理模型，它的输入是原始的文本序列，输出是文本序列的向量表示，因此它可以看作是端到端的。
> 而ResNet卷积神经网络 > ResNet 是一种预训练的计算机视觉模型，它的输入是原始的图像，输出是图像的特征向量，还需要一个后续的分类器或者回归器来完成具体的任务, 所以它不是端到端的.

# 蒸馏

知识迁移的技术，让一个小的模型（学生模型）从一个大的模型（教师模型）或者多个模型（教师集合）中学习知识。
蒸馏的基本思想是让学生模型去拟合教师模型的输出，而不是直接拟合数据的标签。这样，学生模型可以从教师模型的软标签（soft label^68107c）

中获取更多的信息，提高自己的泛化能力。蒸馏的过程可以分为以下几个步骤：

- 首先，用大量的数据训练一个或多个教师模型，得到它们的参数和输出。
- 然后，用相同的数据或者不同的数据训练一个学生模型，但是不使用数据的真实标签，而是使用教师模型的输出作为目标。这时，可以使用温度（temperature）参数来调节教师模型输出的软硬程度，使其更容易被学生模型学习。
- 最后，用新的数据测试学生模型的性能，比较它和教师模型的差异。

# 知识蒸馏常用的损失：

1. **软目标损失（Soft Target Loss）**：
   交叉熵损失函数（加入温度调整）：

$$L_{\text{soft}} = -\sum_i \left( \sigma(\frac{z_i^T}{T}) \log(\sigma(\frac{z_i^S}{T})) \right)$$

   其中 $\sigma$ 表示 `softmax` 函数，$z_i^T$ 和 $z_i^S$ 分别表示教师和学生模型的 logits，$T$ 是温度参数。
2. **特征蒸馏损失（Feature Distillation Loss）**：
   均方误差（MSE）或L2损失函数：

$$L_{\text{feature}} = \frac{1}{N} \sum_{i=1}^{N} \| f_i^T - f_i^S \|^2$$

   其中，$f_i^T$和 $f_i^S$ 分别是教师和学生模型的特征表示，$N$ 是特征的总数。
3. **原始标签的硬目标损失（Hard Target Loss）**：
   标准的交叉熵损失函数：

$$L_{\text{hard}} = -\sum_i \left( y_i \log(\sigma(z_i^S)) \right)$$

   其中，$y_i$ 是真实标签的 one-hot 编码，$\sigma$ 是 `softmax` 函数，$z_i^S$ 是学生模型的 logits。

# BERT

**Bidirectional Encoder Representations from Transformers**(NLP): Transfomers的双向编码器表示

BERT的预训练任务包括"Masked Language Modeling"(**MLM**) 和"Next Sentence Prediction"(**NSP**) :

1. 掩码语言建模 (Masked Language Modeling, MLM)： 在MLM任务中，文本序列中的一些词语会被随机地掩盖或替换成特殊的标记（例如"[MASK]"），然后模型需要预测这些被掩盖的词语。这个任务帮助模型学习理解文本的上下文和语言结构。
2. 下一句预测（Next Sentence Prediction, NSP）： 在NSP任务中，模型需要判断两个句子是否是原始文本中相邻的句子。这个任务有助于BERT理解文本之间的逻辑关系和语境。

> 这两个任务都是无监督的

BERT在大规模文本上完成了预训练，且可以通过微调在各种监督学习任务中使用，如文本分类、问答等，并取得出色的性能。

> **微调**: 通常涉及有标签的数据，因此它是一种监督学习

- BERT模型通常有两个主要规模的版本，分别是 `BERT-Base` 和 `BERT-Large`。BERT-Base 包含约 110 万参数，而 BERT-Large 更大，包含约 3400 万参数。这使得它们在预训练阶段能够学习到非常丰富的文本表示。

---

# ViT

- *ViT B* 对应的就是*ViT*-Base，*ViT* L 对应的是 *ViT*-Large，*ViT* H 对应的是 *ViT*-Huge
- 直接将输入的图像分成固定大小的图块（通常是16x16像素的块），然后将这些图块展平为一维向量，并将它们串联成一个序列。这个序列将作为模型的输入，而不再是传统CNN中的图像矩阵。
- 使用自注意力

