«5th Place Solution»

Munirov Sultan

Savinov Daniil

Winter 2025

Executive Summary

Our solution achieved 5th place by leveraging an ensemble of diverse Recurrent Neural Networks combined with Online Adaptive Meta-Learner.

Our team consists of two students from the **School of Applied Mathematics and Computer Science (SAMCS) at MIPT**. We are passionate about deep learning in quantitative finance and are highly interested in joining the **Wunderfund** team to continue solving challenging problems in this domain.

Data Processing and Feature Engineering

Feature Engineering

To maximize ensemble performance, we implemented specific feature pipelines for different model architectures in `src/features.py`. We applied non-linear pointwise transforms to the original 32 inputs:

- **Features for BiLSTM:** This model received inputs transformed via **Tanh** and **Sigmoid** functions, preserving the original state dynamics while bounding the magnitude.
- **Features for GRU and LSTM:** These models were trained on a different view of the data, utilizing **Sigmoid** and **SiLU** activations.
- **Ensemble Diversity:** By feeding distinct feature subsets to different architectures, we de-correlate the errors of the base models.

Data Augmentation

We implemented a robust augmentation pipeline specifically for the **GRU** and **LSTM** models. This strategy increases the effective dataset size by 5x using the following multi-step logic:

- **Variance Normalization & Random Scaling:** First, each input sequence is rescaled to match the global median variance of the dataset. To introduce amplitude diversity, we then multiply the normalized sequence by a random factor sampled from a uniform distribution $U(0.6, 1.4)$.
- **Noise Injection:** Since the scaling process produces samples that are highly correlated with the originals, we inject Gaussian noise into every augmented sequence. The noise intensity is set to 0.05σ.

Neural Network Architectures

The ensemble consists of three base models.

Base Models

1. **LSTM:** A single-layer **stateful** LSTM with a hidden dimension of 150. We applied a **multi-stage boosting strategy** and **Stochastic Weight Averaging (SWA)** for better performance.
2. **GRU:** A single-layer **stateful** GRU with a hidden dimension of 128. Like the LSTM, this model utilized the **boosting technique** and **SWA**.
3. **BiLSTM (Bidirectional LSTM):** A two-layer Bidirectional LSTM with a hidden dimension of 256. To diversify the ensemble, this model was trained on fixed **rolling windows of size 60 (stateless)**. This allows the bidirectional layers to aggregate context from both past and "future" within the window without causing data leakage. SWA was also applied to the final weights.

Base Model Performance

The individual predictive power of each base model, measured by the R^2 score on the validation set, is presented below.

Table 1: Base Model R^2 Scores

Model	**R^2 Score**
LSTM	0.3541
GRU	0.3563
BiLSTM	0.3509

Ensemble Diversity

The key to a robust ensemble is the diversity of its base models, measured by the correlation of their prediction errors.

Table 2: Predictions Correlation Matrix

	LSTM	**GRU**	**BiLSTM**
LSTM	1.0000	0.9820	0.9764
GRU	0.9820	1.0000	0.9760
BiLSTM	0.9764	0.9760	1.0000

The Strategic Value of BiLSTM

The BiLSTM model presents a significant computational trade-off:

- **Performance Cost:** In terms of raw R^2 score, BiLSTM slightly underperforms the stateful models (Table 1).
- **Computational Cost:** Its bidirectional nature and larger hidden dimension result in an inference speed 7-10 times slower than the stateful LSTM/GRU models.
- **Crucial Diversity:** Despite these drawbacks, BiLSTM is the cornerstone of ensemble performance. As shown in Table 2, it is the least correlated model. Its **stateless, window-based** processing captures different patterns in the data.
- **Adaptive Weighting:** Our online meta-learner capitalizes on this diversity. During inference, when the test sequence dynamics align with the BiLSTM's perspective, the meta-learner rapidly increases its blending weight (as shown in Figure 1). Conversely, when BiLSTM lags, its weight is suppressed (Figure 2), minimizing its negative impact.

