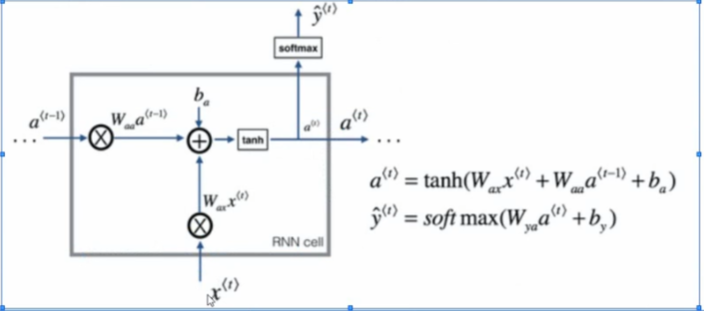
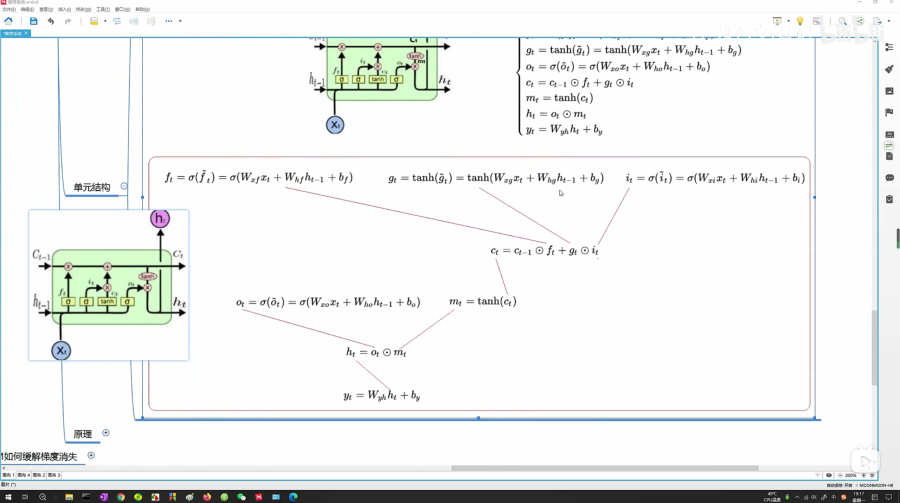
# 基础知识

## RNN



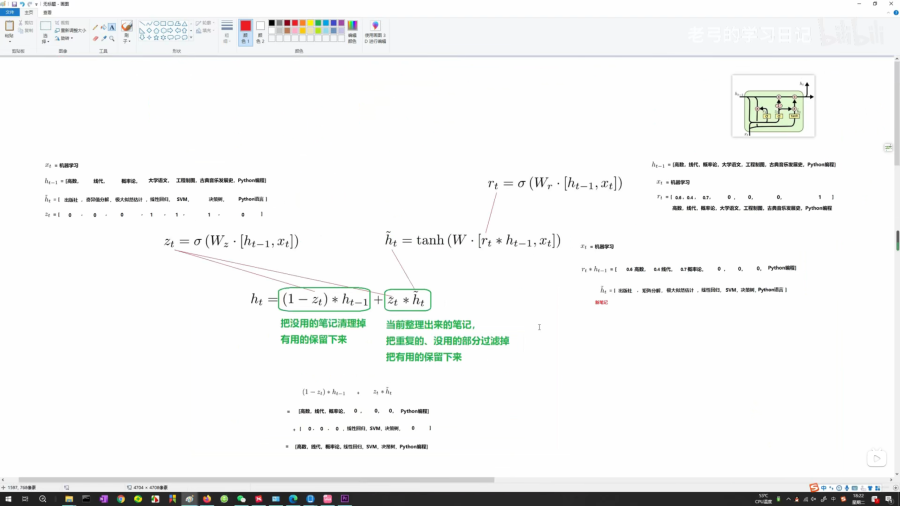
## LSTM



## Bi-LSTM

相当于双向LSTM，只需要将原序列作为正向LSTM的输入，原序列的逆序列作为逆向LSTM输入，再将两个LSTM的输出叠加并经过线性层即可实现Bi-LSTM

## GRU



## Torchtext

1. 安装方式：conda install -c pytorch torchtext
2. Field用于定义字段的处理方式，决定了如何对文本进行预处理，包括分词、转换小写、构建词汇表
3. Tabulardataset用于加载表格形式的数据（从csv中加载），并使用field进行预处理