- 每个patch被encoder后映射到一个高维向量空间
- 多层注意力
- 包括分类头

---

# MAE

包括encoder和decoder

1. 输入数据：将包含缺失值或需要掩码的输入数据提供给模型。
2. 编码器：编码器将输入数据映射到潜在表示空间，通常是一个低维度的向量。编码器的任务是捕获输入数据的关键特征。
3. 解码器：解码器将潜在表示映射回原始输入数据的空间，以尝试重建输入数据。在这个过程中，掩码的部分通常被忽略，不参与重建

---

# Diffusion Model

参考
主要思想:

- 利用一个前向扩散过程，将真实数据逐步加入高斯噪声，直到变成纯噪声
- 利用一个反向扩散过程，从纯噪声逐步恢复出真实数据。
  整个过程可以看作是一种概率建模，通过学习数据的潜在分布来生成新的数据。

> **优点**

- 可以处理高维度和复杂的数据，比如图像和音频，而不需要复杂的网络结构或者特殊的损失函数。
- 可以利用大量的无标签数据进行无监督学习，而不需要额外的信息或者对齐。
- 可以生成多样化和逼真的数据，而不需要对抗训练或者正则化。

> **缺点:**

- 需要大量的计算资源和时间来进行训练和生成，因为它需要多次迭代扩散和反扩散过程。
- 需要合适的噪声水平和步长来控制扩散和反扩散的速度和效果，这可能需要人为的调整和实验。
- 难以控制生成数据的属性和特征，比如风格和内容，因为它没有明确的潜在空间或者条件变量。

---

# CLIP

在预训练的时候利用对比学习的方法对齐两个模型

---

# BLIP(Bootstrapping(双向) Language-Image Pretraining)

介绍: 理解和生成的统一，引入了跨模态的编码器和解码器，实现了跨模态信息流动，在多项视觉和语言任务取得SOTA

BLIP引入了编码器-解码器的多模态混合结构MED（Multimodal mixture of Encoder-Decoder），能够有效地进行多任务预学习和迁移学习。MED包括两个单模态编码器（Image Encoder, Text Encoder），一个以图像为基础的编码器（image-grounded text encoder）和一个以图像为基础的解码器（image-grounded text decoder）。

**该模型通过三个损失函数联合进行预训练：**

1. **图像-文本对比损失 ITC**（Image-Text Contrastive Loss）：针对图像编码器和文本编码器，通过正负图文对的对比学习，**来对齐图像和文本的潜在特征空间。**
2. **图像-文本匹配损失 ITM**（Image-Text Matching Loss）：针对以图像为基础的文本编码器，通过对图文匹配性进行二分类，**建模图文多模态信息的相关性。**

3. **语言建模损失 LM**（Language Modeling Loss ）：针对以图像为基础的文本解码器，通过交叉熵损失进行优化，**训练模型以自回归的方式生成目标caption**

