Adapter in a multimodal model for Image caption task

王君豪

摘要

BERT、GPT3 等预训练语言大模型已经被证明在 NLP 领域可以取得非常好的效果。随着近两年多模态领域的研究逐渐成熟，越来越多的研究者开始关注多模态预训练大模型，例如最初的 ViL-BERT 以及后来 OpenAI 提出的 CLIP，再到近期可以以统一范式支持各种模态任务的 OFA，它们都在各种模态的下游任务中取得了相当不错的效果。但当前多模态大模型仍然面临着两个挑战。一是模型与应用任务的领域不适配，在领域中进行微调后，对原有的知识会出现灾难性遗忘的现象。二是对于规模越大的模型，参数就越多，在微调时所需要调整的时间也更长。而 NLP 领域中为了解决这个问题曾经提出过 Adapter 机制，其通过为网络增加专门应用于下游任务的参数，以避免在微调时对模型的所有参数进行修改，同时保留模型原有的知识及优秀性能。本文中，作者则将 Apdater 机制应用于多模态的模型当中，设计了多模态模型的轻量化适配方案。

关键词：multimodal；adapter；

1 引言

多模态预训练模型是现今深度学习领域的热门方向，与一般计算机视觉与自然语言处理的单模态模型的不同之处在于，多模态模型瞄准的任务不仅仅是图像分类或者是词性标注这种能够仅凭一种模态信息解决的问题，而是更接近现实世界，更复杂的多模态任务，比如物体定位任务，根据文本找到图像中的对应物体。面对更复杂的任务，多模态预训练模型也具备了更复杂的网络结构，多模态预训练模型博取百家之长，主流的多模态模型可以分为两类，分别是通过不同编码器模型处理不同模态信息的双流模型，如 CLIP[1]等。也有将多个模态信息拼接在一起，再输入到同一个编码器的单流模型，如 OFA[2]等。

为了追求卓越的性能，多模态模型的大小，训练规模也在不断剧增。例如 CLIP[1]就使用了 4 亿多对图文对，OFA[2] 中的 large 模型参数也达到了好几亿。大模型虽然带来了更好的结果，但是无论是推理时间还是训练时间都有了几何倍数的增长，这使得普通用户更难将大模型部署到实际应用。尤其是在当前常用的预训练微调范式当中，模型在实际应用到任务之前还需要经过在特定数据集微调的步骤，而微调需要对模型中的每个参数都做出更新调整，要想获得好的效果也必须经过多轮微调，这种代价让计算资源并不丰富的用户望而却步。

NLP 领域中为了解决这个问题，提出了以轻量和扩展性强闻名的 Adapter 方法。它只需要以一个较小的训练和存储代价就可以取得和全模型微调相当的结果。借鉴于 NLP 中的方法，这项工作尝试将 Adapter 应用于多模态模型 OFA[2] 中，在图像标注任务下进行训练，以期在最小化性能损失的情况下，降低训练代价，加快微调速度，让多模态大模型的部署更容易推广，同时保留模型的原有性能。

2 相关工作

2.1 多模态预训练模型

预训练方法基于廉价的标注数据，使模型学到较好的语义以及图像表示，然后再通过微调的方法将模型迁移至下游任务中使用。多模态数据是指相对于同一个描述对象不同的描述方式所获得的数据，而其中的每个视角叫做一个模态。将预训练方法应用到了多模态任务当中，则使模型能够从大规模数据的预训练中挖掘到不同模态之间的语义对应关系。多模态预训练模型在将图像和文本编码至单模态嵌入后，需要集成视觉和语言模态信息，而根据不同的信息集成方式可将模型分类为融合编码器，多编码器，以及混合编码器。

