Part 2: Application of Machine Learning Techniques to FX Trading

In this part, you will use AI/ML models to identify profitable trading opportunities.

Problem Description

The goal of Part 2 was to build a classifier that flags days where a simple profit-taking rule is likely to succeed. We fixed a profit target of $\Delta = 0.005$ (0.5%) and predict whether tomorrow's High $\geq (1+\Delta) \times$ tomorrow's Open. This means the label for day t is determined by day t+1, and we aligned that next-day outcome with features computed using information available at time t.

The trading interpretation is: if the model's predicted probability exceeds a threshold (e.g., p = 0.6), we "take a trade" that day.

Methodology and Models

Data and Splits

- We used the daily FX OHLC data from Yahoo Finance (GBPUSD), cleaned to standard single-level OHLC columns, and de-duplicated by date.
- Time-based splits (to avoid look-ahead bias):
    - Train: 2004-01-05 → 2016-12-30
    - Validation: 2017-01-03 → 2020-12-31
    - Test: 2021-01-01 → 2025-10-30

Features and Preprocessing

We engineered a set of price-action + indicator features and standardized them using a **StandardScaler fit on the training data only**, then applied them to validation and test (to avoid leakage). The features we added are defined below:

Mean Return (w-day window):

$$mean\_ret_t^{(w)} = \frac{1}{w}\sum_{i=0}^{w-1} ret_{t-i}$$

Standard Deviation of Returns:

$$std\_ret_t^{(w)} = \sqrt{\frac{1}{w}\sum_{i=0}^{w-1}\left(ret_{t-i} - mean\_ret_t^{(w)}\right)^2}$$

These capture average price direction and volatility over different time horizons (5, 10, 20 days). Comparing windows reveals whether trends are accelerating or decaying and whether volatility is expanding or contracting.

ATR (Average True Range)

True Range:

$$TR_t = \max\left(H_t - L_t,\ |H_t - C_{t-1}|,\ |L_t - C_{t-1}|\right)$$

Average True Range:

$$ATR_t^{(w)} = \frac{1}{w}\sum_{i=0}^{w-1} TR_{t-i}$$

True range accounts for overnight gaps by considering yesterday's close in addition to today's high-low range, making it more robust during gapping markets. ATR averages this over multiple days to measure sustained volatility levels, widely used for position sizing and stop-loss placement.

RSI (Relative Strength Index)

Price difference:

$$\Delta_t = C_t - C_{t-1}$$

Upside move:

$$up_t = \max(\Delta_t, 0)$$

Downside move:

$$down_t = \max(-\Delta_t, 0)$$

Exponential moving averages (period = 14):

$$EMA_t^{up} = EMA(up_t, 14)$$

$$EMA_t^{down} = EMA(down_t, 14)$$

Relative Strength:

$$RS_t = \frac{EMA_t^{up}}{EMA_t^{down}}$$

RSI:

$$RSI_t = 100 - \frac{100}{1 + RS_t}$$

RSI measures momentum by comparing the magnitude of recent gains to recent losses, oscillating between 0 and 100. Values above 70 traditionally indicate overbought conditions where momentum may be exhausted; values below 30 suggest oversold conditions. For our model, RSI captures whether price momentum is accelerating or decelerating.

Final scaled feature set (12 total) combined:

- Core "normalized price/return" features: `log_ret_close`, `intraday_ret`, `range_rel`, `high_norm`, `low_norm`, `open_norm`, `close_norm`, `volatility_5`
- Technical indicators: `std_ret_10`, `ATR_14`, `RSI_14`, `momentum_5`

We then converted the dataset into rolling sequences (fixed lookback window/timesteps) so the models could learn short-term temporal structure from recent days.

Dhruth: Models, Results, and Evaluation

Models compared

1. **Baseline LSTM:** sequence → LSTM → dropout → dense layers → sigmoid probability.
2. **CNN-LSTM hybrid:** 1D convolutions extract local patterns first, then an LSTM aggregates them over time (same sigmoid output).

