周报（2023. 3.8-2023.3.14）姓名：孙瑞阳

每日小结

	周一	周二	周三	周四	周五
早	跑 ConvTrans 上课	跑 ConvTrans	跑 CBAM，上课	上课	跑 ConvTrans
中	跑 CBAM，上课	论文阅读	论文阅读	跑代码，上课	跑 ConvTrans
晚	上课	上课	组会	ConvTrans 学习	跑 GPT

注：简单表述当前时间段工作，如看文献 1，整理数据等

科研详情

文献阅读

文献1

题目： Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting

作者：Shiyang Li，Xiaoyong Jin，Yao Xuan，Xiyou Zhou，Wenhu Chen，Yu-Xiang Wang，Xifeng Yan

出处：NLPS 2019

方法：

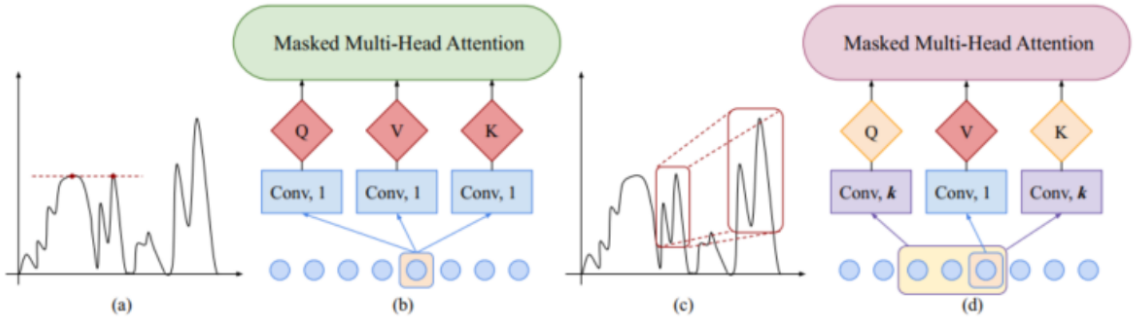
本文充分利用了 Transformer 的优势，并在 Transformer 的基础上改进了 Attention 的计算方式以适应时序数据

本文针对 Transformer 的两个主要弱点提出**卷积自注意**：

（1）局部不可知论：规范 Transformer 架构中的逐点点积自我注意对局部上下文不敏感，这会使模型容易出现异常在时间序列中；

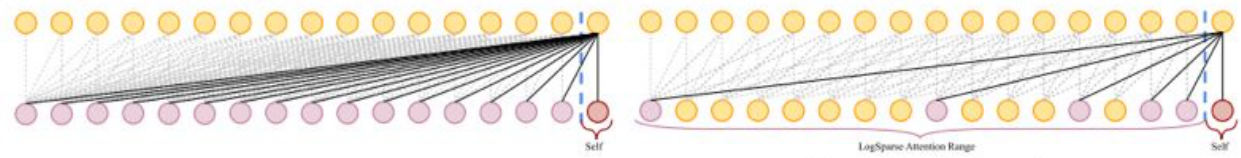
(2) 内存瓶颈：规范 Transformer 的空间复杂度随序列长度 L 二次增长，使得直接建模长时间序列不可行。

为了解决这两个问题，首先提出**卷积自注意**，通过因果卷积产生查询和键，以便将局部上下文更好地纳入注意机制。然后提出了只有 O(L(log L)2) 内存成本的 LogSparse Transformer，提高了在受限内存预算下具有细粒度和强长期依赖性的时间序列的预测准确性。



本文使用 kernel 大小 k 的因果卷积（causal convolution）（步幅为 1）将输入（使用适当的 paddings）转换为 queries 和 keys。因果关系会确保当前位置永远无法获得未来的信息。通过使用因果卷积，生成的 queries 和 keys 可以更好地感知局部上下文，从而通过局部上下文信息（如局部形状）来计算它们的相似性，而不是点状值，这有助于准确预测。当 k=1 时，卷积自注意退化为规范自注意，因而可以看作是一个泛化过程。