融合编码器融合编码器以文本和图像嵌入作为输入，然后通过自注意力模块或是交叉注意力模块来建模视觉与语言模态信息之间的交互。融合编码器又可分为单流架构和双流架构。单流架构模型将提取到的文本和图像嵌入进行拼接，然后输入到基于 Transformer 的编码器中，通过 Transformer 的自注意力机制实现视觉与语言信息之间的交互。VisualBert[3] 将文本和图片统一进行语义学习，在参数更少的情况下，达到了与双流架构中的 VilBERT[4] 同等的效果。许多的多模态预训练模型都仅使用图文对进行预训练，导致它们在应用于单模态任务时效果欠佳。基于此点，UNIMO[5] 在图文对的基础上融入单模态的文本和图片数据进行训练，大幅度提高了多项单模态和多模态任务的性能指标。因为单流架构假设两个模态之间的潜在关系是简单的，而将文本和图像特征直接拼接在了一起执行自注意力机制，一些工作认为它们可能忽略了各模态内的交互，因此使用了双流架构。双流架构采用交叉注意力机制来进行视觉与语言信息之间的交互，交叉注意力机制中 query 向量来自一个模态，而 key 和 value 向量来自另一个模态。一个交叉注意力层往往包含了从语言到视觉和从视觉到语言两个交叉注意力模块，实现了两个模态间的信息抽取和语言对齐。VilBERT[4]在跨模态信息交互的模块后面加入了两个 Transformer，使得各模态内部信息能够更好交互。ALBEF[6]先对图像和文本特征采用了单独的 Transformers，再输入到交叉注意力模块当中，使模型能够更好的解构模态内交互与模态间交互。

多编码器融合编码器都依赖于一个庞大的 Transformer 网络来进行视觉与语言信息间的交互，这会导致模型在某些多模态任务上速度太慢。多编码器放弃了 Transformer 中复杂的交叉注意力，分别使用单独的编码器来编码不同模态，然后采用点积等简单的方法来将图像特征和文本特征投影至相同的语义空间。CLIP[1]使用文本信息作为图像训练的监督信号，以文本与图像间的对比损失作为损失函数，在表征学习任务上表现出色。ALIGN 通过减少数据预处理工作获得了一个更大的图文对噪音数据集，基于简单的双编码器结构获得了 SOTA 的效果。

混合编码器根据经验表示，融合编码器在 VL 理解任务上表现更好，而双编码器在检索任务上的效果更好。为了融合两种结构之间的优点，FLAVA[7] 将经过了双编码器的不同模态信息拼接起来送到了融合编码器当中，以此获得多模态表示。Vlmo[8] 统一了双编码器和融合编码器在单个框架中，在 V-L 理解任务和图文检索任务当中都表现良好。FLAVA[7] 和 Vlmo[8] 都采用了类似于 UNIMO[5] 的训练策略，在预训练过程中加入了单模态数据，分别训练单模态的编码器。

2.2 Adapter

Adapter 方法的原理并不复杂，它是通过在原始的预训练模型中的每个 transformer block 中加入一些参数可训练的模块实现的。假设原始的预训练模型的参数为 ω，加入的 adapter 参数为 υ，在针对不

同下游任务进行调整时，只需要将预训练参数固定住，只针对 adapter 参数 υ 进行训练。通常情况下，参数量 υ«ω, 因此在对多个下游任务调整时，只需要调整极小数量的参数，大大的提高了预训练模型的扩展性和实用性。对于 adapter 模块的网络组成，不同文章中针对不同任务略有不同。但是比较一致的结论是，bottleneck 形式的两层全连接神经网络就已经可以满足要求。最早的 Adapter 模型采用的就是简单的两层全连接神经网络，称为 bottleneck Adapter[9]。Bottleneck Adapter 中采用了两个全连接层和一个非线性激活函数，其中两个全连接层分别进行了下采样和上采样操作。Adapter fusion[10]在 Adapter Finetune 的基础上，提出当下游存在多个任务的时候，使用两阶段的 Finetune，实现从多个 Adapter 产出信息中的选择和融合。这样模型实现了根据每个样本选择合适的 adapter 输出，综合了多任务的 adapter 信息。