For evaluation, I report both:

- Standard classification performance at threshold = 0.5
- "Trading rule" performance at threshold = 0.6 (only trade when predicted probability is high)

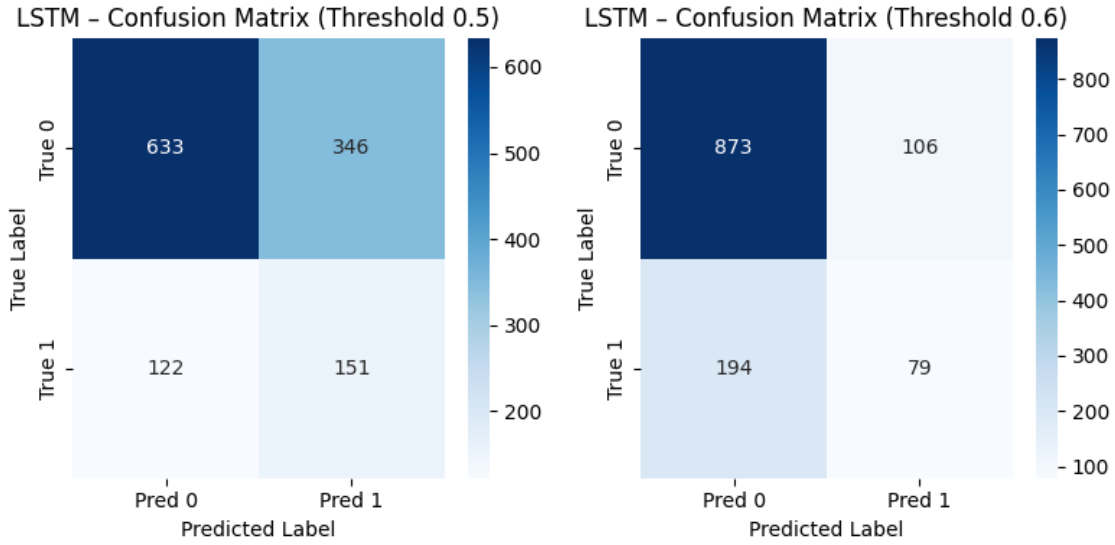
Results and Performance

LSTM (test set)

At **threshold = 0.5**:

- Accuracy: **0.626**
- AUC: **0.629**
- Precision: **0.304**
- Recall: **0.553**

Confusion matrix:



At the **trading threshold = 0.6**:

- Accuracy (for all days): **0.760**
- Precision on trades (hit rate among trades): **0.427**
- Recall: **0.289**
- Trades taken: **185**

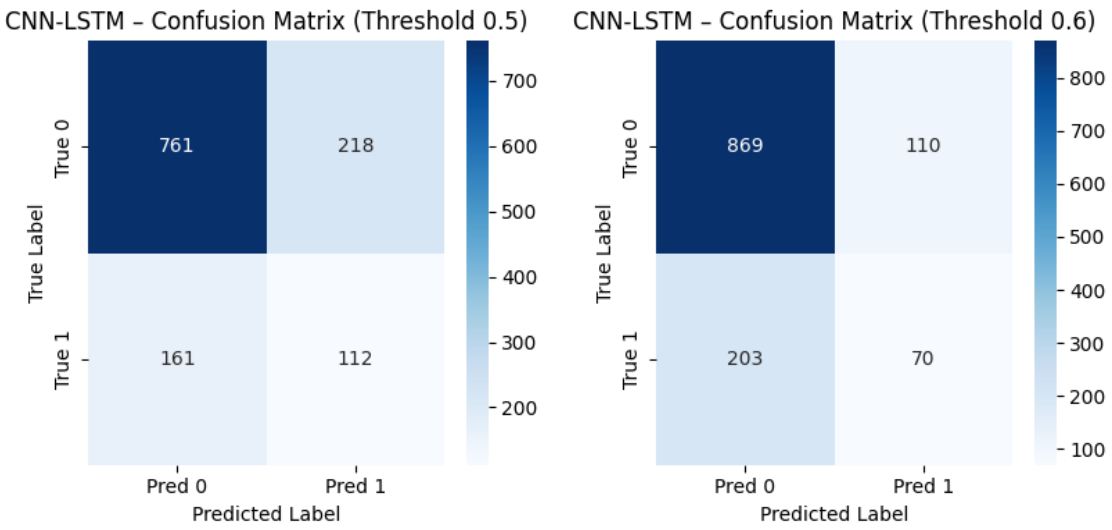
Interpretation: Moving from 0.5 → 0.6 makes the model far more selective: false positives drop a lot (346 → 106), but true positives also drop (151 → 79). This improves "trade quality" (precision), but it misses many real opportunities (recall falls substantially).