Highway Output Head

For the LSTM and GRU models, we replaced the standard Linear output layer with a custom **Highway Head**. This component is designed to adaptively regulate the flow of information from the RNN hidden states to the final prediction, allowing the network to dynamically choose between non-linear processing and a linear residual path.

Mathematically, let $\mathbf{x} \in \mathbb{R}^{d_{in}}$ be the input vector (the output of the RNN layer). The computation proceeds as follows:

1. **Normalization:** First, we apply Layer Normalization to stabilize the inputs:

$$\tilde{\mathbf{x}} = \text{LayerNorm}(\mathbf{x})$$

2. **Gating and Transformation:** We compute the transform gate $\mathbf{t}$, the non-linear transformation $\mathbf{h}$, and the linear projection of the input $\mathbf{x}_s$ into the hidden dimension d_{mid}:

$$\begin{aligned} \mathbf{h} &= \text{GELU}(W_h\tilde{\mathbf{x}} + \mathbf{b}_h) \\ \mathbf{t} &= \sigma(W_t\tilde{\mathbf{x}} + \mathbf{b}_t) \\ \mathbf{x}_s &= W_x\tilde{\mathbf{x}} \end{aligned}$$

   where $\sigma(\cdot)$ is the Sigmoid function, and W_h, W_t, W_x are learnable weight matrices. The bias $\mathbf{b}_t$ is initialized to 0 (or -1) to bias the network towards passing information through initially.

3. **Highway Aggregation:** The intermediate representation $\mathbf{y}_{mid}$ is a convex combination of the non-linear path and the linear projection, controlled by the gate $\mathbf{t}$:

$$\mathbf{y}_{mid} = \mathbf{t} \odot \mathbf{h} + (1 - \mathbf{t}) \odot \mathbf{x}_s$$

   where $\odot$ denotes element-wise multiplication.

4. **Final Projection:** The output is obtained by applying Dropout and projecting to the target dimension $d_{out} = 32$:

$$\mathbf{y}_{out} = W_{out}(\text{Dropout}(\mathbf{y}_{mid})) + \mathbf{b}_{out}$$

Chrono Initialization

We implemented **Chrono Initialization**, where the bias terms of the forget/update gates are initialized using a log-uniform distribution:

$$b \sim \log(U(1, T_{max}) - 1)$$

With $T_{max} = 1000$, this forces the RNNs to preserve memory over longer horizons immediately at the start of training.

Training Strategy

Loss Function

We utilized the standard **Mean Squared Error** loss function.

Validation Strategy

We employed a standard hold-out validation approach to evaluate model performance.

- **Initial Approach:** For each training run, the last 20% of the data was held out as the validation set.
- **Improved Approach:** After detecting minor overfitting to the public leaderboard, we implemented a shuffled split while preserving the proportion. The data was first shuffled, after which the last 20% of the shuffled set was used for validation.

Optimization Schedule

- **Optimizer:** AdamW with weight decay (0.1) is used to ensure regularization and prevent overfitting on augmented data.
- **Boosting Stages:** Training was performed in sequential stages (e.g., 64 epochs + 31 epochs for GRU). Between stages, the best weights are reloaded, allowing the optimizer to escape local minima.

- **Stochastic Weight Averaging (SWA):** Instead of picking the single best epoch, we implemented a **Last-N SWA** strategy. This approach averages the weights of the final $N = 5$ (or $N = 3$) checkpoints. By smoothing the optimization trajectory, SWA helps the model settle in the center of a flat minimum, leading to better generalization on unseen data and a simpler selection of epochs for training on full data, since training becomes more stable.

Adaptive Meta-Learning (Inference)

Offline Optimization

Initial blending weights $W \in \mathbb{R}^{3\times 32}$ were optimized on the Out-Of-Fold (OOF) predictions from the validation set (20%).

Online Adaptation

During inference, the model adapts to the specific dynamics of the test sequence. After predicting step t, the true state is revealed. We update the ensemble weights for step $t+1$ based on the error at step t:

$$W_{t+1} = W_t \cdot \exp\left(-\eta \cdot (y_{true} - y_{pred})^2\right)$$

Where $\eta = 0.1$ is the adaptation rate. This update is performed **feature-wise**, meaning the model can trust the LSTM for feature i and the GRU for feature j simultaneously.

