TSINGHUA UNIVERSITY

Homework 3

Student name & ID: 黄嘉诚 *2023311381*

Course: 深度学习 – Professor: 龙明盛

Due date: *December 27, 2023*

Part One: Transformer

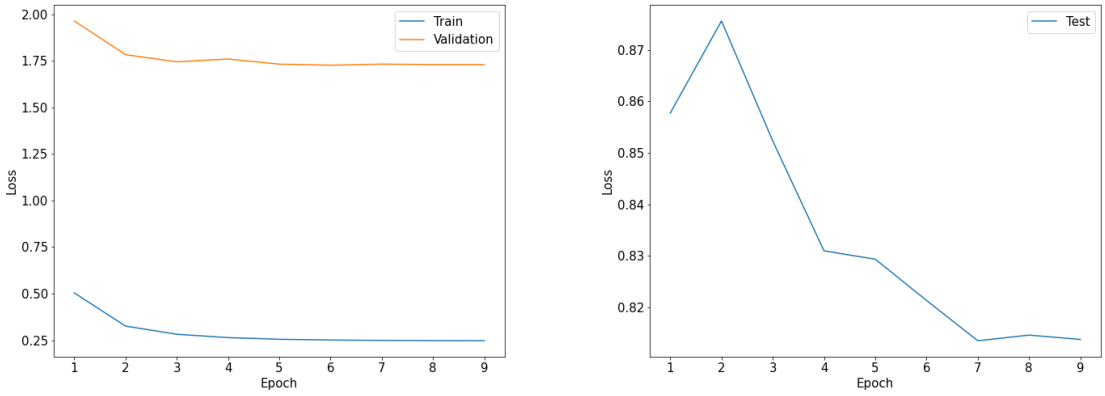
Task 1

Construct the standard Transformer and train your model from scratch using the recommended deep learning framework. Validate your model on the valid set, and report training and validation curves. Evaluate the best model on the test set.

Solution. 补充后的注意力模块见 attention.py 文件，直接运行它可得

```
self_attn_output error:  0.00014436155772317141
masked_self_attn_output error:  0.00014208846529652346
attn_output error:  0.00012462518421286511
```

结果非常接近 0，说明注意力模块实现正确。直接运行 run.py，将得到的结果绘图：最终在测试集上的 MSE 为 0.8216503858566284，MAE 为 0.6647586226463318。



(a) 训练和验证集　　(b) 测试集

Figure 1: Transformer 运行结果

Task 2

Please explain why the mask is necessary in Transformer and how it is implemented in your code.

Solution. 在 Transformer 网络中，mask 技术是必须的，主要原因是为了处理自注意力机制（self-attention）时的信息泄露问题。自注意力机制允许模型在处理序列数据时关注序列中的不同位置，但在一个序列中的某个位置计算注意力时，理论上它可以看到该位置之后的所有信息。在训练过程中，为了保证模型是自回归的，需要增加 mask 以避免预测时模型看到待预测单元及其后面的序列。如果没有 mask，那么模型将会作弊，无法学习到有效信息，泛化能力大大降低。同时，采用 mask 技术可以代替原始的逐个生成的自回归技术，采用矩阵计算加速了学习过程。

Task 3

Please visualize the attention values of some typical words and check whether the Transformer can learn meaningful attention values.

Solution.

Task 4

Please adopt an extra technique (Hint: some linear attention or sparse attention) you find in other materials to further improve your model. Please explain why you chose it, check whether it works on our given dataset, and give a convincing reason.