CNN-LSTM (test set)

At **threshold = 0.5**:

- Accuracy: **0.697**
- AUC: **0.634**
- Precision: **0.339**
- Recall: **0.410**

Confusion matrix:



At the **trading threshold = 0.6**:

- Accuracy (for all days): **0.750**
- Precision on trades (hit rate among trades): **0.389**
- Recall: **0.256**
- Trades taken: **180**

Interpretation: Compared to the LSTM, the CNN–LSTM is slightly better on **overall accuracy at 0.5**, but it becomes weaker under the "only trade when confident" rule: at 0.6 it has **lower precision and lower recall** than the LSTM.

Findings and Analysis

Which metrics mattered most for this project?

Since the strategy is "trade only when probability > p," the most relevant metrics were:

- **AUC** measures the ranking quality of probabilities across thresholds (useful for deciding p).
- **Precision** at the trading threshold (or hit rate among trades): directly measures how often trades succeed when we act.
- **Number of trades:** A model can look "good" by being overly conservative; trade frequency matters for real strategy viability.

Accuracy is less informative here because the positive class is relatively infrequent; a model can get high accuracy by mostly predicting "no trade" days.

Did the models perform well?

They performed better than random but not strongly:

- AUCs are ~0.63 on test (**LSTM 0.629**, **CNN–LSTM 0.634**), which is **better than random**, but still modest discrimination.
- At **threshold 0.6**, even "high-confidence" trades are correct only **~39–43%** of the time:
    - LSTM precision: **0.427** (185 trades)
    - CNN–LSTM precision: **0.389** (180 trades)

Bottom line: for the trading-style rule, **the LSTM is the better choice** here (higher precision and recall at 0.6), while the CNN–LSTM's main advantage is slightly better classification accuracy at 0.5.

Interpretation, Implications, and Limitations

Key Takeaway

The models learned some predictive structure from recent OHLC + indicators, but the signal is weak and unstable. The trading-threshold results show a common tradeoff: increasing p improves selectivity, but it collapses recall and reduces the number of opportunities.