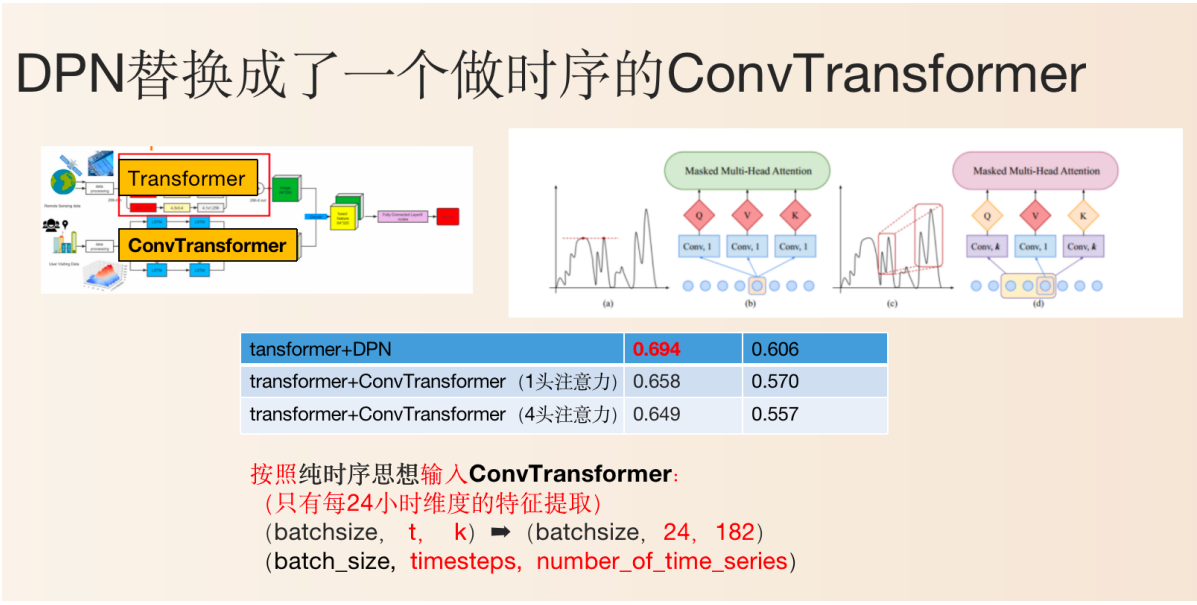
论文作者们认为引入某种程度的稀疏性，不会显著影响性能，反而为模型带来了处理具备细粒度和强长期依赖的长时间序列的能力。为了使得最终每个点都能接触到它的所有历史值的信息，所以便提出了 LogSparse 的设计，通过**堆叠多个自注意力层**来实现这个目的，如下图所示：



(a). Full Self Attention　　(b). LogSparse Self Attention

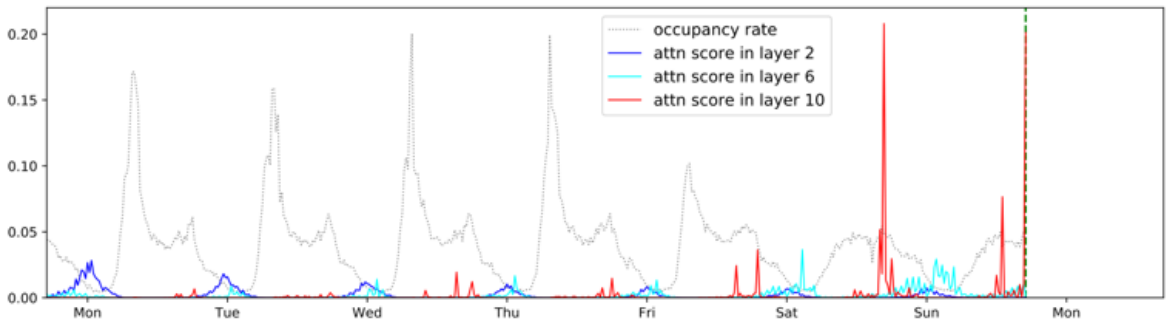
启发：

1. 在自己的网络里**复现**了该 ConvTransformer 网络，替换了 DPN 网络抓取时序，效果有 0.65，和 GRU 效果差不多



2. 支持并行，训练得更快。基于 RNN 的模型中每一个隐状态都依赖于它前一步的隐状态，因此必须从前向后必须逐个计算，每一次都只能前进一步。而 Transformer 没有这样的约束，输入的序列被并行处理，由此带来更快的训练速度。

3.觉得作者的这个图和分析特别好，吻合我的数据特点：针对 Transformer 的存储瓶颈问题，文中引入了 LogSparse 机制，原始 Transformer 在交通数据集上训练学习得到的注意力得分分布情况：