3 本文方法

3.1 本文方法概述

本文基于 OFA 模型的基础上进行改进，OFA 是基于 Transformer 所建立的统一编码器解码器多模态模型。OFA 模型由图像嵌入，文本嵌入，Transformer 三个模块组成。而在引入 Adapter 机制后，在原有的每两层 transformer layer 中插入了一层 adapter layer。

3.2 OFA 模型总架构

OFA 的模型结构是统一的 transformer 编码器解码器结构，属于多模态模型中的单流结构，如图 1 所示。OFA 可以分成图像嵌入提取模块，文本嵌入提取模块，以及统一模态的编解码器。OFA 先通过 Resnet 提取图像特征，然后通过 BPE 分词法以及线性层投影获得文本特征。之后将文本特征及图像特征拼接在一起，输入到传统的 Transformer 结构当中。

OFA 将多模态及单模态的理解和生成任务统一到 1 个简单的 Seq2Seq 生成式框架中，OFA 执行预训练并使用任务指令进行微调，并且没有引入额外的任务特定层进行微调。

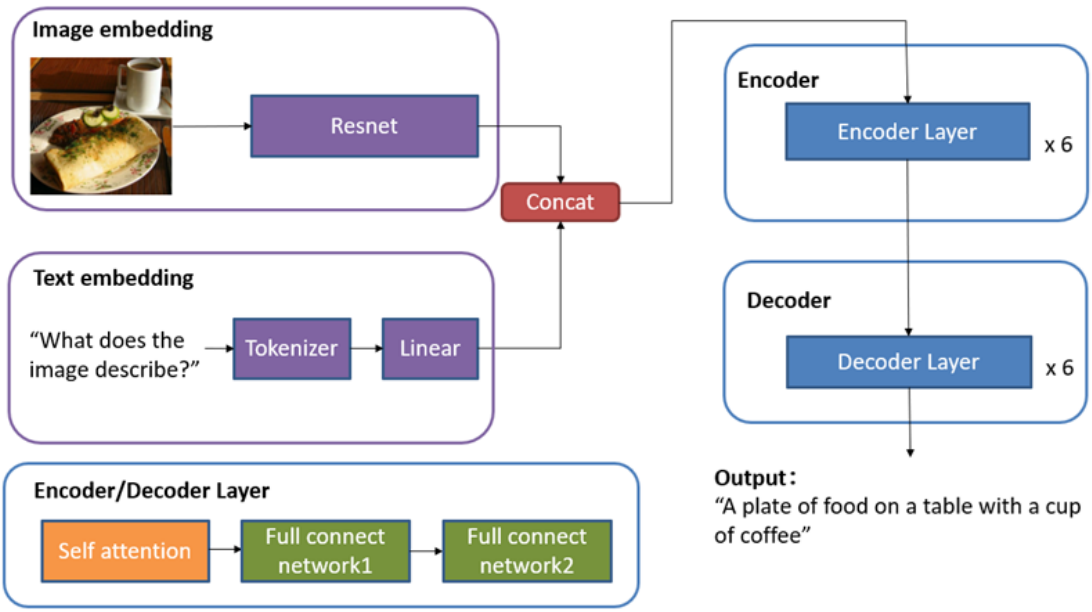

图 1: 模型结构

3.3 Transformer layer

OFA 模型中的 transformer layer 采用了最经典的 transformer layer 的设置，即由一层多头自注意力层和两层全连接网络组成。其中各层的维度数量与模型规模有关。

3.4 Adapter layer

本文中所使用 Adapter 设置借鉴于最经典也是最简单的 Bottle Neck Adapter，其结构如图 2 所示。在编码器和解码器的每个 transformer layer 当中两层全连接层后面插入 Adapter layer。在每个 adapter 模块中，先将输入做一个降维的映射。经过一个非线性激活层后，再将特征矢量映射回原始的维度。Bottle neck Adapter 的两个特性分别是残差连接和下投影及上投影层。

