深度学习第二次作业

未央-能动 11　张立博　2021012487

2024 年 11 月 22 日

1 Modeling World Dynamics with Recurrent Neural Networks

1.1 Define and train an LSTM-based world model

分别使用一个卷积层和一个线性层对 state 和 action 进行编码，然后将两者拼接作为当前的输入
LSTM 使用 pytorch 内置的实现
训练和验证曲线如下：

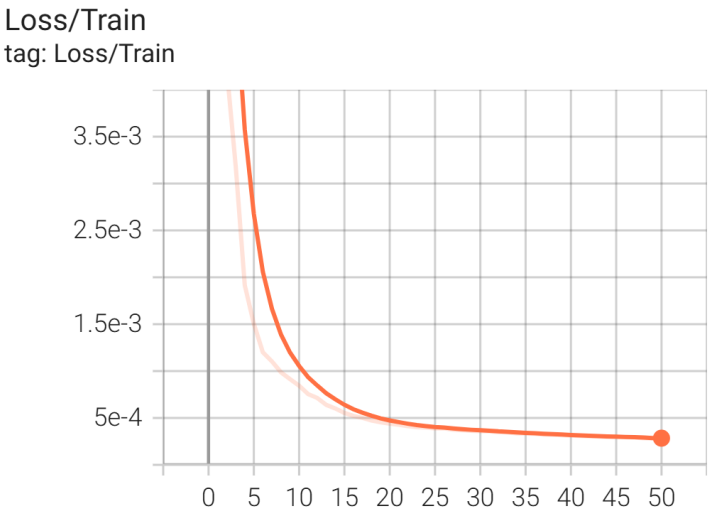

图 1: 训练曲线

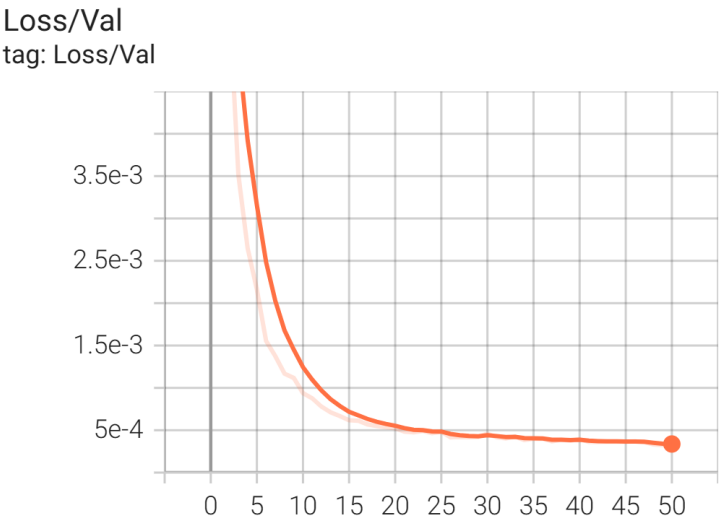

图 2: 验证曲线

Test Loss = 0.0003

1.2 Implement an LSTM layer from scratch

使用手动实现的 LSTM 层，训练和验证曲线如下：

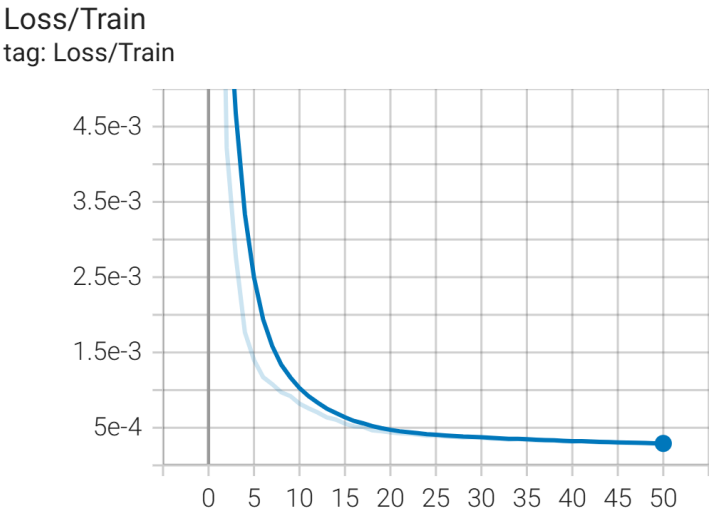

图 3: 训练曲线

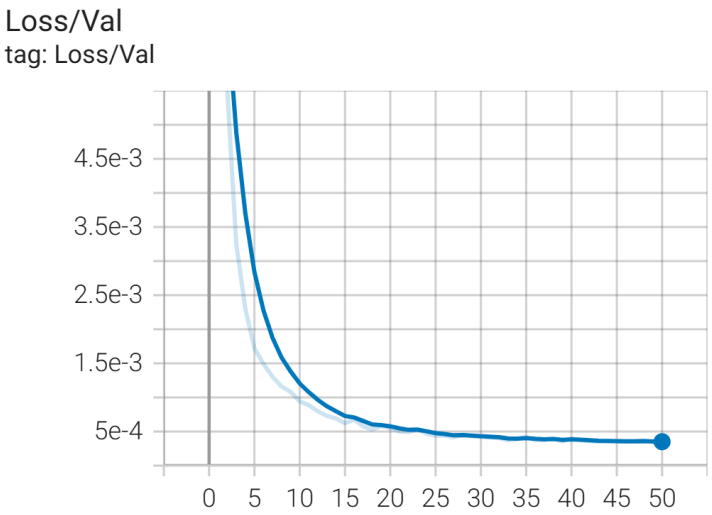

图 4: 验证曲线

Test Loss = 0.0004

与 pytorch 内置 LSTM 层的训练和验证曲线对比如下：

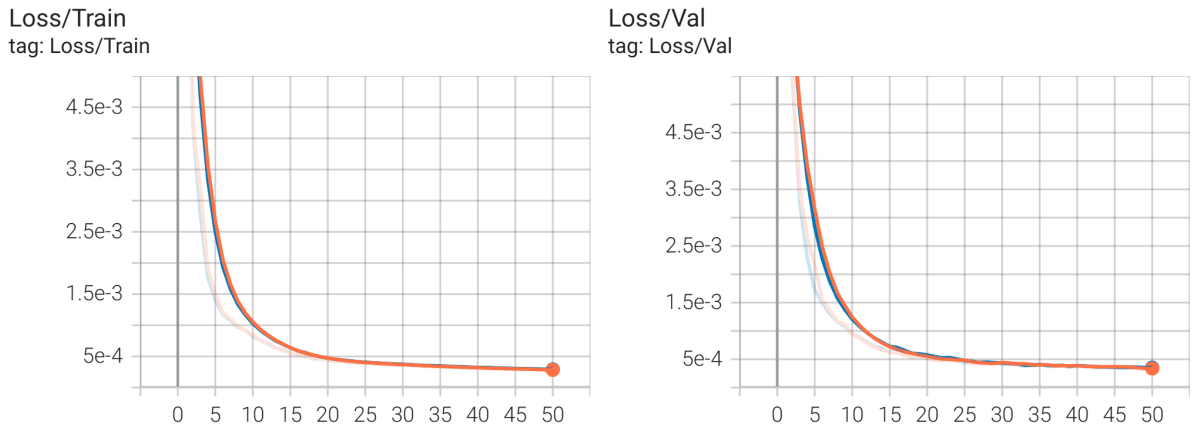

图 5: 对比