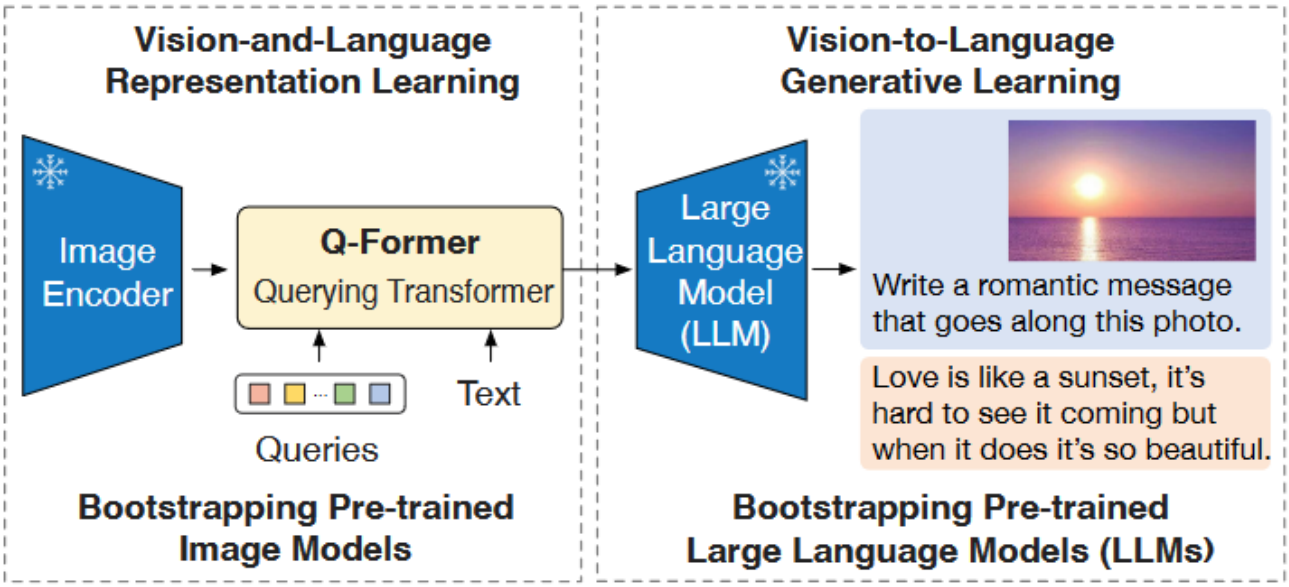
---

# BLIP-2

提出了一种借助现成的**冻结参数的预训练视觉模型**和**大型语言模型**的，高效的视觉语言预训练方法。

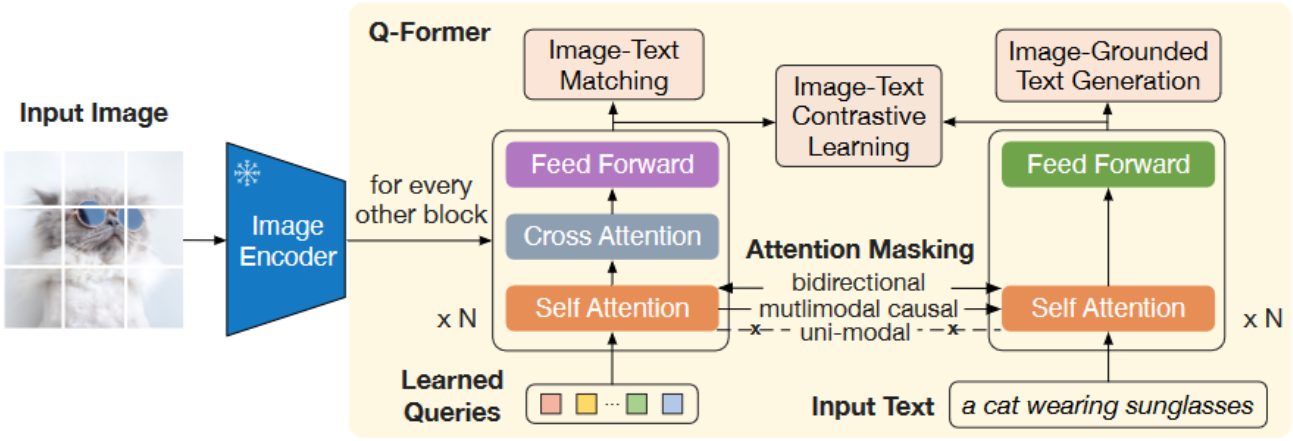
但是，简单的冻结预训练好的视觉模型的参数或者语言模型的参数会带来一个问题：就是视觉特征的空间和文本特征的空间不容易对齐。为了解决该问题，BLIP-2 提出了一个轻量级的 Querying Transformer (即为Q-Former)，该 Transformer 分两个阶段进行预训练。

## 训练过程




## 第一阶段(Left)

从冻结的视觉编码器中引导多模态学习-




## 第二阶段(Right)

从冻结的文本编码器中引导多模态学习。

# BLIP和BLIP-2的区别

- BLIP是一个端到端的模型，它使用了一个视觉编码器和一个语言编码器，然后在它们之间添加了一个交叉注意力层来对齐视觉和语言特征。BLIP的视觉编码器和语言编码器都是可训练的，它需要在大规模的视觉-语言数据集上进行预训练。
- BLIP-2仅仅使用一个轻量级的查询变换器（Q-Former）来桥接视觉和语言。Q-Former是**唯一可训练**的部分，视觉编码器和语言模型都保持冻结。BLIP-2的Q-Former分为两个阶段进行预训练，第一个阶段从**视觉编码器**中提取视觉-语言的表示，第二个阶段从**语言模型**中提取视觉-语言的生成能力。
- 所以BLIP-2是轻量级的，而BLIP大量级
- 而且BLIP-2可以直接使用任何两个模型，不需要对这些模型进行训练

---

# CLIP和BLIP-2的区别

> 简而言之就是CLIP在预训练的时候就直接对齐两种模态的数据。而BLIP-2是使用了一种技术来链接两个不同模态的模型，例如BERT和ViT。而且BLIP-2是不限于使用哪两个模型的。