Solution. 1. 优点：字母表字母少，所需参数量相应较小；避免了单词不在词表中的情形，预处理数据更方便；更适合词性变化等字母级别的任务学习。

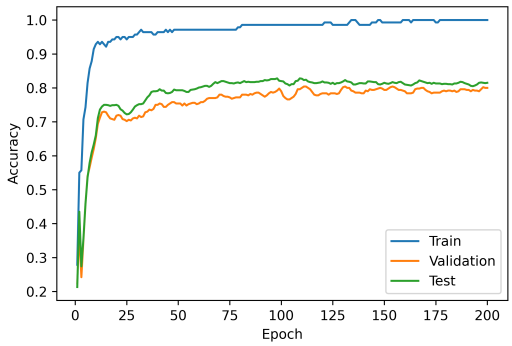
2. 缺点：字母出发需要先学会构建单词，单词中字母的组合方式非常复杂，RNN 仅仅学习到这一点就很困难；字母基础的 RNN 更难捕捉到长程关系，因为一段话可能只有几十个单词，但字母有上百个；因此注意力机制会消耗大量内存，对模型进一步改进不利。

Part Two: Generative Adversarial Networks (GAN)

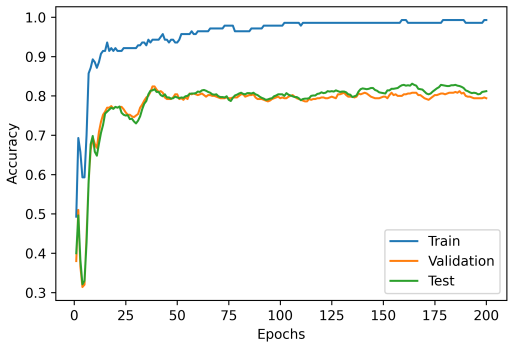
GAN Implementation

Solution. 直接运行即可，训练曲线见图3。

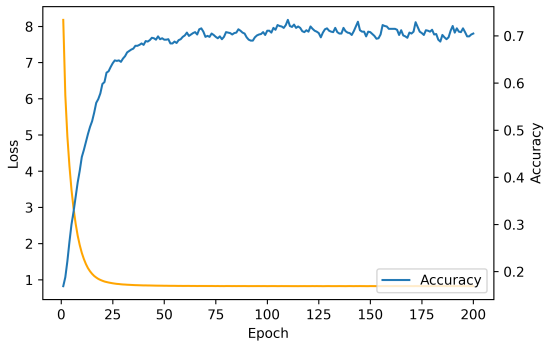
1. GCN, Best Test Accuracy: 0.828, Train: 0.986, Val: 0.792;
2. GAT, Best Test Accuracy: 0.831, Train: 0.986, Val: 0.808;
3. Node2Vec, Best Test Accuracy: 0.734, Loss: 0.8247



(a) GCN



(b) GAT



(c) Node2Vec

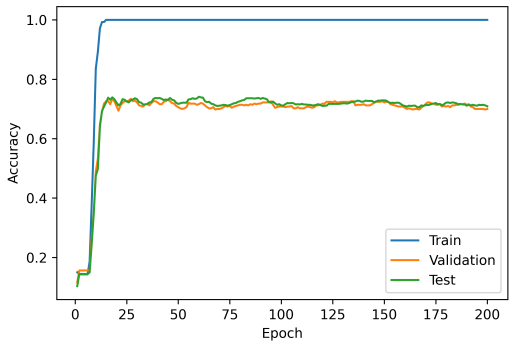
(d) Node2Vec(visualize)

Figure 2: GCN，GAT 与 Node2Vec 运行结果

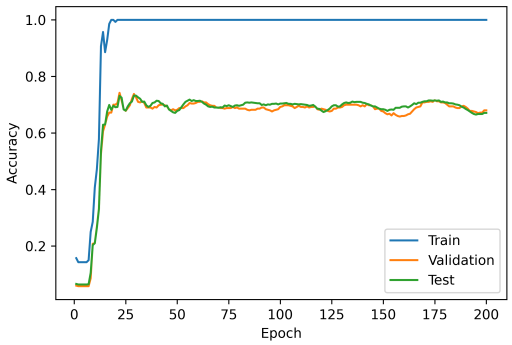
Least Squire GAN

Solution. 选择实现 DeepGCN，代码在同名文件 DeepGCN.py 中。参考了 PyG 所给实例（ogbn_proteins_deepgcn），得到结果如图4所示，层数太大，效果反而不好。