（1）有了残差连接之后，通过较小的参数初始化 adapter，就可以使得输出接近于恒等映射。可以保证在刚开始时模型依然表现良好。

（2）Adapter 内部进行了先降维再升维，而不是直接使用两层输入输出相同的全连接网络，也是为了减少参数量考虑。

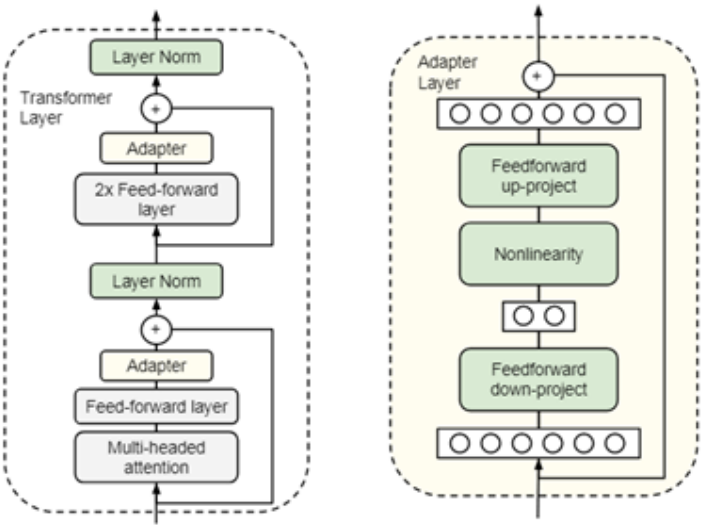

图 2: Adapter layer

3.5 Adapter 多模态模型

本文在 OFA 模型的基础上，在各 transformer layer 中加入了 Adapter layer。选择使用 OFA 模型的理由有如下三点：

（1）OFA 模型的论文发表于 2022 年 3 月，是当今多模态大模型领域中的前沿技术，在多项单模态以及多模态下游任务当中取得了 SOTA 结果。

（2）OFA 模型的规模较大，参数较大，即使是 OFA-Base 也有 1 亿多参数，在大模型上进行改进更能体现 Adapter Layer 在应用到下游任务时的加速作用。（3）OFA 模型采用了统一的 Transformer 结构，将图像信息和文本信息映射到了统一的嵌入空间，然后再输送到 Transformer 当中。在统一的基础上进行 Adapter 的映射操作后，能够得到融合模态的语音信息。同时，本文所采用的 Adapter layer 是 Bottleneck Adapter，选用 bottleneck adapter 的原因在于

（1）bottleneck adapter 虽是最简单的 Adapter 结构，但在许多下游任务中也有不俗的发挥

（2）bottleneck adapter 的简单结构使得它在调整参数量上远小于微调范式，在训练速度上提升显著。本文所提出的模型架构图如图 3 所示。在预训练完毕后，冻结模型中原有的特征提取模块（包括

图像特征提取的 Resnet 以及文本特征提取中的线性层），以及冻结原有 transformer layer 中的自注意力模块以及全连接层，并在每一层 transformer layer 的最后加入 Adapter layer。Adapter layer 与 Bottleneck Adapter 的设置一样，分为上投影层和下投影层以及一个激活函数。