其中黄色曲线为 built-in LSTM，蓝色曲线为手动实现的 LSTM

可以看出，两者的训练和验证曲线基本一致，说明手动实现的 LSTM 层是正确的

1.3 Evaluate Model with Two Rollout Strategies

分别使用 Teacher Forcing 和 Autoregressive 策略进行 rollout，结果如下：

1.3.1 Qualitative Results

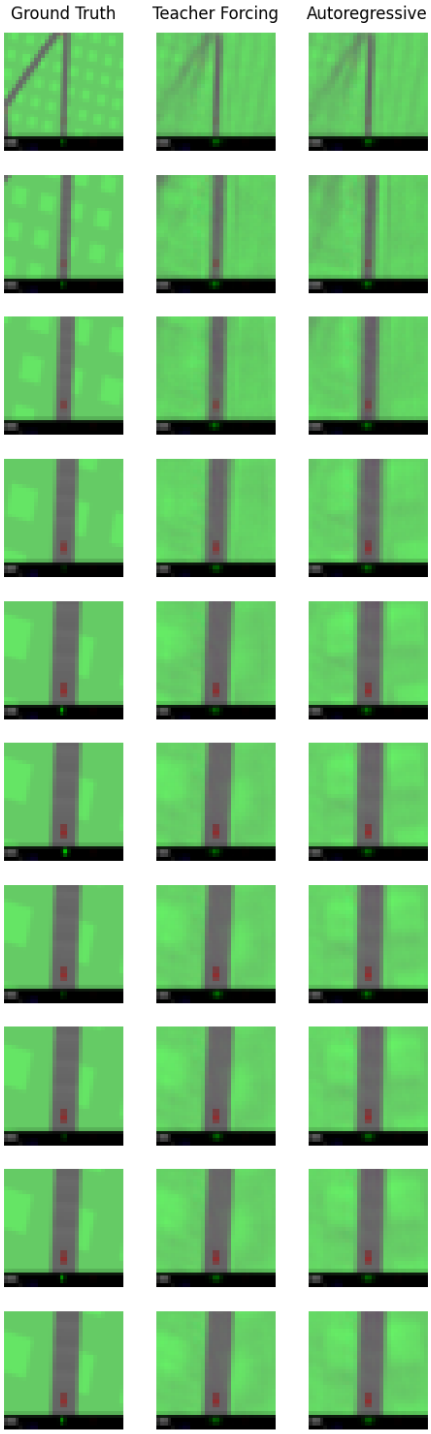

图 6: Qualitative Results

1.3.2 Quantitative Results

Rollout Steps	Teacher Forcing	Autoregressive
5	0.0003	0.0004
10	0.0003	0.0006
15	0.0004	0.0008

表 1: Quantitative Results on Test Dataset

图 6 展示了 ground truth 和两种策略的 10 步 rollout 结果，可以看出两种策略的预测结果都与 ground truth 较为接近

表 1 展示了两种策略在不同 rollout 步数下的测试集 loss，可以看出在 5-15 步的 rollout 下，teacher forcing 策略的 loss 都要低于 autoregressive 策略，且随着 rollout 步数的增加，Autoregressive 策略的 loss 也逐渐增加，说明相比之下 teacher forcing 策略更加稳定