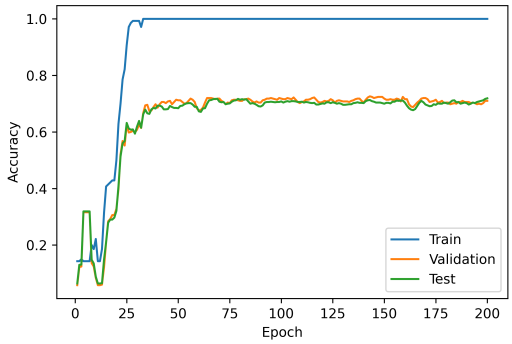
Layers = 1, Best Test Accuracy: 0.741, Train: 1.000, Val: 0.716
Layers = 2, Best Test Accuracy: 0.732, Train: 1.000, Val: 0.738
Layers = 4, Best Test Accuracy: 0.719, Train: 1.000, Val: 0.71
Layers = 8, Best Test Accuracy: 0.743, Train: 1.000, Val: 0.7
Layers = 16, Best Test Accuracy: 0.746, Train: 1.000, Val: 0.752
Layers = 32, Best Test Accuracy: 0.681, Train: 1.000, Val: 0.648
Layers = 64, Best Test Accuracy: 0.6, Train: 1.000, Val: 0.622



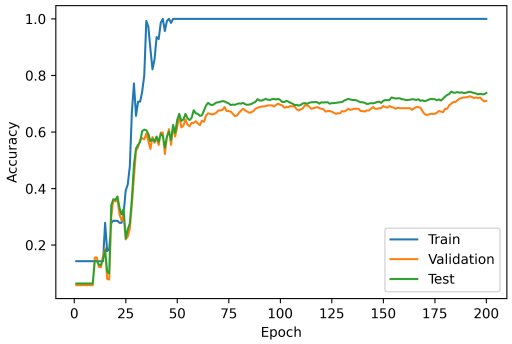
(a) Layers = 1



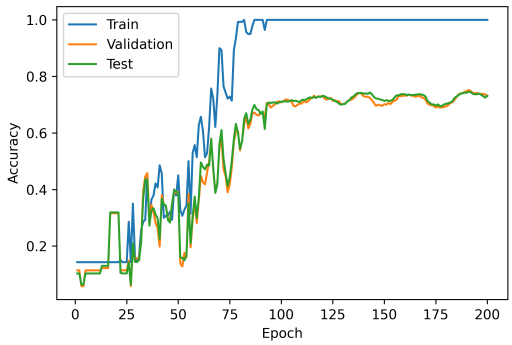
(b) Layers = 2



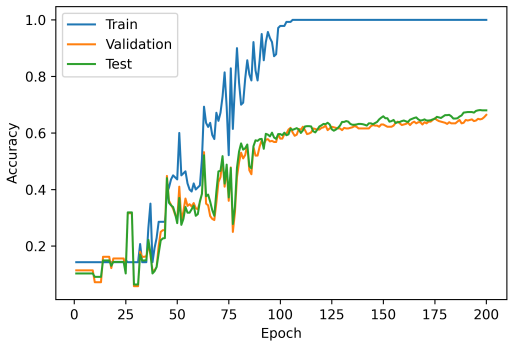
(c) Layers = 4



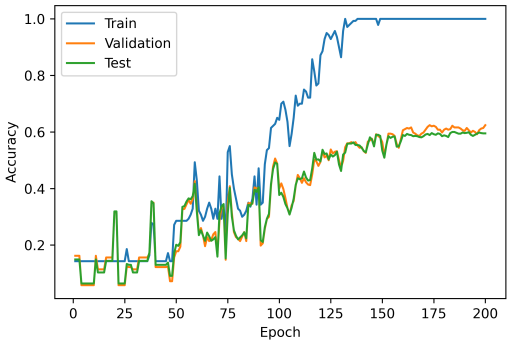
(d) Layers = 8



(e) Layers = 16



(f) Layers = 32



(g) Layers = 64

Figure 3: GCN，GAT 与 Node2Vec 运行结果