黄品溢个人学习笔记

Title 2022.07.31 学习笔记

Tips 论文复现及学习笔记

学生姓名 黄品溢

指导教师 王磊

学　　院 计算机学院

目　　录

第一章　问题整理与记录

第二章 OpenCV 函数学习记录

2.1 关于每个 image 的知识

通过 *cv2.imread* 读取的每个 image 都是一个 np.array 数组。

通常其形状为 *(height, width, 3)*，一定要注意 height 位于 width 之前。

2.2 关于 *cv2.circle* 函数

cv2.circle(img, center, radius, color, thickness)

其中，一定要注意，center 接收的坐标为 (width, height)，这与一般的坐标是反过来的。

2.3 列表元素相加减

列表对应元素相加减，可以使用 zip 函数进行。如 a 和 b 是两个同形状列表：

result = [i - j for i, j in zip(a, b)]

第三章 论文阅读及思考记录

以下为阅读过的论文，及其主要贡献的个人思考和总结。

3.1 *Short and Long Range Relation Based Spatio-Temporal Transformer for Micro Expression Recognition(2021.12)* [1]

3.1.1 论文的主要贡献和主要思想

个人总结，这篇论文的主要贡献和主要思想如下：

1. 将视频中每一帧与微表情起始帧计算光流场，而非任意两连续帧。
   计算相邻帧之间的光流场时，在前半部分有相似的光流，后半部分则强度相似，方向相反。
   而用这里提出的方法，发现光流场总是沿同样的方向，只是前半部分的强度逐渐递增，后半部分的强度逐渐递减。这导致了与每个微表情相关的更稳定和可区别的特征。
2. 首个完全摒弃 CNN 进行特征提取的 MER 神经网络模型。其认为，CNN 只能在固定窗口大小中进行短期空间特则提取，由此无法学习到同样十分重要的长期空间特征。而 Encoder of Transformer 中使用到的 Multi-head Self-attention Mechanism（MSM）自注意力机制，可以很好的学习到短期和长期空间特征，由此其完全摒弃了 CNN 的使用。
3. 时间聚合：使用了 LSTM Aggregator 进行聚合，并与 Mean Aggregator 进行了对比。聚合函数确保了 Transformer 模型可以被训练并应用于每一帧的空间特征集，然后处理每个样本中各帧之间的时间关系。

3.1.2 数据预处理阶段

首先是人脸图像裁剪阶段：其在表情峰值顶点帧提取人脸 68 个特征点，然后根据公式计算人脸的大小，并以第 30 号点为中心（标号以 0 开始），截取一帧中的这一部分图像。由此可以保证几乎整张人脸都被囊括在这个裁剪的图像内部。

$$s = (y_{apex[8]} - y_{apex[19]}) + (y_{apex[8]} - y_{apex[57]}) \tag{3.1}$$

其中，s 表示截取图像的长和宽；y 轴表示的是图像的 height 轴，注意在图像当中，height 轴向下为正方向。

接下来是时空插值阶段：其在原始视频中应用了帧插值方法，有效地进行了数据增强。

与简单的线性插值不同，这里提出了一种新的插值方法，这种方法在光流方面更平滑。这里使用了 RIFE（*Real-time Intermediate Flow Estimation*）方法[2]，这是一个端到端的可训练神经网络，它能很快和直接的估计中间光流。

传统的 RIFE 在连续的两帧之间进行插值，得到一个中间帧。而本文中，通过递归插值的方法获取多个中间帧。具体的说，操作步骤如下：

- 首先以连续的两帧 I_0, I_1 作为输入，得到一个中间帧 $\hat{I}_{0.5}$，即 t=0.5 时刻的帧；
- 其次递归使用 RIFE，对 I_0 和 $\hat{I}_{0.5}$ 进行插值，得到 $\hat{I}_{0.25}$。如此反复；
- 另外有一点，本文中优先对顶点帧附近进行插值，最后得到的插值队列可以被表示为：$\{\hat{I}_{a-0.5}, \hat{I}_{a+0.5}, \hat{I}_{a-1.5}, \hat{I}_{a+1.5}, ..., \hat{I}_{o+0.5} \ or \ \hat{I}_{f-0.5}\}$，这里的 a, o, f 分别表示顶点帧、起始帧（onset frame）和偏移帧（offset frame）。

值得注意的是，在 CASME II 和 SAMM 数据库中，顶点帧的位置已经被明确指出了；而在 SMIC-HS 数据库中则没有。因此采用了 SMIC-HS 的中间帧作为该样本的顶点帧。

