时序预测学习赛 Report

任子骏
SA24229038
University of Science and Technology of China
renzj@mail.ustc.edu.cn

杨金霖
BE24229005
University of Science and Technology of China
yangjinlin@mail.ustc.edu.cn

潘逸轩
BZ24229001
University of Science and Technology of China
pyx2020@mail.ustc.edu.cn

陈泳融
BZ23006009
University of Science and Technology of China
yrchen@mail.ustc.edu.cn

1 Introduction

AI 在金融营销中的应用时序预测学习赛是阿里云天池比赛中对于基金申赎预测的日常学习赛。

与普通的时序预测任务不同的是，基金申赎预测不仅需要考虑基金收益表现等产品特征变化，更需要考虑到金融营销场景下的特殊属性，财富场景碎片化，用户行为周期长且受到金融行情的波动影响，产品间存在着挤压和替代等复杂关系。

本文旨在详细介绍我们参与这项比赛的具体工作，包括问题描述、问题分析、实验方案、实验过程以及最终结果和分析。

2 Task description and data construction

本金融时序预测学习赛期望参赛者能通过对基金历史申赎时序及基金特征信息的分析和建模，考虑融入对金融行情的洞察（训练数据和测试数据上都存在行情异常波动的时段），最后能精准预测未来每日的基金申购和赎回金额。

比赛链接 https://tianchi.aliyun.com/competition/entrance/532224/information

2.1 问题介绍

基金的申赎金额不仅与时序本身的周期性和波动性有关，也与基金的收益表现和用户行为密切相关，更容易受到行情波动影响，又由于在节假日要归属到下一交易日，故增加了时序的异步性，因此，提供以下数据作为建模数据：

A、基金申赎数据及基金特征信息表

表中包含了本赛题的预测目标，基金每个交易日的申购金额 (apply_amt)、赎回金额 (redeem_amt) 和净申购金额 (net_in_amt)，其中，净申购 = 申购-赎回。

同时，表中给出了基金在每个交易日的基金信息、用户访问信息（各页面曝光 uv 等）

序号	字段	描述	数据类型
1	product_pid	产品ID	string
2	transaction_date	交易日期	string
3	apply_amt	申购金额	double
4	redeem_amt	赎回金额	double
5	net_in_amt	净申购金额=申购金额-赎回金额	double
6	uv_fundown	基金持仓页面曝光uv	double
7	uv_stableown	稳健持仓页面曝光uv	double
8	uv_fundopt	基金自选页面曝光uv	double
9	uv_fundmarket	基金市场页面曝光uv	double
10	uv_termmarket	定期市场页面曝光uv	double
11	during_days	基金持有期（天），即指基金赎回和申购期间间隔的自然日最小天数	bigint
12	total_net_value	累计净值	double

图 1: 基金申赎数据及基金特征信息表

B、行情信息表

表中给出了「一年期中债商业银行同业存单收益率曲线 (AAA)」

序号	字段	描述	数据类型
1	enddate	日期	string
2	yield	收益率(%)	double

图 2: 行情信息表

C、时间信息表

表中包含了交易日信息，记录了境内市场的交易日及每日的前后交易日期等

比赛数据以.csv 文件方式提供。

序号	字段	描述	数据类型
1	stat_date	日期	string
2	is_trade	是否交易日	bigint
3	next_trade_date	下一个交易日	string
4	last_trade_date	上一个交易日	string
5	is_week_end	是否本周最后一个交易日	bigint
6	is_month_end	是否月末最后一个交易日	bigint
7	is_quarter_end	是否季末最后一个交易日	bigint
8	is_year_end	是否年末最后一个交易日	bigint
9	trade_day_rank	交易日全局排序（19901219是第1个交易日）	bigint

图 3: 时间信息表

2.2 评价指标

评分数据格式要求与“结果数据样例文件”一致，结果表命名为：predict_table.csv，字段之间以逗号为分隔符，一共包括 5 个字段:

（1）product_pid 和 transacation_date 字段给出了需要预测的产品 ID 及日期；

（2）apply_amt_pred 、redeem_amt_pred、net_in_amt_pred 字段是需要提交的预测结果。

评价指标：

WMAPE（Weighted Mean Absolute Percentage Error，加权平均绝对百分比误差）：

$$WMAPE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i|} \times 100\%$$

从理论上说，WMAPE 值衡量的是加权平均绝对百分比误差，所以 WMAPE 越小代表模型预测越精确，**然而这个比赛的排行榜是倒着排的！这个比赛的排行榜是倒着排的！**，我们一开始尝试发现效果十分不行，WMAPE 值在 1 附近，进行多次调节后排名不升反降，如图所示：

学习赛-长期: **1/5.00**

图 4: 预测值全填 1000 遥遥领先

随后由于尝试太多次，随手将预测值全填了 1000，发现得分居然拿到了 5 分，非常的 Amazing 啊，**所以这个排行榜的排名应是倒序记录。**此后的结果我们只比较得分而不比较排行榜的排名（因为排行榜排名是按照 WMAPE 取高值排名的）。

3 Experiment and Discussion

3.1 代码结构

所有必要代码由 config、data_loader、preprocessing、feature_engineer、model_builder、train、evaluate 七部分组成，具体文件说明如下：

- config.py：用于存放相关参数和全局配置。；
- data_loader.py：处理数据集，负责数据的加载与初步清洗；
- preprocessing.py：负责数据集划分、缩放器拟合、过采样、异常值处理、Label 编码、和最终数据分割函数；
- feature_engineer.py：负责特征工程，包括日期特征提取、周期性编码、滞后特征构造、UV 归一化处理，以及缺失值和日志变换；
- model_builder.py：用于定义位置编码层、Transformer 编码器块以及最终模型构建的类
- train.py：主训练脚本，调用上述模块进行数据加载、特征工程、预处理、模型构建、训练和验证。
- evaluate.py：用于可视化与评价指标计算的辅助类与方法。
- cnn_train.py: 包括 CNN 架构以及所有训练过程。

3.2 数据预处理

3.2.1 数据加载与初步清洗

在数据加载与初步清洗阶段，代码首先通过 DataLoader 类从指定的 CSV 文件中读取产品数据集（df_product）、收益率数据集（df_yield）和时间特征数据集（df_time），以及预测用的数据表（df_predict）。

我们通过 pd.to_datetime 将字符串格式的日期字段转换为 datetime 类型，如 transaction_date、end-date、stat_date，以确保数据在时间维度上保持一致性。转换格式并删除无效日期行，可以保证数据的时间统一性和有效性，这对时间序列模型极为重要。我们将三个 dataframe 通过时间聚合到同一个 dataframe 中，方便后续操作。

随后，我们对关键特征（如 during_days、total_net_value）的缺失值在 PID 分组下使用中位数填补，以减少数据稀疏和异常分布的影响。

同时，我们注意到在训练集中的部分 PID 在测试集中并不需要预测，通过对比训练集和预测集中的产品 PID，我们删除仅在训练集中出现而不在预测集中出现的 PID 对应数据，确保训练数据与预测数据的产品集合一致。

3.2.2 特征工程

在特征工程环节，代码通过 FeatureEngineer 类对已合并整合的产品数据进行进一步加工。首先提取时间特征，由于原始的年、月、日信息不足以体现时间维度上的周期变化，我们从 transaction_date 中分解出年、月、日、星期等信息，利用 weekday 的值判断是否为周末，并通过正弦和余弦函数实现周期性时间特征的编码，从而捕捉数据在周、月循环中的特征模式。同时为保留日期的时序性，我们利用 add_normalized_time_feature 函数将日期转化为以 20210104 为起点的天数。这种转换方法使得日期变成了一个连续的数值型特征，可以更好地与其他特征进行数值化分析。

考虑到序列预测本质上依赖历史信息，大部分都时间序列预测都存在滞后性，我们为 apply_amt、redeem_amt、net_in_amt 等需要预测的金融交易特征生成滞后特征（lag 特征），如前 1、3、7 天的值，以使模型能够学习历史趋势和周期变化。