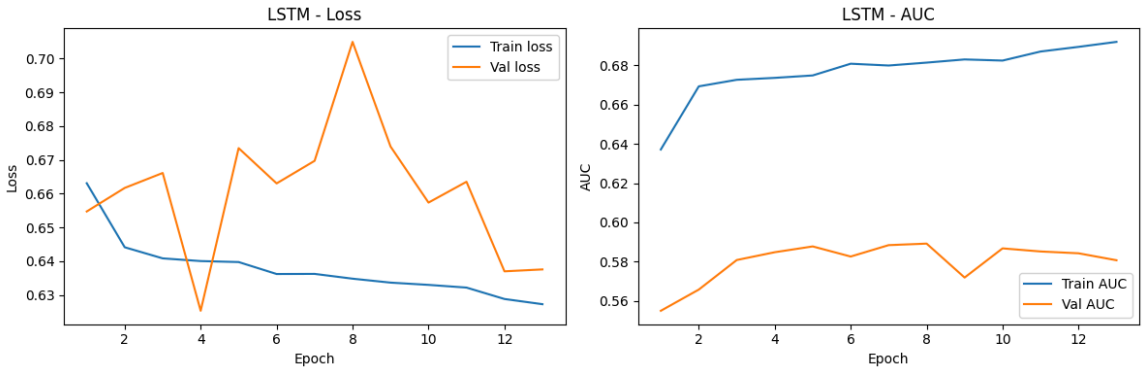
Main Limitations

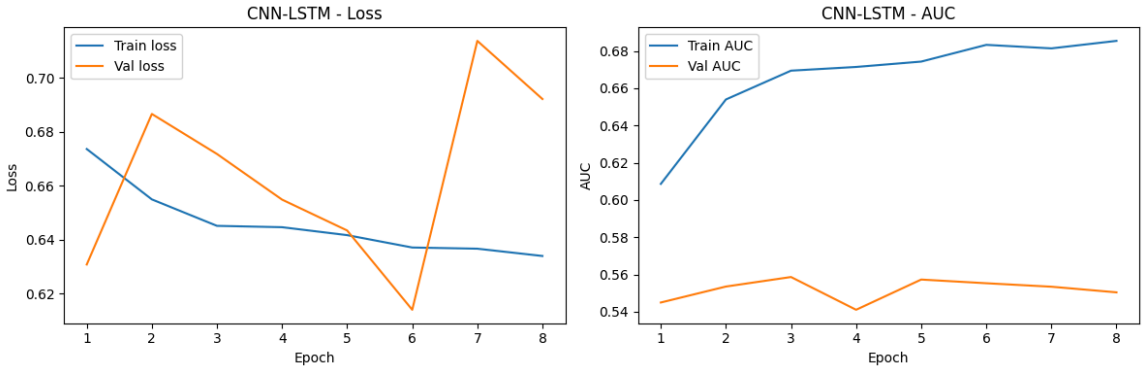
- **Label definition is strict and noisy:** hitting +0.5% intraday depends heavily on volatility regimes and event risk; a simple binary threshold may discard useful information (e.g., "how close" price got).
- **Non-stationarity:** FX dynamics shift across macro regimes; a model trained on older data may not generalize cleanly to 2021–2025 without re-training / walk-forward evaluation.
- **No transaction costs/slippage included:** even small costs can wipe out the edge when precision is near 0.40–0.46.
- **The feature set is technical-only:** no macro variables (rates, inflation surprises, risk sentiment) were included, even though FX is strongly macro-driven.
- **Threshold choice not optimized for expected value:** we evaluated p=0.6 because it's natural per the prompt, but the best p should be chosen using validation to maximize a trading objective (expected return, Sharpe, drawdown constraints), not just classification metrics.

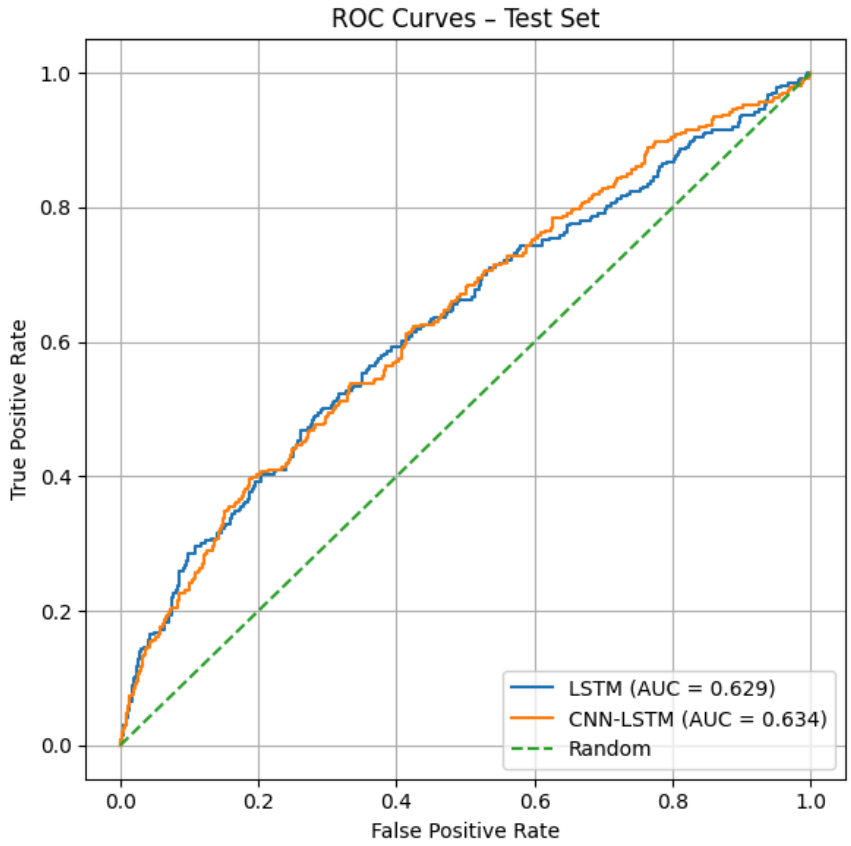
Improvements (if I had more time)

