Knowledge Base

Santosh Linkha

October 19, 2025

10/19/2025

0.1 GRU class

GRU class takes already scaled data, fits the model using train and val data, and predicts using the input data. This is the optimal code.

10/17/2025

0.2 Original Results

Table 1: Original Metrics

Metric	ARIMA(4,1,4)	ARIMAX(3,1,3)	Linear Regression
RMSE (Root Mean Square Error)	1.1659	0.454	0.450
MAE (Mean Absolute Error)	0.9007	0.3874	**0.374**
MAPE (Mean Absolute Percentage Error)	38.43%	20.70%	**18.85%**
RMSLE (Root Mean Squared Log Error)	0.3309	–	–
sMAPE (Symmetric MAPE)	38.47%	–	–
MBD (Mean Bias Deviation)	-0.6123	–	–
Theil's U Statistic	0.2538	–	–
R-squared (R^2)	0.37	0.36	**0.931**
Adjusted R^2	–	–	0.930
AIC (Akaike Information Criterion)	-33159.74	**-40500.03**	15397.06
BIC (Bayesian Information Criterion)	-33108.55	**-40449.57**	15447.52
MBD (Mean Bias Deviation)	-0.6123	–	–
MASE (Mean Absolute Scaled Error)	–	0.5492 (Test)	–
Training RMSE	–	0.789	–
Training MAE	–	0.697	–
Testing RMSE	–	0.02965	–
Testing MAE	–	0.01759	–
AICc (Corrected AIC)	-30459.53	-41919.23	–
BIC	-30394.66	-41825.54	15447.52
Computation Time (seconds)	1.27	5.38	**0.02**
Best Model Based on Accuracy			**Linear Regression**

0.3 LSTM code for time series forecasting.

Max you should plan for: 3 LSTM layers.
Why 3 (and not more):

- With ~14k daily points and ~10 predictors, **2 layers** usually suffice; **3 layers** is the practical ceiling before vanishing gradients/overfit and training instability dominate.
- Going >**3** rarely helps for macro/market series; gains are typically < 1–2% and cost a lot of stability unless you redesign with residuals/LayerNorm/skip connections—in which case a Transformer or hybrid often wins anyway.

Use this decision rule:

Component	Recommendation	Rationale
Layer 1	128–192 hidden units	Extracts short-term & nonlinear interactions
Layer 2	64–96 hidden units	Consolidates intermediate temporal patterns
Layer 3 *(optional)*	32–64 hidden units	Adds long-term context; use only if underfitting
Dropout	0.25–0.35	Prevents overfitting; tune with validation loss
LayerNorm	Yes	Stabilizes deeper LSTM stacks
Gradient Clipping	1.0	Avoid exploding gradients
Optimizer	AdamW(lr=1e-3, weight_decay=1e-4)	Smooth convergence, mild regularization
Early Stopping	patience=10–15	Prevents overtraining
Sequence Window	60–180 days	Captures 3–6 months of momentum and lag effects
Batch Size	64	Good balance for GPU memory and gradient stability

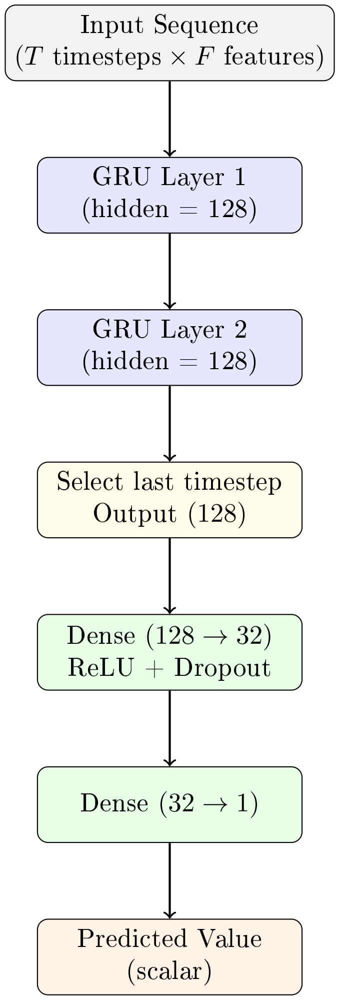
Table 2: LSTM configuration recommendations for time series forecasting.

1. Start with **2 layers** (e.g., 128 $\rightarrow$ 64 units).
2. If you see **clear underfitting** (val loss $\gg$ train loss, both high) after tuning window/LR/units, try **3 layers** (128 $\rightarrow$ 64 $\rightarrow$ 32) with:
   - LayerNorm after each LSTM output
   - Dropout 0.25–0.35
   - Grad clip = 1.0
   - Early stopping (patience 10–15)
3. If 3 layers still underfit, **depth isn't the bottleneck**. Try:
   - Longer/seasonal windows (e.g., 90–180d), lag features
   - Attention on top of LSTM (LSTM+Attention)
   - Hybrid/CNN-LSTM or a Transformer (Informer/Temporal Fusion)

Bottom line: for your dataset, **cap depth at 3**; most runs will pick **2 layers** as the best.

0.4 GRU

Try GRU before attempting LSTM. GRUs are computationally cheaper and often perform comparably to LSTMs, especially on smaller datasets. They have fewer parameters, which can help mitigate overfitting.