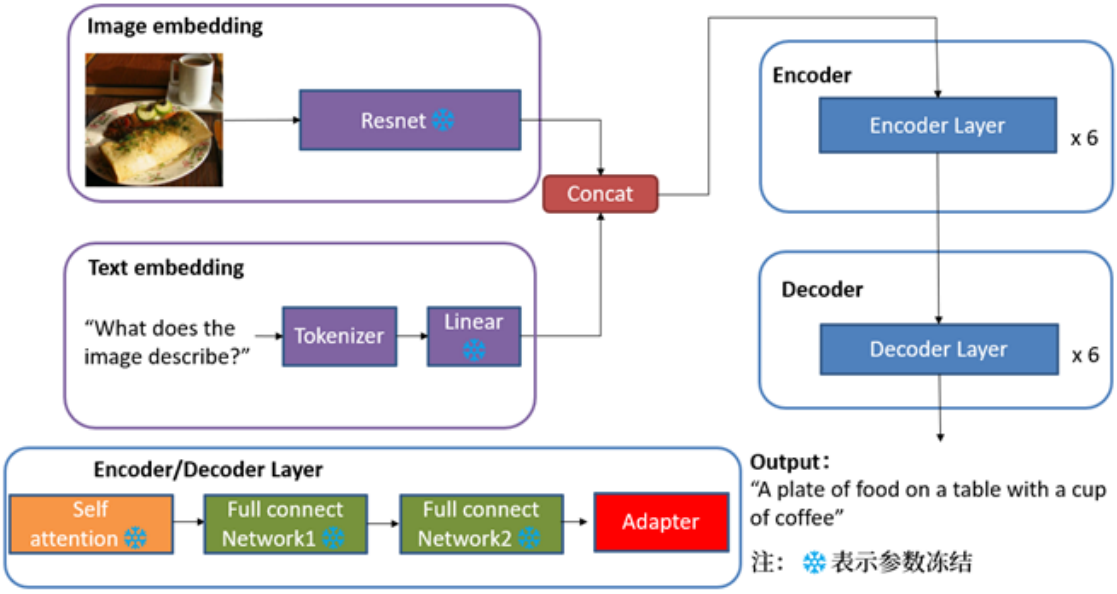

图 3: Adapter-ofa 模型

4 复现细节

4.1 与已有开源代码对比

在开源代码的基础上进行了改进，这项工作主要是在原有的模型中插入了 Adapter layer，并且重新基于图像标注任务，在 COCO2014 及 COCO2017 数据集上进行了重新的训练和测试，得出分析结果。以下是本项工作中的核心代码。如图 4 为 Adapter layer 的类别定义，其继承自 torch 的 nn.module 类，从而可以直接插入到原有的模型当中而不需要对整体框架进行更改。图 5 图 6 分别是 Adapter layer 的前向传播函数及插入位置。

```python
k1tano
class Adapter_Layer(torch.nn.Module):
    k1tano
    def __init__(self,
                 d_model=None,
                 down_size=None,
                 dropout=0.0,
                 init_option="bert",
                 adapter_scalar="1.0"):
        super().__init__()
        self.n_embd = d_model
        self.down_size = down_size


        if adapter_scalar == "learnable_scalar":
            self.scale = nn.Parameter(torch.ones(1))
        else:
            self.scale = float(adapter_scalar)

        self.down_proj = nn.Linear(self.n_embd, self.down_size)
        self.non_linear_func = nn.ReLU()
        self.up_proj = nn.Linear(self.down_size, self.n_embd)

        self.dropout = dropout
        if init_option == "bert":
            self.apply(init_bert_weights)
        elif init_option == "lora":
            with torch.no_grad():
                nn.init.kaiming_uniform_(self.down_proj.weight, a=math.sqrt(5))
                nn.init.zeros_(self.up_proj.weight)
                nn.init.zeros_(self.down_proj.bias)
                nn.init.zeros_(self.up_proj.bias)
```

图 4: init 函数

```python
    k1tano
    def forward(self, x, add_residual=True, residual=None):
        residual = x if residual is None else residual

        down = self.down_proj(x)
        down = self.non_linear_func(down)
        down = nn.functional.dropout(down, p=self.dropout, training=self.training)
        up = self.up_proj(down)
        up = up * self.scale
        if add_residual:
            output = up + residual
        else:
            output = up

        return output
```