这是因为在 autoregressive 策略中，模型的预测结果会作为下一步的输入，而这个预测结果可能会有误差，导致后续的预测结果也会受到影响，因此随着 rollout 步数的增加，误差会逐渐累积，从而导致 loss 增加

1.4 Potential Improvements

1.4.1 引入噪声进行数据增强

方法: 在训练阶段，对模型的输入（比如状态和动作）添加小幅噪声，模拟 Autoregressive Rollout 中误差累积的场景
原因: 在 Autoregressive Rollout 中，模型使用的是自己的预测值作为下一步输入，预测误差会逐步累积。而训练时采用的 Teacher Forcing 方法使用 Ground Truth 的 state 和 action 作为输入，没有考虑误差累积的情况。通过在训练中添加噪声，模型可以学习对预测误差的泛化能力，提升长期稳定性

1.4.2 使用序列级别的 loss

方法: 在训练时，不仅对每一步单独计算误差，还对整个序列的预测结果计算累积误差，可以通过引入序列级 L2 损失或动态时间规整损失
原因: 传统的逐步损失无法捕捉序列之间的整体一致性，而 Autoregressive Rollout 的主要问题是长期预测时误差的累积。通过优化整个序列的预测一致性，可以让模型更好地学习到全局动态特性，减少长序列预测的偏差