0.5 Data Scaling

The data is split into 70%, 15%, and 15% for training, validation, and testing, respectively. StandardScaler from sklearn is used to standardize the features by removing the mean and scaling to unit variance based on the training set statistics. The same scaler is then applied to the validation and test sets to ensure consistency.

0.6 Results

The following training data was obtained from the initial implementation of the GRU model:

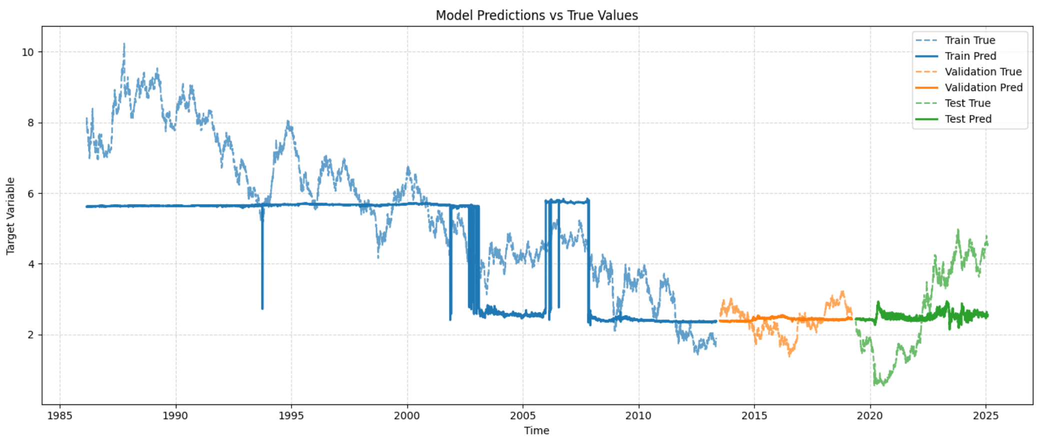


Figure 1: GRU initial test Results

Table 3: Model Performance Summary (Train, Validation, Test)

	Train	Validation	Test
ME	-0.9174	0.0362	-0.1128
MAE	1.2891	0.3542	1.2160
RMSE	1.5852	0.4279	1.3378
MAPE (%)	23.0996	16.1259	72.5879
sMAPE (%)	25.8694	15.0534	50.5200
R2	0.3572	-0.1246	0.0244
AIC	349787.1355	314822.7608	319570.4453
BIC	1475157.0000	1196026.0000	1200849.0000

Fine Turning Hyperparameters

Table 4: Summary of GRU Hyperparameters and Example Optuna Search Space

Hyperparameter	**Description**	**Typical Range / Choices**	**Comments / Optuna Search Example**
Model Architecture			
`hidden_dim`	Size of the GRU hidden state (neurons per layer)	32–512 (commonly 64–256)	Larger values increase capacity but risk overfitting. *Optuna:* trial.suggest_categorical("hidd [64, 128, 256])
`num_layers`	Number of stacked GRU layers	1–3 (sometimes up to 4)	Deeper networks capture more complex temporal dependencies. *Optuna:* trial.suggest_int("num_layers 1, 3)
`dropout`	Fraction of units dropped during training	0.0–0.5	Regularizes the model and prevents overfitting. *Optuna:* trial.suggest_float("dropout", 0.1, 0.5)
`bidirectional`	Use bidirectional GRU	{True, False}	Useful if future context is available (e.g., offline inference).
`activation`	Activation function in fully connected layer	{ReLU, LeakyReLU, tanh}	Affects learning dynamics.
`fc_layers / fc_dims`	Hidden size(s) of dense head	[32], [64, 32], etc.	Defines how GRU outputs are mapped to predictions.
Training Hyperparameters			
`learning_rate`	Step size of the optimizer	10^{-5}–10^{-2} (log scale)	Most critical hyperparameter for convergence. *Optuna:* trial.suggest_float("lr", 1e-4, 1e-2, log=True)
`optimizer`	Optimization algorithm	{Adam, AdamW, RMSProp, SGD}	Adam or AdamW are standard.
`weight_decay`	L2 regularization factor	0–10^{-3}	Helps generalization. *Optuna:* trial.suggest_float("weight_de 0.0, 1e-3, log=True)
`batch_size`	Number of samples per gradient update	16–256	Larger batch = smoother gradients but slower. *Optuna:* trial.suggest_categorical("batc [32, 64, 128])
`num_epochs`	Number of training epochs	30–200	Use early stopping for efficiency.
`scheduler`	Learning rate adjustment strategy	{StepLR, CosineAnnealing, OneCycleLR}	Can improve convergence.
Data and Sequence Parameters			
`sequence_length`	Number of time steps fed into GRU	10–200	Longer sequences capture more context but increase computation. *Optuna:* trial.suggest_categorical("seq_ [30, 60, 90])
`forecast_horizon`	Prediction step ahead	1–N	Determines how far the model predicts into the future.
`normalization`	Data scaling method	{StandardScaler, MinMaxScaler}	Required for stable training.
Regularization and Miscellaneous			
Regularization	Dropout, Weight Decay, Gradient Clipping	–	Prevents overfitting and improves stability.
Early Stopping	Monitor validation loss	–	Stops training when validation loss stops improving.
Pruning (Optuna)	Trial pruning	–	Stops poor trials early to save computation.
Random Seed	Fixed random initialization	–	Ensures reproducibility.