可以看到该模型共 10 层，图中蓝色、青色、红色的线分别是第 2，6，10 层的注意力得分，灰色的线为原始数据。注意到：**不同层对不同频率信息的关注度不同**

- **第 2 层（蓝色）倾向于学习每一天的模式**
- **第 6 层（青色）则更关注周末的影响**
- **而第 10 层（红色）对最近的时刻（邻近预测点）关注较高。**

文献2

题目：Conformer: Local Features Coupling Global Representations for Visual Recognition

作者：Zhiliang Peng, Wei Huang, Shanzhi Gu, Lingxi Xie, Yaowei Wang, Jianbin Jiao, Qixiang Ye

出处：arxiv 2021

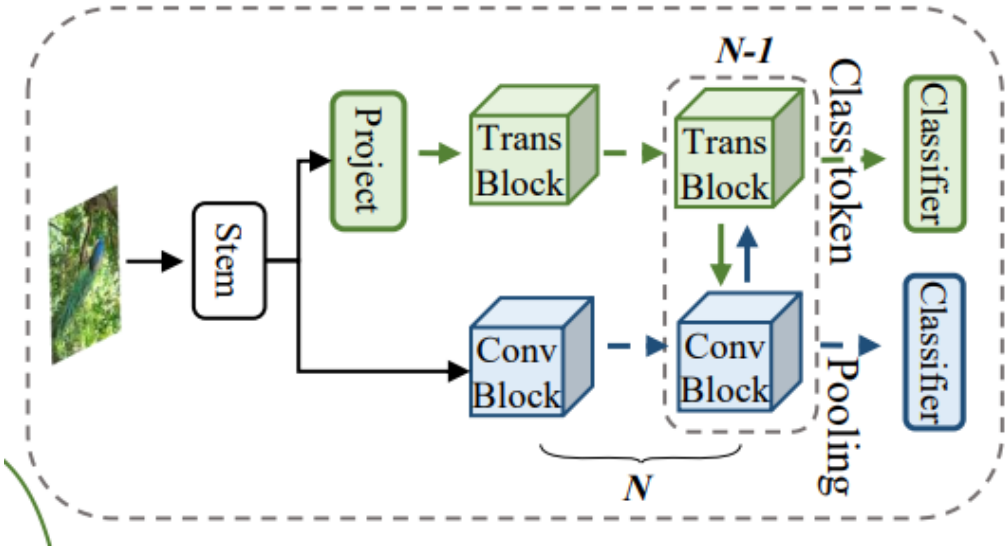
方法：

viT 后很多工作都是想办法把 transform 和 convolution 结合起来，希望同时享受各自的优点。**本文的想法是：**做两个分支，分别是卷积分支和 transformer 分支，然后并行的同时还相互补充。

这篇文章主要贡献：

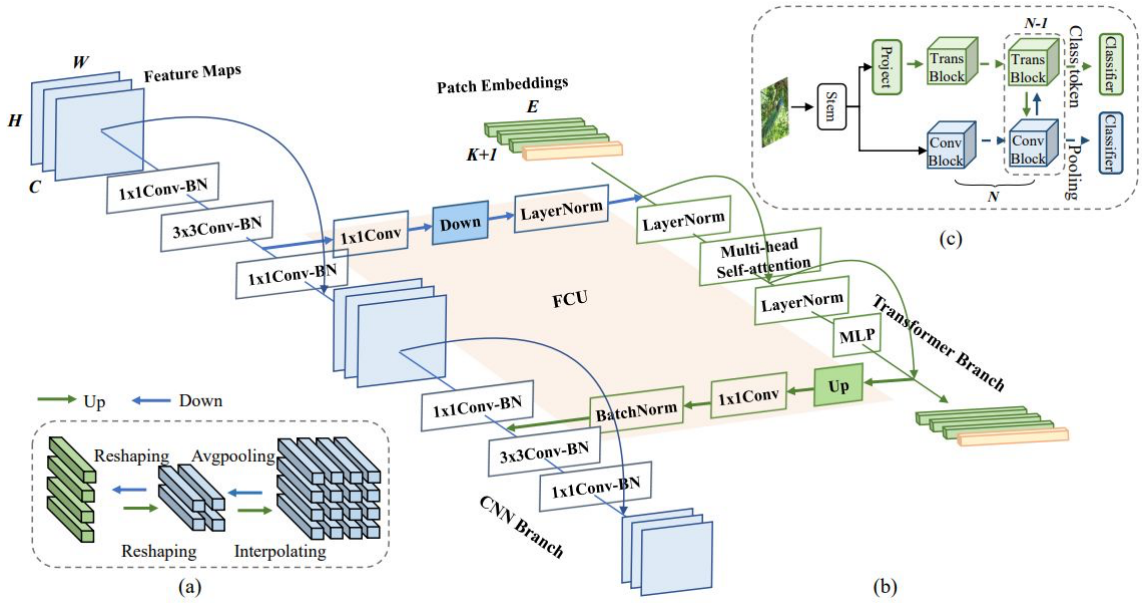
1. 给出 dual 网络结构，最大化保留两个 level 特征。
2. 提出 Feature Coupling Unit(FCU),交互的融合两个 style 的特征