## 准确率、召回率、精确率、F1值

假设有a、b、c三类

1. 如果有 100个样本，分类器正确分类了 90 个，那么准确率为 90%。
2. 如果100个样本里有50个c类，分类器正确识别了30个c类，那么c类召回率为 60%。
3. 如果分类器将50个样本分为c类，而实际只有30个是c类样本，那么c类精确率为 60%。
4. F1值是精确率和召回率的调和平均值，用于综合评估分类器的性能。

# 实验环境

1. GPU：RTX3080
2. 系统：Linux

# 实验内容

## 文本分类

1. 首先使用split\_data.py对数据集进行处理，具体包括：
   1. 使用正则表达式过滤标点符号
   2. 将中文类别映射为整数
   3. 将数据集划分为训练集、测试集、验证集
2. 随后使用random\_embedding.py在./.vector\_cache中生成使用jieba库分词后的词汇表的随机词向量
3. 最后使用训练代码train\_classify.py进行训练，训练代码如下
   1. python train\_classify.py --model LSTM/BiLSTM/RNN/GRU --lr 0.001
4. plot.py为绘图函数，用于绘制混淆矩阵

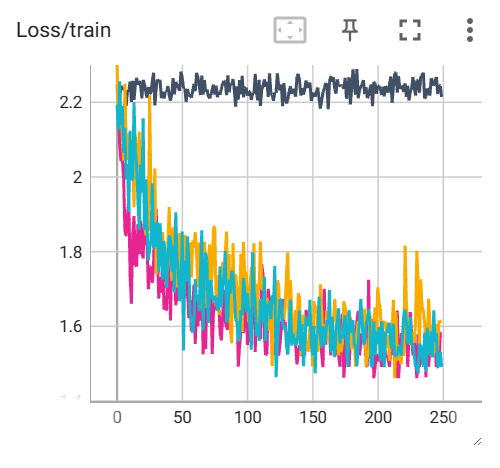
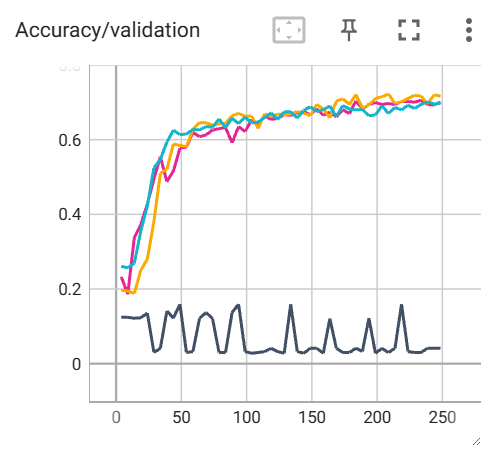
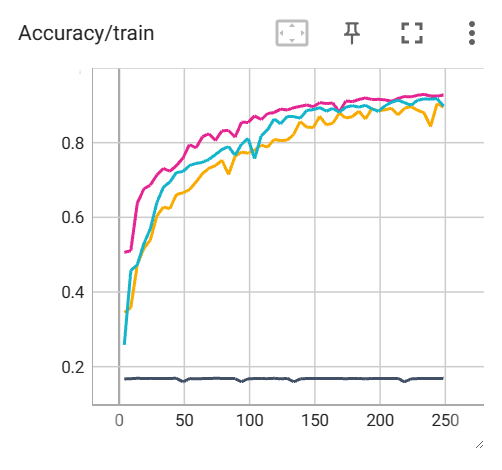
### 温度预测

1. 使用代码train\_forecast.py进行温度预测，网络为LSTM
2. 训练时输入一段序列，并将模型输出序列与标准输出序列（输入序列往后偏移一个时刻）做MSELOSS来对网络进行优化
3. 测试时输入长度为5天的序列，随后用输出的最后时刻温度和隐藏层作为输入，再次输入到模型中往下预测温度，预测后再次取出预测温度和隐藏层作为输入进行预测，以此循环预测后2天的温度