图 5: 前向传播函数

```
k1tano
def __init__(self, args, drop_path_rate=0.0, use_adapter=False, adapter_dim=200):
    super().__init__()
    self.args = args
    self.use_adapter = use_adapter
    self.embed_dim = args.encoder_embed_dim
    if use_adapter:
        self.adapter = Adapter_Layer(d_model=self.embed_dim, down_size=adapter_dim)
```

图 6: Adapter layer 的插入位置

4.2 实验环境搭建

1. 安装环境的所需包

pip install -r requirements.txt

2. 下载 COCO2014 及 COCO2017 数据集

3. 运行 run script 文件夹下的命令进行训练及测试。

4.3 创新点

本项工作在多模态模型的网络结构中加入了 Adapter layer，并且重新进行了微调。在预训练完毕后冻结所有除了 Adapter layer 的参数，从而使得在针对下游任务的微调过程中仅仅只有 Adapter layer 的参数改变了，这样做的目的一是为了节省微调的时间，二是为了防止原有的模型参数在微调之后过拟合下游任务数据集，而损失了原有的知识。

5 实验结果分析

从表 1 可以看出，由于微调参数量的不足，Adapter 结构的精度相比于在原有的架构上进行为微调，性能上还是会有少许差异。这是由于可供学习的参数较少，模型的拟合程度也不够。但是由表 2 中的参数量比较中更能得知，在 Adapter 方法中需要调整的参数仅仅约等于为 finetune 方法中的百分之二，因此，速度上的提升还是非常显著的。

Method	B@4	M	C	S
finetunning	42.40%	31.50%	142.2%	24.50%
Adapter	39.38%	28.16%	137.5%	22.30%

表 1: 数据集上结果指标对比

Method	size
finetunning	180M
Adapter	3.68M

表 2: 参数量对比

6 总结与展望

本工作针对大模型在下游任务微调耗时费力的问题，在多模态模型 OFA 上加入了轻量化且易于部署的 Adapter 结构，在对模型性能影响较小的情况下节省了训练所需的时间和空间，让多模态大模型可以在下游任务上更快应用。但也有不足之处，比如没有没有尝试多任务 adapter 的组合，也没有尝试将同个 adapter 迁移到其他任务当中，研究不同任务 adapter 之间的交互影响。

参考文献

[1] RADFORD A, KIM J W, HALLACY C, et al. Learning transferable visual models from natural language

supervision[C]//International conference on machine learning. 2021: 8748-8763.

[2] WANG P, YANG A, MEN R, et al. Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework[J]. arXiv preprint arXiv:2202.03052, 2022.

[3] LI L H, YATSKAR M, YIN D, et al. Visualbert: A simple and performant baseline for vision and language[J]. arXiv preprint arXiv:1908.03557, 2019.

[4] LU J, BATRA D, PARIKH D, et al. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks[J]. Advances in neural information processing systems, 2019, 32.

[5] LI W, GAO C, NIU G, et al. Unimo: Towards unified-modal understanding and generation via cross-modal contrastive learning[J]. arXiv preprint arXiv:2012.15409, 2020.

[6] LI J, SELVARAJU R, GOTMARE A, et al. Align before fuse: Vision and language representation learning with momentum distillation[J]. Advances in neural information processing systems, 2021, 34: 9694-9705.

[7] SINGH A, HU R, GOSWAMI V, et al. Flava: A foundational language and vision alignment model [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 15638-15650.

[8] BAO H, WANG W, DONG L, et al. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts[J]. arXiv preprint arXiv:2111.02358, 2021.

[9] HOULSBY N, GIURGIU A, JASTRZEBSKI S, et al. Parameter-efficient transfer learning for NLP[C]//International Conference on Machine Learning. 2019: 2790-2799.

[10] PFEIFFER J, KAMATH A, RÜCKLÉ A, et al. AdapterFusion: Non-destructive task composition for transfer learning[J]. arXiv preprint arXiv:2005.00247, 2020.