同时，我们对浏览量类 UV 特征（uv_fundown、uv_stableown 等）进行简单最大值归一化，降低不同数值量级的特征之间的差异。在收益率特征上使用对数变换，将 yield 转换为 log_yield，以减小偏态，提高训练的稳定性和模型对数据分布的适应性。

考虑到数据中涵盖多个 PID 数据，为防止不同 PID 数据对时间序列的影响，我们使用 LabelEncoder 对 product_pid 进行编码，将 product_pid 视为特定时间序列的特征，为 Transformer 模型的 PID 嵌入层输入提供整数索引。

同时，我们还需要去除一些不需要或者联系不大的特征。我们通过皮尔逊相关系数计算与预测列的相关性，相关性热力图如下所示

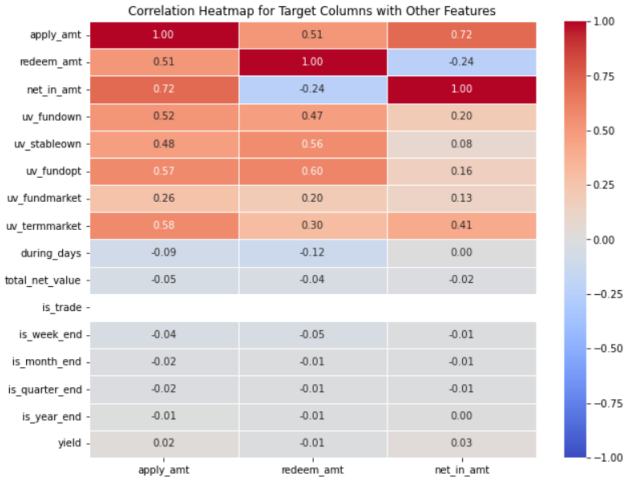

图 5: 皮尔逊相关系数热力图

对于皮尔逊相关系数小于 0.1 的特征将不会纳入考虑。

3.2.3 数据划分

数据预处理部分由 DataPreprocessor 类主导执行，聚焦于特征缩放、数据集划分、异常值处理等关键步骤。

通过对目标特征进行时间序列可视化后我们发现，不同 PID 之间时间跨度太大，且存在明显异常特征，我们首先指定的训练开始与结束日期对数据按时间切片筛选训练集，以确保模型在最近特定的历史时段进行学习，太久远的过去我们选择直接删去。同时，我们对目标列执行 IQR 异常值截断，去除极端值，维持数据稳定分布和模型的鲁棒性。随后使用 MinMaxScaler 对特征和目标进行缩放器拟合操作，仅在训练集特征与目标上拟合缩放参数，避免数据泄露。