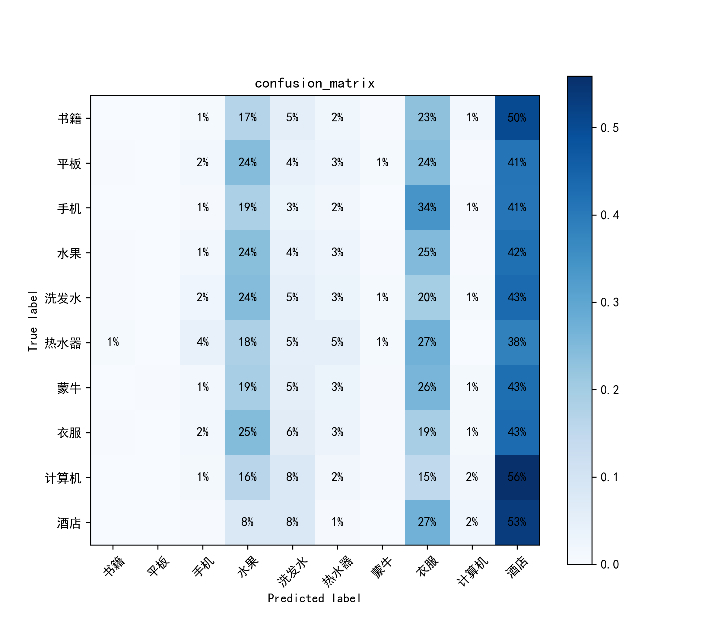
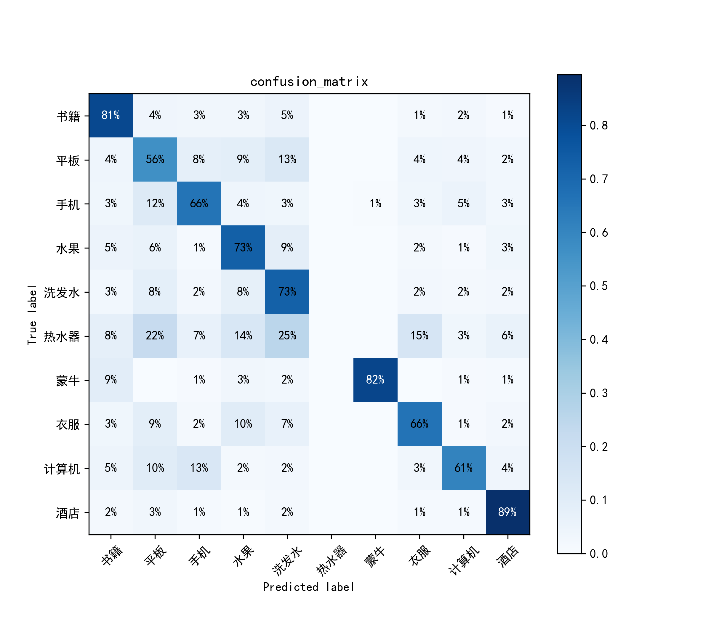
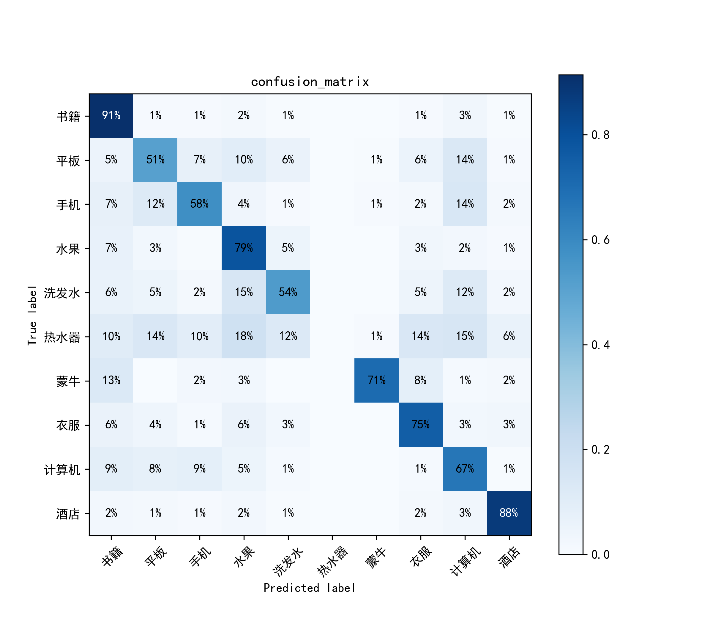
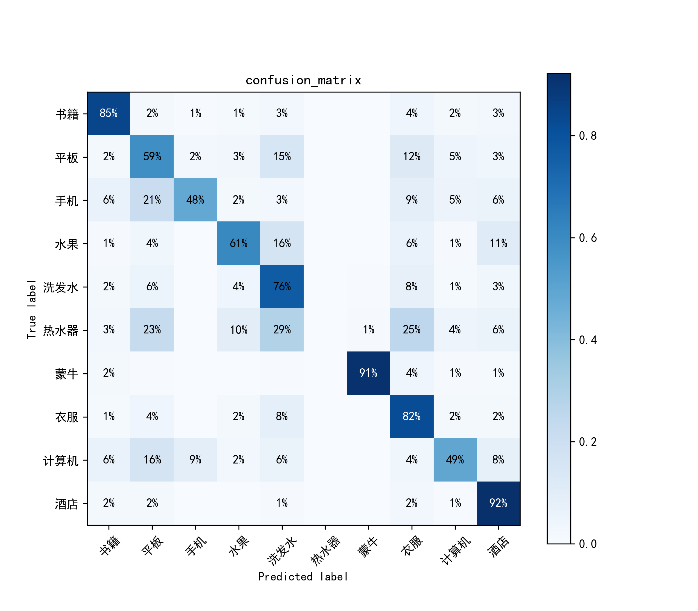
# 实验结果