conformer 的想法来源于 Feature Coupling Unit(FCU),他能让不同分辨率的局部和全局表征，以一种**交互的方式进行融合**。 结构设计上，Conformer，选取 concurrent 结构来最大程度的保留全局和局部特征。实验上，效果有一定提升。



在 conformer 中，连续地把来自 transformer 分支的全局语义丢入 CNN 分支来扩大全集感受能力。类似的，把来自 CNN 的局部特征又传入 patch embedding，给 transformer 提供细节信息。（**就是加上一个分支路径，让特征交替的过 CNN 和 transformer**）

具体的，Conformer 由 stem 模块，dual 分支，FCU 桥梁，两个分类器（fc 层）。stem 模块，为一个步长 2 的 7*7 卷积，后接一个步长 2 的 3*3 最大值池化。stem 是用来提出初始的局部特征（边缘和纹理），把初步处理的特征丢入 dual 分支。注意，分支里的 blocks 都是 N 个的堆叠。接着，从第二个 block 开始，使用 FCU,因为两分支初始的特征都是相同的。



启发：

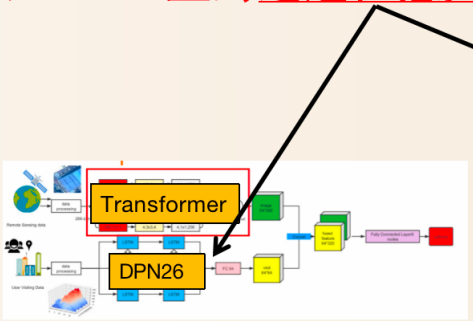
1. 我做实验时候发现，单独的 transformer 很难训练，很容易掉到局部最优坑里，看的很多论文都是得在 imagenet 上进行预训练，然后再搬到具体任务上。但是很多论文都是给 transformer 结构加入一点卷积，效果就十分好，应该是由于卷积特性帮 transformer 进行了补充，这样的话，融合结构可能确实会更好。
2. 主要是对比文献 1 的纯时序 ConvTransformer，看看图像网络能不能也用到这样的卷积结合 Transformer

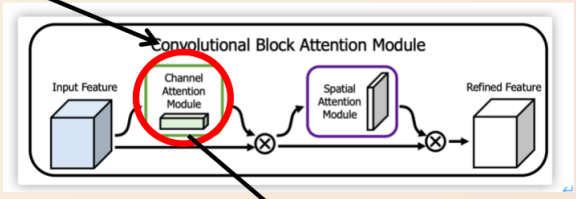
工作进展

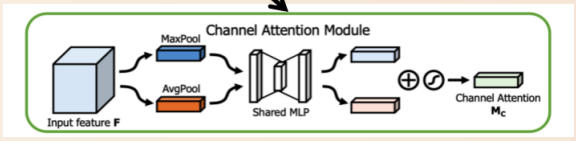
1：阅读文献；

2：URFC **实验结果：**

- 加CBAM里的<u>通道注意力</u>在DPN后面，0.695（原来0.694）





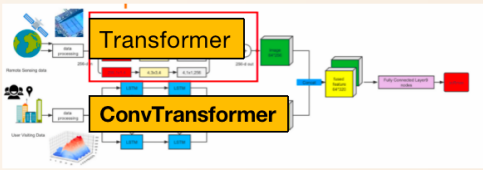


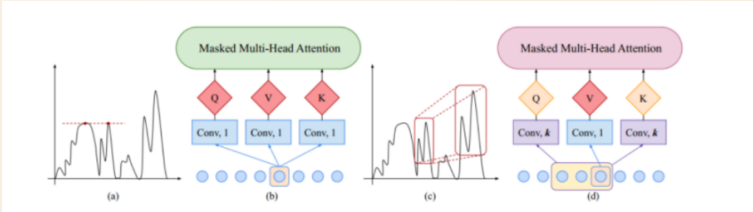
tansformer+DPN+CBAM通道部分	0.695	0.607
transformer+DPN+完整CBAM	0.685	0.591
tansformer+DPN	0.694	0.606