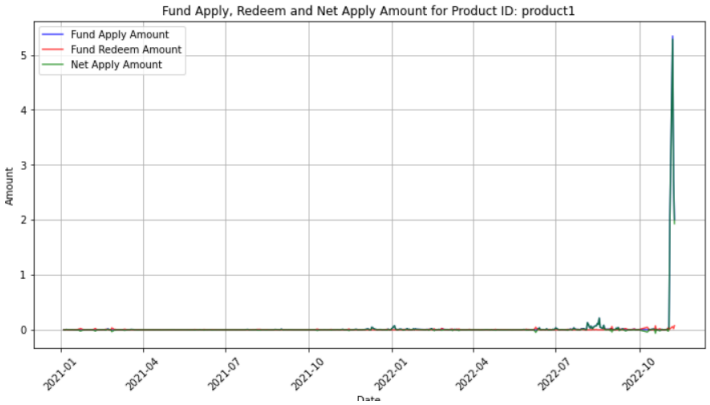

图 6: 明显异常特征可视化

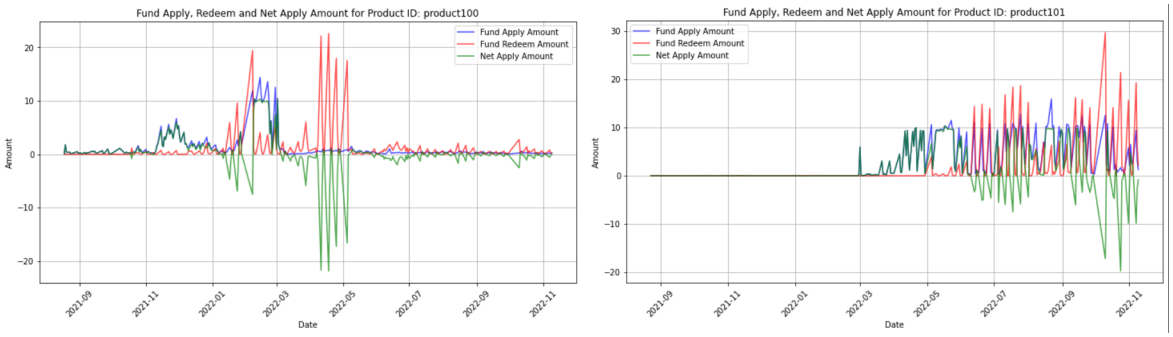

图 7: 不同 PID 特征可视化对比

在划分训练集与验证集阶段，以滑动窗口的方式对序列数据构造训练样本和验证样本，使每个样本的输入为历史窗口的特征数据，输出为未来预测的目标数据，确保模型学习到时间序列的动态特征。我们发现测试集需要预测的数据都是 10 天，所以我们选择滑动窗口大小为 10，直接预测十天之后的数据对于测试集我们采用相同构造，选择测试集前十天的数据放入模型验证，使用同样的预处理流程对测试数据进行处理。

4 Modal Principle and Advantage

本任务是一个基金申赎量预测问题，由于需要预测测试集中各只基金在未来交易日的申购金额与赎回金额，这属于一个时间序列回归问题。

由此，我们参考相关金融时间序列预测文献，选择使用 CNN 以及 Transformer 架构的时间序列模型进行训练。

4.1 CNN

传统的卷积神经网络（CNN）最初是在计算机视觉领域广泛应用的模型架构，其核心思想是通过卷积操作在局部区域内提取特征，再结合池化层（Pooling）和全连接层（Fully Connected Layer）逐级提炼信息，从而实现如图像分类、目标检测等任务。CNN 的优势在于它能够利用卷积核共享参数的方式，在数据的局部结构中提取可迁移、可复用的特征表示。

我们将 CNN 引入时间序列或数值预测任务中时，需要对输入数据加以适应性改造。传统的时间序列数据是一维随时间变化的信号，我们通过在特征与时间维度上组合形成“类似图像”的二维输入数据将其转换为 CNN 可处理的形式，使卷积核在时间轴和特征轴上进行扫描，从而提取时间序列在局部时间范围内的特征模式。

为了实现这一点，我们在数据集处理环节将输入数据通过 TimeSeriesDataset 类打包为 (batch_size, 1, feature_size, num_features) 的张量格式，即将每个样本的特征集作为单通道图像输入（channel=1），其中 feature_size 对应的是时间维度的长度（如 10 天的窗口），num_features 对应的是特征维度的数量（如各类 UV 特征、时间特征、收益率等）。

我们在 Net 类中首先定义了两组卷积层与池化层的组合。第一组卷积层（conv1）包含 Conv2d、BatchNorm2d、ReLU 和 MaxPool2d，其中卷积核和池化层会在特征和时间轴上移动，通过本地感受野捕捉子序列间的特征模式。Conv2d 负责在局部区域提取特征，BatchNorm2d 使特征归一化，加速训练并提高稳定性，ReLU 作为激活函数引入非线性，MaxPool2d 进一步降低特征图的尺寸，提取关键信息。第二组卷积层（conv2）进一步抽取更高级别的特征，并再次通过池化缩小特征图的空间维度。经过两次卷积与下采样后，输入张量被提炼出相对紧凑和高级的特征表示。然后模型将四维的特征图展平为一维向量，再通过全连接层（fc1）和归一化层（BatchNorm1d）及 ReLU 激活函数等组合，对提炼后的特征进行整合与映射。Dropout 则用来减少过拟合风险。最后一层全连接层（fc2）将特征映射到所需预测的维度上，通过设置输出大小为 num_label * 10，实现对未来 10 天（或指定预测窗口）的多维目标值进行同时预测。