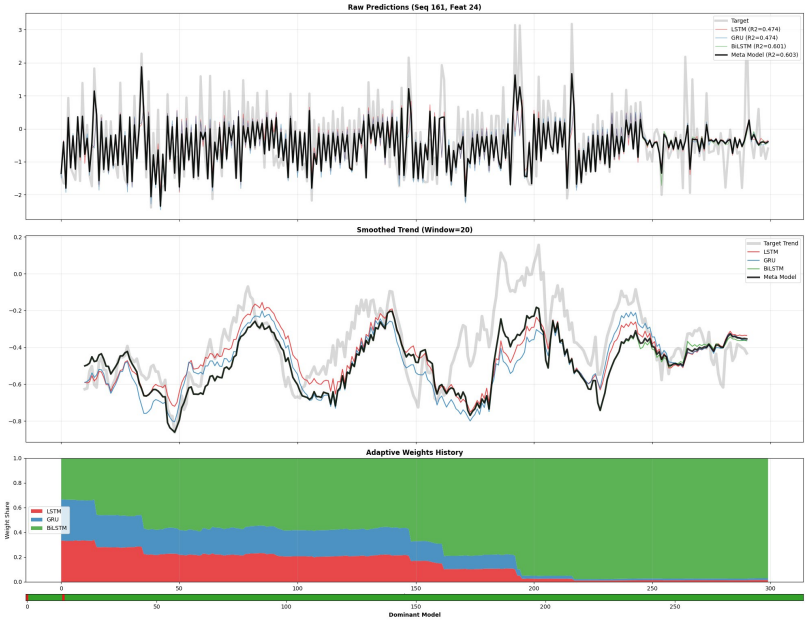

Figure 1: **BiLSTM Dominance.** In this sequence, the BiLSTM model (Green) quickly achieves the lowest error. The Meta-Learner detects this and increases BiLSTM's weight to nearly 1.0, effectively suppressing the noise from LSTM (Red) and GRU (Blue).

Figure 2: **Mixed Regime / BiLSTM Lags.** Here, the signal dynamics favor the uni-directional models. The LSTM (Red) and GRU (Blue) capture the trend better, causing the Meta-Learner to reduce the BiLSTM (Green) contribution significantly compared to the previous case.

Key boosters (critical steps)

To understand the contribution of each component in our solution, we conducted a systematic ablation study on the GRU model. The study reveals which innovations provide meaningful performance gains versus marginal improvements.

Results

Table 3: Ablation Study Results on GRU Model (Validation R^2)

Variant	**R^2 Score**	**Absolute**	**Relative**	**Training Time (s)**
Baseline	0.35639	-	-	371.9
No Highway	0.35523	-0.00116	-0.33%	309.8
No Chrono	0.35581	-0.00058	-0.16%	396.1
Simple Features (id)	0.35594	-0.00045	-0.13%	344.3
Single Boost	0.35587	-0.00052	-0.15%	236.0
No Augmentation	**0.35289**	**-0.00350**	**-0.98%**	223.9
No SWA	0.35640	+0.00001	+0.003%	408.4

Key Findings

Major Performance Driver: Data Augmentation

The most significant performance drop occurred when removing data augmentation (-0.00350 R^2). Our augmentation pipeline provides several benefits:

- **5x Effective Dataset:** Increases training data diversity
- **Variance Normalization:** Aligns sequence statistics across the dataset
- **Controlled Noise Injection:** Improves robustness without corrupting signal

Minor but Consistent Improvements

The remaining components provide smaller but cumulative benefits:

- **Highway Head:** Provides +0.00116 R^2 by enabling adaptive information flow
- **Chrono Initialization:** Adds +0.00058 R^2, helping preserve long-term memory
- **Non-linear Features:** Contributes +0.00045 R^2 via sigmoid/silu transforms
- **Boosting:** Provides +0.00052 R^2 through sequential error correction