空间注意力在时序上表现都很差，通道注意力有微小提升

把文献 1 的 ConvTransfomer 加入我的网络跑了：之前 spatial temporal Attention 代码的实现是有问题的，完全按照图像思路走的，换成时序做，效果没降低

DPN替换成了一个做时序的ConvTransformer

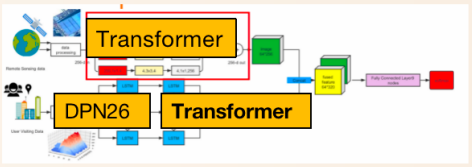


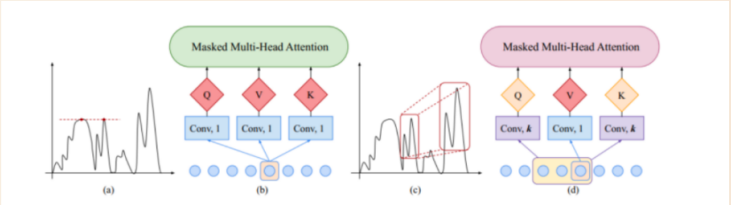


tansformer+DPN	0.694	0.606
transformer+ConvTransformer（1头注意力）	0.658	0.570
transformer+ConvTransformer（4头注意力）	0.649	0.557

按照纯时序思想输入**ConvTransformer**：
（只有每24小时维度的特征提取）
（batchsize， t， k）➡（batchsize，24，182）
（batch_size，timesteps，number_of_time_series）