相较于传统的 CNN 结构，我们最明显的修改是输入数据的组织方式和最后输出层的大小设定。输入不再是彩色图像（3 通道），而是单通道的二维数据（时间维度与特征维度）；输出层的神经元数量也不再对应图像分类的类别数，而是对应接下来数天内需要预测的时间序列数据点个数。

4.2 Transformer

Transformer 架构是由 Vaswani 等人在论文《Attention is All You Need》中提出的一种完全基于注意力机制的序列到序列模型，其主要创新在于彻底摒弃了传统序列模型中常用的 RNN 或 CNN 结构，而只使用注意力（Attention）来建模序列中元素间的关系。Transformer 最核心的模块包括自注意力（Self-Attention）和前馈神经网络（Feed-Forward Network），并通过多头注意力（Multi-Head Attention）和位置编码（Positional Encoding）进一步增强模型捕捉复杂特征和长期依赖的能力。

传统的 RNN 或 LSTM 在处理长序列时面临序列长度与计算效率的矛盾，而 Transformer 利用自注意力机制可以在常数时间内（相对于序列长度）对序列中任意位置的元素进行建模，从而有效解决了长距离依赖问题。此外，Transformer 通过多头注意力让模型能够从多个子空间中并行提取特征，提升了模型的表达能力。位置编码的引入是因为 Transformer 没有像 RNN 那样的序列顺序信息，位置编码将时

间序列的位置信息以正弦和余弦波的方式加入到特征中，让模型通过周期函数的变化捕捉位置依赖关系。

首先，我们通过 PositionalEncodingLayer 类实现位置编码层，这一模块在原始 Transformer 中用于给序列中每个位置的表示添加可区分的位置信息。代码中使用正弦和余弦函数生成了一个可复用的位置编码张量，然后在 call 方法中将这一位置编码与输入特征 x 相加，从而为序列的每个时间步引入了位置信息。

接下来，TransformerModelBuilder 类中的 transformer_encoder 方法实现了 Transformer 的编码器子层结构。这包括两部分：多头自注意力层（MultiHeadAttention）：通过 layers.MultiHeadAttention 对输入序列进行自注意力计算，让序列中的任意两个位置彼此信息交互，得到加权平均后的表示。前馈神经网络（Feed-Forward Network）：在注意力输出的基础上加入由全连接层构成的前馈网络，用激活函数（ReLU）将线性映射空间提升到更高维度表示空间，从而增强特征变换能力。整个编码器块包含了残差连接与 Layer Normalization，通过 inputs + x 和 LayerNormalization 的组合使得训练更加稳定并加快收敛。build_transformer_model_with_pid 方法则在完整模型构建环节对 Transformer 加以适配与扩展。该方法构建了两个输入：一个是 features_input（特征序列输入），另一个是 pid_input（产品 ID 输入），并为 PID 定义了嵌入层（Embedding），将离散的产品 ID 映射为稠密向量表示。这是对标准 Transformer 的修改：传统 Transformer 大多只处理序列数据，这里额外加入 PID 嵌入作为全局上下文特征，帮助模型在预测时区分不同的产品特性。

在数据通过位置编码和多个 Transformer 编码器块处理之后，代码使用全局平均池化层（GlobalAveragePooling1D）来对时序维度进行聚合，以得到固定大小的隐藏表示。随后通过全连接层和 Dropout 层进一步特征变换与正则化，然后将得到的特征与 PID 的嵌入向量级联（Concatenate）在一起。这样，模型最终的输出层（Dense）会基于序列特征和产品特征的综合表示来预测多步未来的目标值。

4.3 基于 LSTM 门控与 Attention 注意力机制的混合模型

LSTM 通过门控机制有效捕捉序列中的长期依赖关系，防止梯度消失或爆炸，在处理具有长期依赖性的时间序列数据时有显著优势。而 Transformer 通过自注意力和多头注意力机制全面捕捉序列依赖，能够同时考虑输入序列中的所有位置，更好地理解上下文关系，实现高效的并行计算。

混合模型的输入为一个时间序列数据张量，形状为 (batch_size, time_steps, features)，例如过去若干天的多维度特征。PID 编码后通过 Embedding 层将 ID 转化为密集向量表示，并在后续步骤与序列特征提取的结果融合。网络架构如下文所叙：