SWA Impact

Stochastic Weight Averaging (SWA) showed minimal impact on final R^2 but significantly increased training time. However, SWA provides other benefits:

- More stable training convergence
- Reduced variance in model selection
- Consistent performance across different random seeds

Meta-Learner Analysis

For the adaptive meta-learner, we compared:

- **Static Weights:** $R^2 = 0.35832$ (optimized on OOF predictions)
- **Adaptive Weights:** $R^2 = 0.35866$ (online feature-wise adaptation)

The adaptive approach provides a **+0.00034 R^2** improvement by dynamically reweighting models based on recent errors. While modest, this gain comes at virtually no computational cost during inference.

What Didn't Work

Despite extensive experimentation with various modern architectures and techniques, we discovered that many approaches either failed to improve performance or actively degraded it compared to our final ensemble design. This section documents our unsuccessful attempts, which collectively helped narrow down the solution space.

Alternative Neural Architectures

We tested several state-of-the-art sequence modeling architectures, all of which underperformed relative to the simple RNN ensemble:

- **TCN (Temporal Convolutional Networks):** Despite their strong performance in many time-series tasks, TCNs failed to capture the long-term dependencies in data.
- **Transformer / Attention-based Models:** Surprisingly, attention mechanisms consistently **degraded** performance.
- **Mamba2 (State Space Models):** As a modern alternative to RNNs, Mamba2 showed promise theoretically but failed to deliver competitive results in practice.
- **Window-based LSTM (vs BiLSTM):** Replacing the bidirectional LSTM with a standard unidirectional LSTM trained on rolling windows performed worse. The bidirectional context aggregation appears crucial for the BiLSTM's diversity contribution.
- **Gradient Boosting:** We attempted to use gradient boosting trees (both multi-output and per-feature). None approached the performance of our RNN ensemble.
- **Denoising with VAE:** Preprocessing the data with a Variational Autoencoder to remove noise before RNN training did not help, suggesting the "noise" might contain useful signal or the VAE was removing important dynamics.

Feature Engineering

We experimented extensively with feature engineering beyond the simple activation functions:

- **Rolling Statistics:** Features based on rolling windows (means, standard deviations, maxima, minima) across multiple window sizes did not improve performance.
- **Cross-feature Statistics:** Similarly, statistics computed across feature dimensions (cross-means, cross-stds, etc.) added noise without signal.
- **Other features:** Non-functional features didn't help at all.

Data Augmentation Techniques

Beyond our successful variance normalization approach, many standard time-series augmentations failed:

- **Sign Flipping:** Multiplying sequences by -1 (inverting the signal) proved detrimental, indicating important directional information in the data (or small dataset).
- **Magnitude Warping:** Non-uniform scaling of sequences introduced unrealistic artifacts without improving generalization.
- **Peak Removal:** Manually clipping extreme values to reduce outliers hurt performance, suggesting these "peaks" contain meaningful information rather than being measurement errors.

Architectural Modifications

Several modifications to our core architecture were tested without success:

- **BiLSTM Boosting:** Applying the same boosting strategy to BiLSTM as used for LSTM/GRU did not improve its standalone performance.
- **BiLSTM Augmentation:** Training BiLSTM on augmented data (which works well for stateful RNNs) actually reduced its effectiveness.
- **BiLSTM Highway Head:** Adding the Highway Head to BiLSTM provided no benefit, unlike for LSTM/GRU.
- **Standard Residual Connections:** Simple residual/skip connections between RNN layers did not help, whereas the Highway Head's adaptive gating proved essential.

Training and Optimization

- **Learning Rate Schedulers:** Various schedulers (Cosine, ReduceLROnPlateau) did not improve over our simple constant learning rate with boosting stages.
- **Advanced Meta-Models:** Replacing our simple adaptive weighting with more complex meta-learners (MLPs, GRUs, etc.) consistently performed worse. The online exponential weighting proved surprisingly robust and effective.