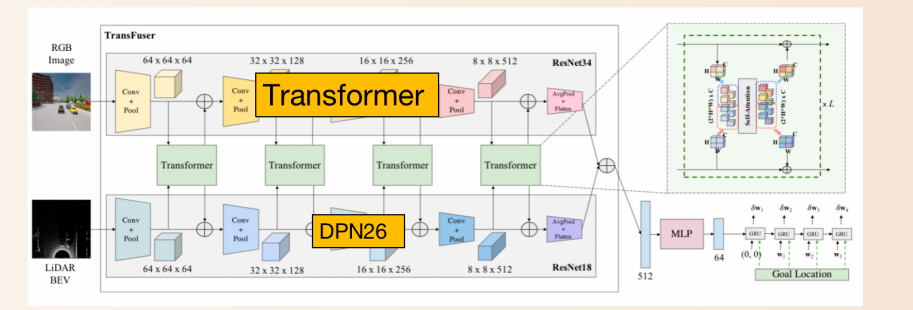
DPN后加了简单时序Transformer（相当于自注意力？）



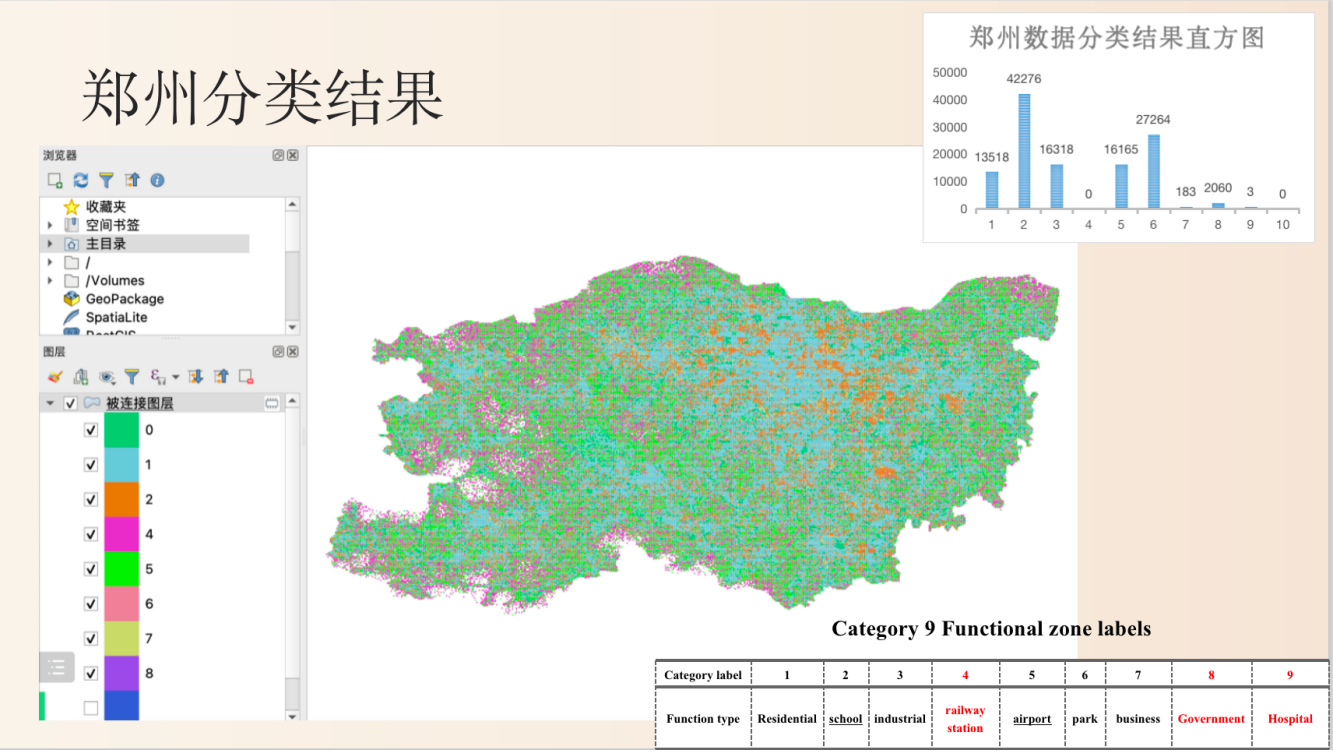


tansformer+DPN	0.694	0.606
transformer+ConvTransformer（1头注意力）	0.658	0.570
transformer+ConvTransformer（4头注意力）	0.649	0.557
tansformer+DPN+Transformer	0.670	0.576

3. 在实现 GPT 结构的层间嵌入，代码没改完，改的比较辛苦



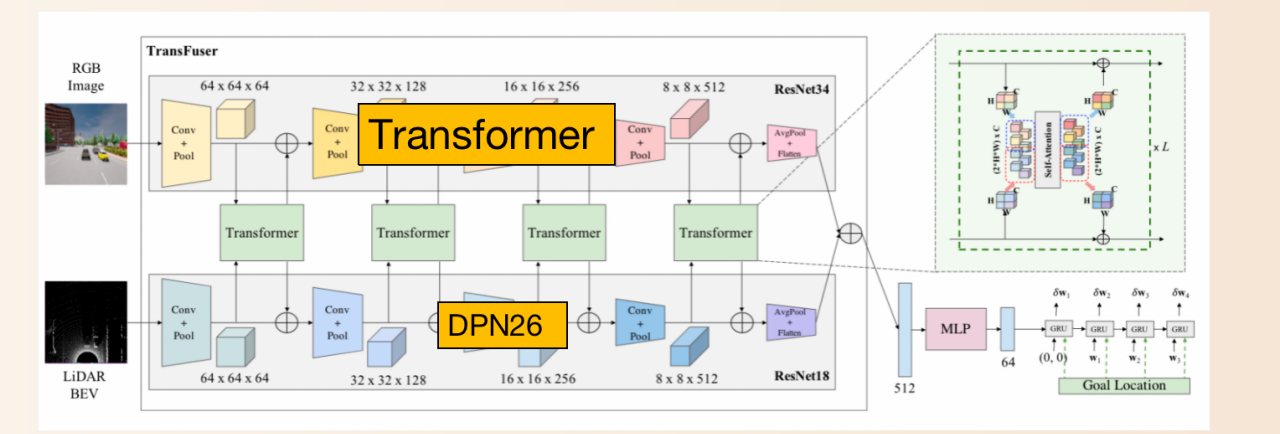
4:郑州数据集：分类结果可视化：



下周计划

1：跑 Temporal Attention 在 DPN 后试试

2: **实现 GPT 结构的层间嵌入代码**



3: 跑URFC大数据集的实验，说不定时序convTransfomer在大数据集上比DPN更好