LSTM 编码层：首先采用 LSTM（Long Short-Term Memory）网络对输入序列进行编码。在处理序列时，LSTM 会在每个时间步输出一个隐状态 h_t 和单元状态 c_t。通常，我们会设置 LSTM 层返回整个序列的隐状态张量 (batch_size, time_steps, hidden_dim)，并在最后一个时间步得到最终的隐藏状态 (batch_size, hidden_dim)。

Attention 层：在获得 LSTM 输出的所有时间步隐状态后，我们引入 Attention 层。将 LSTM 所有时间步的隐藏状态作为“键”（key）与“值”（value）对进行匹配。Attention 通过计算 query 与每个 key 的相似度来分配权重，从而对整个序列信息进行加权求和，得到上下文向量（context vector）。与单纯的 LSTM 相比，Attention 可在预测阶段动态关注序列中最重要的时间点和特征；与单独的 Attention 架

构（例如纯 Transformer）相比，LSTM 已经通过门控机制捕捉了输入序列的时间依赖结构，Attention 则强化了对关键时间步的选择能力。上下文向量与 LSTM 最终状态融合：Attention 计算得到的上下文向量与 LSTM 的最终隐藏状态（或者最后一步的隐状态）进行拼接或合并，这样可以在保留序列全局语义的同时突出关键时间步特征。

PID 嵌入与合并：对于多实体（如不同产品）的预测任务，会将 PID 映射到 Embedding 向量，并将该 PID 向量与前面得到的序列表示（上下文 + 最终隐状态）再次融合。这使模型能够在预测时同时考虑不同实体的特性差异。

预测层：最后通过全连接层进行回归或分类输出。例如，对于时间序列预测任务，会将上述合并后的特征送入 Dense 层以预测未来若干步的数据，输出形状 (batch_size, forecast_horizon, target_dim)。

4.4 参数调整过程

对于 CNN 模型，直接通过接受参数的形式进行调参，对于 Transformer 以及 LSTM+Attention 的混合模型，调节的参数主要在 Config.py 文件中，主要参数调节为滑动窗口大小。由于测试集需要预测的数据都是 10 天，所以我们当时选择滑动窗口大小为 10，即前 10 天的特征 $t_1, ..., t_{10}$ 预测后 10 天 $t_{11}, ..., t_{20}$ 的目标值。当然，我们后续调整了滑动窗口大小，以前 20 天 $t_1, ..., t_{20}$ 的特征预测后 1 天 t_{21} 的目标值，从理论上说只预测后 1 天的目标值结果更加鲁棒，但由于需要预测的时间有 10 天，将 t_{21} 的预测值作为新特征继续预测 t_{22} 甚至不如直接将 t_{21} 的预测值直接赋予 t_{22}。前 20 天 $t_1, ..., t_{20}$ 的特征预测后 1 天 t_{21} 的目标值从效果来看比前 10 天的特征 $t_1, ..., t_{10}$ 预测后 10 天 $t_{11}, ..., t_{20}$ 的目标值好一些，但由于缺乏可解释性，且效果并没有大幅提升，我们最后还是选择前 10 天的特征 $t_1, ..., t_{10}$ 预测后 10 天 $t_{11}, ..., t_{20}$ 的目标值。

考虑到预测申购金额 (apply_amt)、赎回金额 (redeem_amt) 和净申购金额 (net_in_amt) 存在如下关系净申购 = 申购-赎回，同时预测的评价指标为加权平均绝对百分比误差 WMAPE，故存在几种利用该关系的方式，在 CNN 架构中，我们只预测 apply_amt 与 redeem_amt，net_in_amt 由相减得出，将 WMAPE 作为损失函数指导梯度下降；在 Transformer 架构以及混合架构中，我们同时预测 apply_amt、redeem_amt、net_in_amt 三个指标，在预测时使用加权计算 $df\_predict[net\_in\_amt\_pred] = (apply\_amt\_pred - redeem\_amt) * 0.5 + net\_in\_amt\_pred * 0.5$,WMAPE 作为损失函数，同时以 WMAPE、MAE、MSE 作为 metric 监控指标。从结果来看，使用 WMAPE 作为损失函数比使用其他诸如 MSE 等其他损失在网站结果上表现更好，而在预测时使用加权计算表现更加鲁棒，不会受到较大异常值的干扰。