## 文本分类

### 结果

1. 总体loss及accuracy（黄：BiLSTM, 粉：GRU，蓝：LSTM，黑：RNN）
   1. 
   2. 
2. 准确率、召回率、精确率、F1值

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **RNN** | **GRU** | **LSTM** | **BiLSTM** |
| **Accuracy** | 0.1632 | 0.7086 | 0.6945 | 0.7196 |
| **Recall** | 0.1093 | 0.6470 | 0.6328 | 0.6430 |
| **Precision** | 0.1006 | 0.7058 | 0.6856 | 0.7297 |
| **F1 score** | 0.1048 | 0.6751 | 0.6581 | 0.6836 |

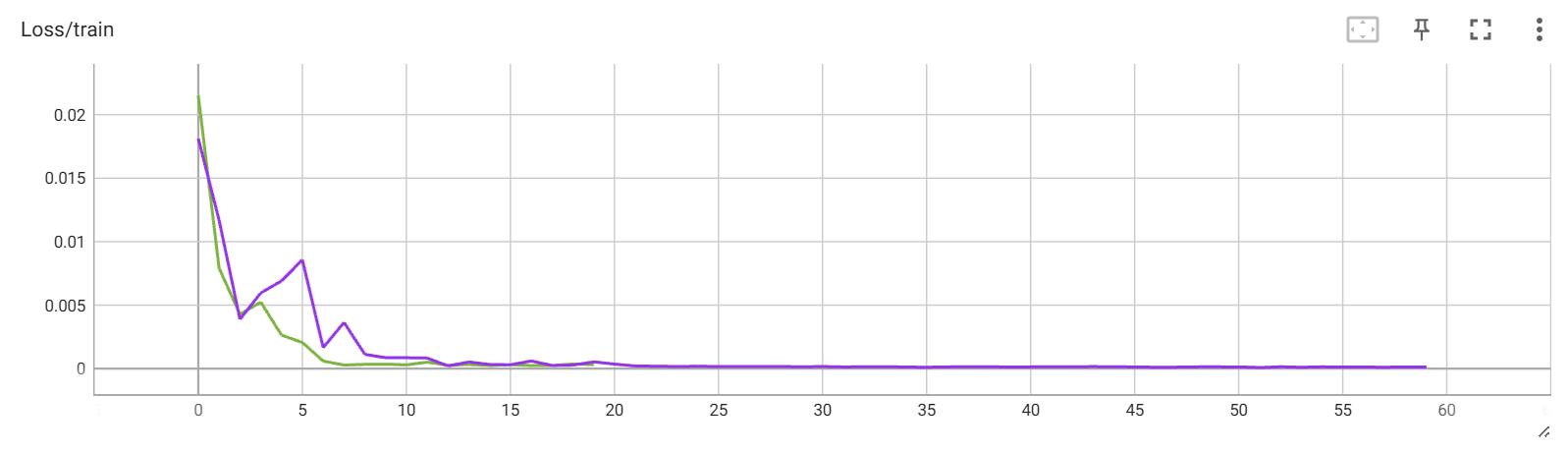
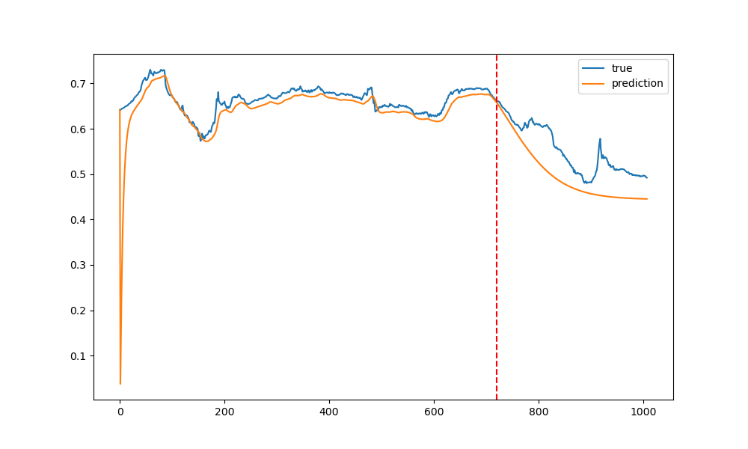
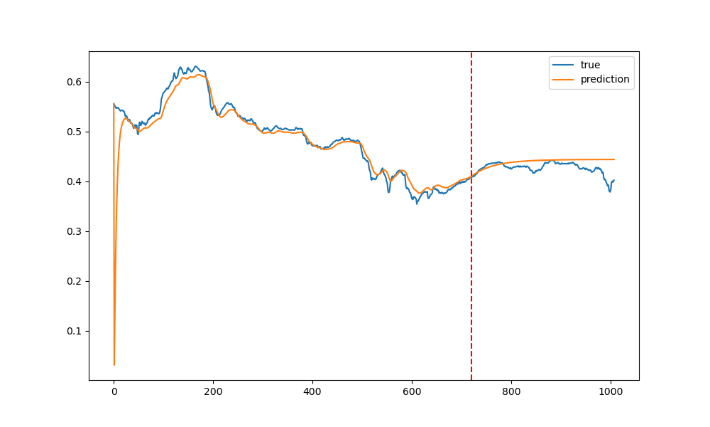
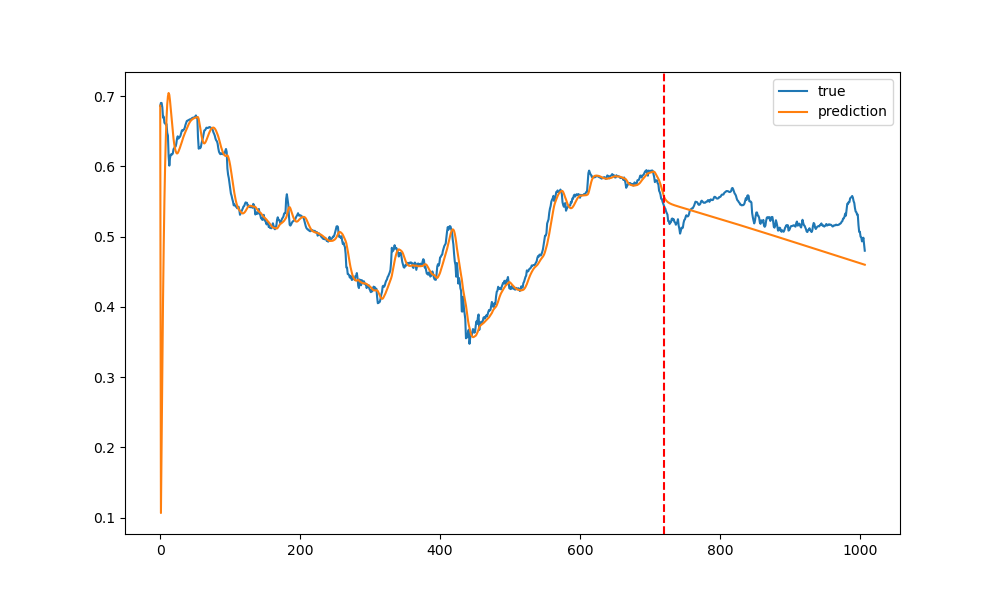
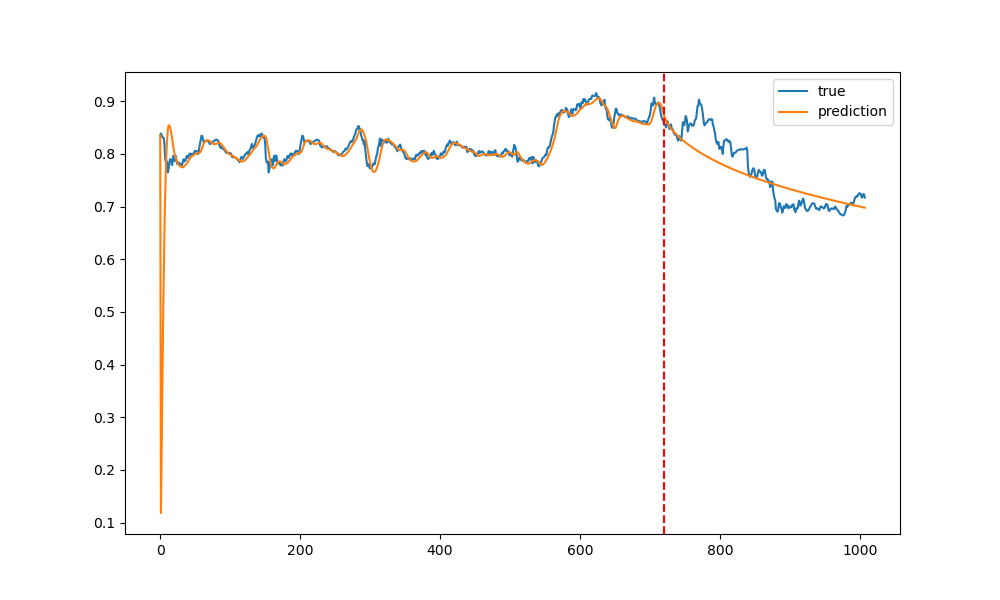
1. 混淆矩阵
   1. RNN  
      