3.1.3 ViT（*Vision Transformer*）的应用

神经网络初期使用了一个 ViT 结构，该结构的示意图如下所示。

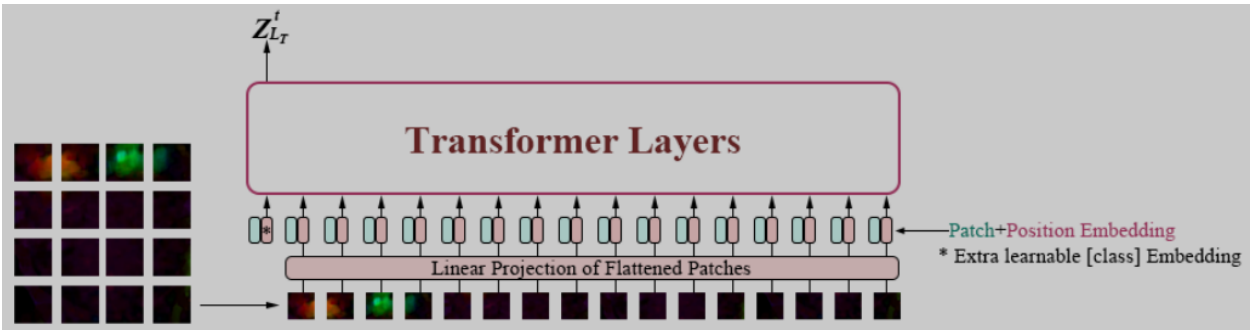


图 3.1 ViT 模型结构示意图

对模型的理解：

由于 Transformer 是用于 NLP 任务的结构，其输入需要是一个词结构。而对于 MER 来说，其输入通常是一个视频（并从中提取每一帧），由此便需要将图像转换为词结构。这里使用到的方法是将图片分割成若干个小块 patch，这里的每个小块就相当于句子里的每个词。而 *Patch Embedding* 就是把每个 patch 经过一个全连接网络压缩成一定维度的向量，即词向量。

而这里的词嵌入过程，就是图中的 *Linear Projection of Flattened Patches* 部分。

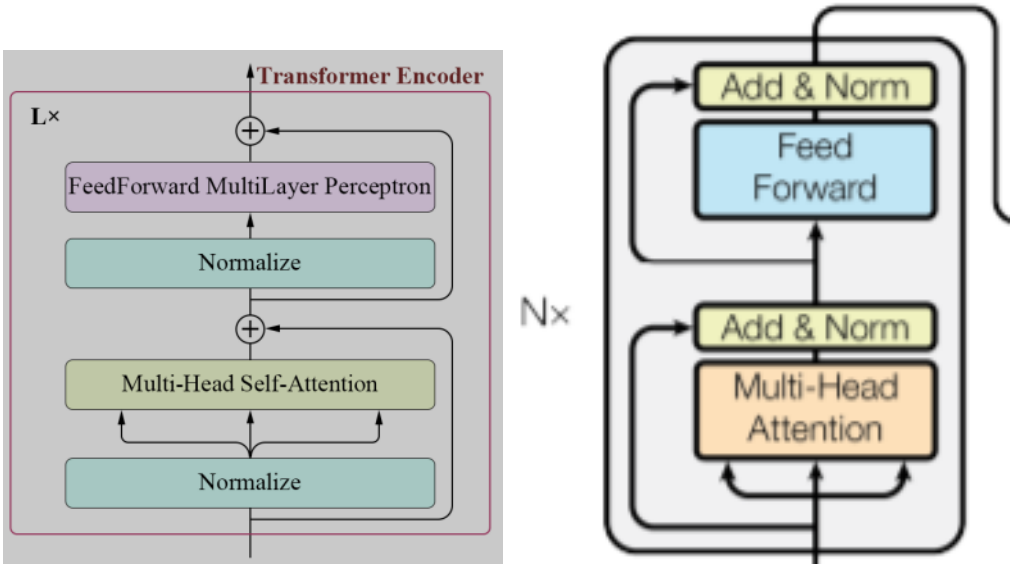
具体来说，就是使用了一个 kernel_size = stride = patch_size 的 nn.Conv2d 进行压缩，并且 in_channels = C，图像通道数（一般为 RGB 图像，即 C = 3），out_channels = embed_dim，即词向量的维度。

此外，为了表示每个 patch 的位置信息（即每个词在句子中的位置信息），还需要加入一个 *Position Embedding* 位置嵌入。此处使用到的位置信息通常是可训练的位置信息，参数

通过训练来进行调整（具体怎么实现暂时还未了解，似乎是一个 nn.Embedding 模块）。

同时，ViT 添加了一个 cls_token（*classification token*），可以这样理解：其他的 embedding 表达的都是不同的 patch 的特征，而 cls_token 是要综合所有 patch 的信息，产生一个新的 embedding，来表达整个图的信息。这个 token 也是可学习的。

并且，这里的 Encoder of Transformer 过程与传统的 Encoder 略有不同，其结构图对比如下所示。



(a) Encoder of ViT (b) Encoder of original Transformer

图 3.2 两种不同的 Encoder 结构的对比

由上图可以发现，两者关于归一化操作和 MSM、FFN（*Feed-Forward Network*）的顺序恰好相反。

并且，ViT 中使用**DropPath 代替了传统的 Dropout**，其主要效果是其效果是将深度学习模型中的多分支结构的子路径随机“删除”，可以防止过拟合，提升模型表现，而且克服了网络退化问题。