5 Results

CNN 最好训练集损失为 1.0797、验证集损失为 0.7758，取不同随机种子五次上传平均分数为 0.9671 训练集与验证集损失如图所示，可以看出模型很好的收敛了：

Transformer 最好训练集损失为 wmape: 0.0127（归一化损失，训练集没有反归一化操作），验证集反归一化后 WMAPE: 0.8498565274524923，取不同随机种子五次上传平均数为 0.9566 训练集与验证集损失如图所示。

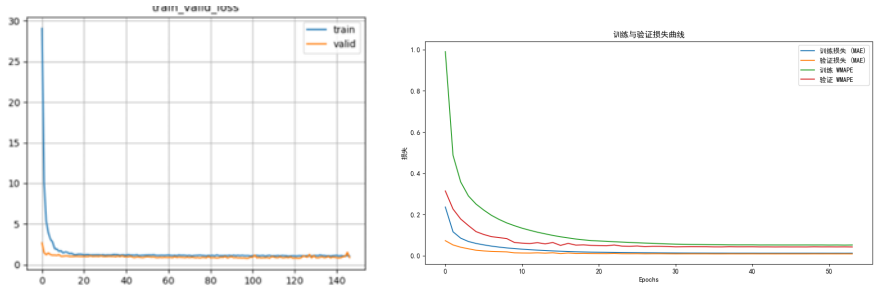

图 8: CNN 损失曲线

lstm + Transformer 最好训练集损失为 wmape: 0.0120（归一化损失，训练集没有反归一化操作），验证集反归一化后 WMAPE: 0.4859062038670395，取不同随机种子五次上传平均分数为 0.9081 训练集与验证集损失如下，同时我们可视化验证集预测值与真值的对比，可以看出预测在趋势上基本符合基金变化：

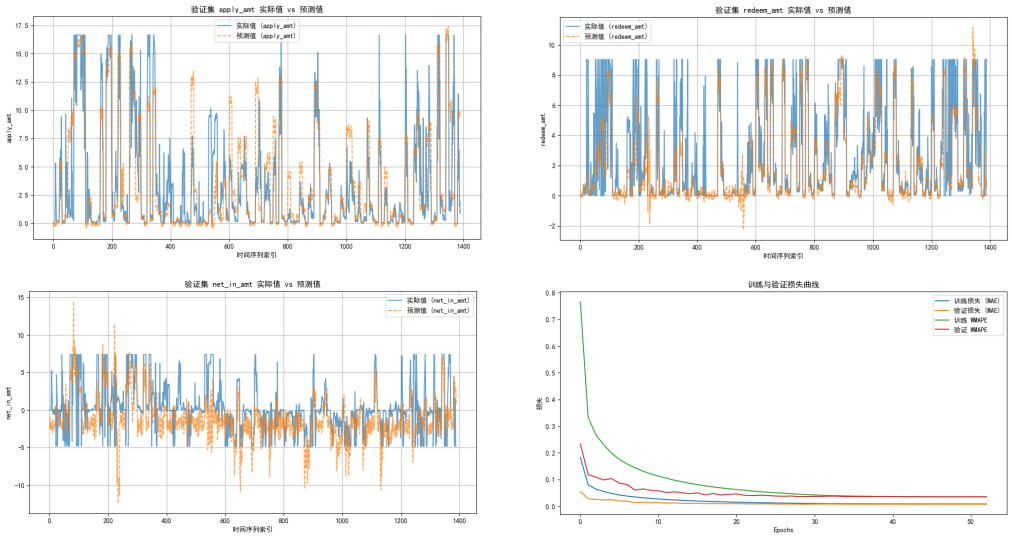

图 9: 混合模型损失曲线与验证集预测值与真值可视化对比

我们队伍的最好模型得分为 0.8554，排名 225/226（wmape 越小越好）。