2 Graph Neural Networks (GNN)

2.1 Task A

训练并测试 GCN, GAT, GraphSAGE 三种模型，结果如下：

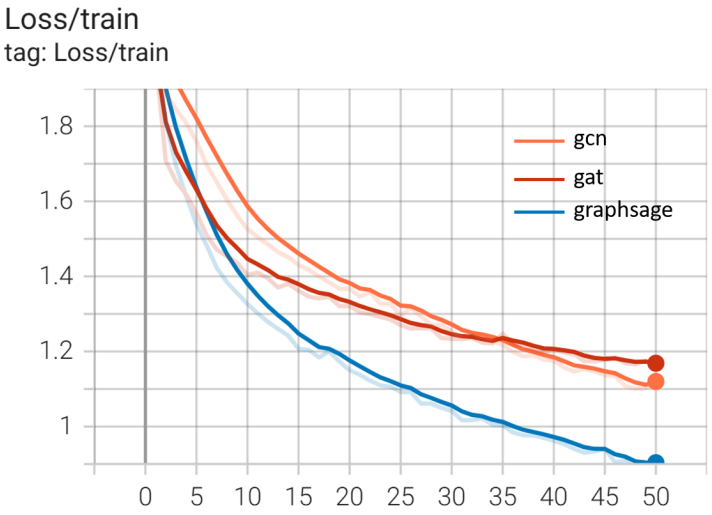

图 7: train curve

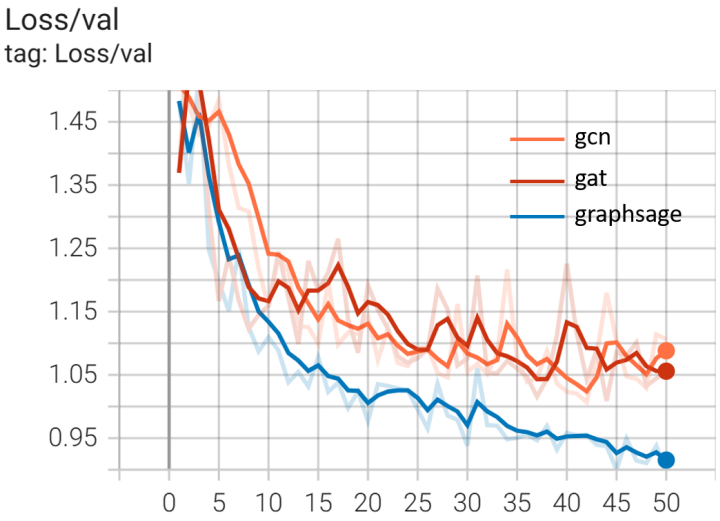

图 8: eval curve

Model	GCN	GAT	GraphSAGE
Test Loss	0.9389	0.9114	0.8197

表 2: Test Loss

三种方法的 R^2 score 如下：

Model	GCN	GAT	GraphSAGE
R^2 score	0.4446	0.4425	0.4829

表 3: R^2 score

结合训练，验证曲线以及在测试集上的 loss 和 R^2 score，可以看出 GraphSAGE 模型在这个任务上表现最好，GAT 和 GCN 表现相近

可能的原因是 GraphSAGE 通过采样邻居节点进行特征聚合，从而避免了信息的过度传播，降低了过平滑问题的影响，在大规模图上表现更好

2.2 Task B

选择手动实现 Deep GCN 模型，设计细节如下：

- num_layers: 6
- hidden_dim: 64
- 每层包含 BatchNorm 和残差连接

具体实现见 deep_gcn.py
该模型的测试集 Loss 和 R^2 score 如下：

Method	GCN	GAT	GraphSAGE	Deep GCN
Test Loss	0.9389	0.9114	0.8197	**0.7790**
R^2 score	0.4446	0.4425	0.4829	**0.5153**

表 4: Test Loss and R^2 score

可以发现更深的 Deep GCN 模型在这个任务上表现更好，说明在这个任务上增加网络深度可以捕获更复杂的图结构特征，提升模型性能