   2. GRU  
      
   3. LSTM  
      
   4. Bi-LSTM  
      

### 分析

1. 输入的序列较长，RNN网络发生了梯度消失的问题，所以训练结果并不理想，loss不收敛
2. 其余的三个网络均有较好的表现，从训练速度上看，GRU速度相对更快，原因是其有更少的参数
3. 其中Bi-LSTM表现最好，模型见./best\_model/classify\_bilstm\_model\_0.7196909351601083\_0.6836805665133082.pth

## 温度预测

### 结果

1. Loss：其中紫线训练了60 epochs，绿线训练了20 epochs，模型见./best\_model/forecast\_model\_60epochs.pth，./best\_model/forecast\_model\_20epochs.pth  
   
2. 预测结果（前720（红色虚线前）为单数据预测结果（输入一个标准输出一个预测），后720（红色虚线后）为多数据预测结果（输入前5天数据，预测后两天，每次预测均以上次预测为输入））
   1. 20 epochs
      1. Mean Squared Error: 0.0035744132474064827  
         Mean Absolute Error: 0.053843267261981964  
         
      2. Mean Squared Error: 0.0003311230393592268  
         Mean Absolute Error: 0.01338923815637827  
         
   2. 60 epochs
      1. Mean Squared Error: 0.0012411749921739101  
         Mean Absolute Error: 0.030614519491791725  
         
      2. Mean Squared Error: 0.0012458749115467072  
         Mean Absolute Error: 0.028334451839327812  
         

### 分析

1. 前144\*5为序列到序列的预测结果（输入上一时刻的标准温度，输出下一时刻的预测温度），可以发现在仅预测一天的情况下网络有较好的发挥
2. 后144\*2为LSTM自主预测的结果（输入上一时刻的预测温度，输出下一时刻的预测温度（本次输出作为下次输入）），可以发现预测较长序列时，模型表现并不佳，仅能预测短期的大致趋势
3. 通过对比20和60 epochs的模型结果，可以发现训练更多epochs的模型有更好的拟合能力，在单数据预测上表现更好，而也正是因为略微过拟合，导致它在多数据预测上表现较差