在向前传播的时候，让神经元以一定概率停止工作。这样可以使模型泛化能力变强，因为神经元会以一定概率失效，这样的机制会使结果不会过分依赖于个别神经元。训练阶段，以 keep_prob 概率使神经元失效，而推理的时候，会保留所有神经元的有效性，因此，训练时候加了 dropout 的神经元推理出来的结果要除以 keep_prob。

在 Dropout 中也有同样的体现：未被置零的元素需要除以（1 - dropout），由此来保证整体的期望在 Dropout 前后不会发生变化。

两者的主要区别有：

1. Dropout 是以一定比例，将某个向量内部的元素置为 0；

2. DropPath 是以一定的概率，将神经网络中的一条路径“删除”。

在传统的 Transformer 结构论文代码[3] 重写中，我自己曾写过其中的 Dropout 代码，其都是使用 nn.Dropout(dropout=0.5) 来创建的。而相对应的，ViT 中的 dropout 位置相同，而其需要改成使用 DropPath 进行使用即可。

有一点很奇怪，有两个不同的 github 仓库，均对 ViT 模型进行了实现，且均使用 PyTorch 进行编写。但两者实现 Dropout 的方式不相同，其一 直接使用 nn.Dropout，而 另外一个则使用了 DropPath 的方式。

而 官方实现中则是使用了 nn.Dropout 版本。这是否意味着两种方式均可实现，而从实际效果来说没有较大区别？

3.1.4 *Temporal Aggregation* 时空聚合

时空聚合中使用了 LSTM Aggregator 进行聚合操作，并将其与 Mean Aggregator 进行了对比。对比后发现 LSTM Aggregator 的效果好于 Mean Aggregator。其具体的使用步骤如下，以一层 LSTM Aggregator 为例。

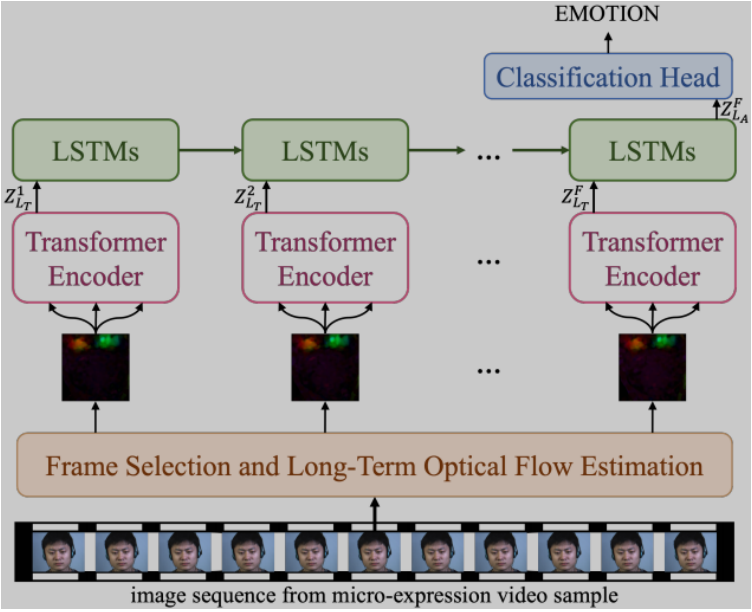


图 3.3 这里指应该是画的只有一层 LSTM Aggregator 的情况

这里的时空聚合的输入是每一帧经过一系列操作后，由 Encoder 输出的结果，即一个视频的帧数即为 LSTM 迭代的次数。图中画出了一系列的 LSTMs，实际上就只是一个 LSTM 层被数次迭代。

而最后的输出 $Z_{L_A}^F$ 的含义是，经过了 F 帧数和 L_A 个层数。这里的 L_A 表示 $L_T + L_{LSTM}$，其中 L_T 表示 Transformer 的总层数，L_{LSTM} 表示 LSTM Aggregator 的总层数。

聚合函数确保了文中的 Transformer 模型可以被训练和应用于每一帧的空间特征集，然后处理每个样本中的帧之间的时间关系。由于 LSTM 可以学习时间关系，因此帧间的时序

关系就被学习了。

并且很重要的一点是，与其他通过改变视频帧顺序，来增强数据库的方法不同，这里的聚合并没有改变微表情视频样本的时序关系。

参考文献

[1] L. Zhang, X. Hong, O. Arandjelovic, and G. Zhao, "Short and long range relation based spatio-temporal transformer for micro-expression recognition," *arXiv preprint arXiv:2112.05851*, 2021.

[2] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, "Rife: Real-time intermediate flow estimation for video frame interpolation," *arXiv preprint arXiv:2011.06294*, 2020.

[3] L. Lei, T. Chen, S. Li, and J. Li, "Micro-expression recognition based on facial graph representation learning and facial action unit fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1571–1580, 2021.