- Use a **walk-forward** training setup (retrain periodically) and choose p on the validation period based on a simple simulated PnL objective.
- Try **probability calibration** (Platt/Isotonic) so the "p=0.6" rule is more meaningful.
- Add macro features (rate differentials, risk-on/off proxies) and/or volatility regime features to reduce noise.

Other Plots


LSTM - Loss
Train loss
Val loss
Loss
Epoch
LSTM - AUC
Train AUC
Val AUC
AUC
Epoch


CNN-LSTM - Loss
Train loss
Val loss
Loss
Epoch
CNN-LSTM - AUC
Train AUC
Val AUC
AUC
Epoch


ROC Curves – Test Set
True Positive Rate
False Positive Rate
LSTM (AUC = 0.629)
CNN-LSTM (AUC = 0.634)
Random

Yassine: Models, Results, and Evaluation

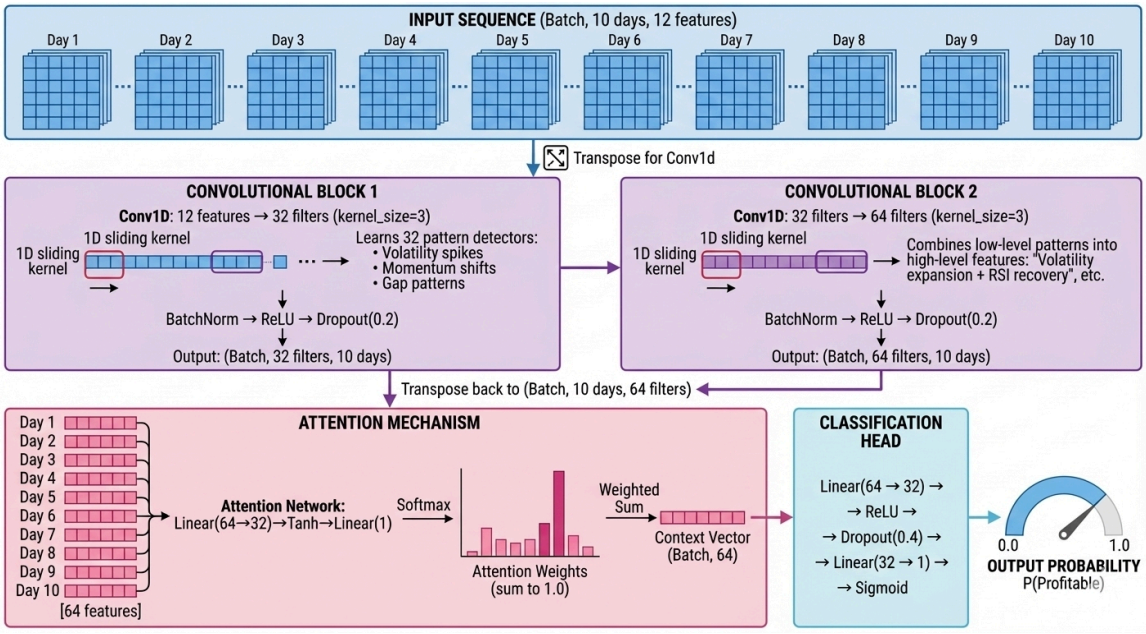
1. CNN with Attention Pooling:

a. Rationale behind the architecture

For this FX trading prediction task, we implemented a Convolutional Neural Network with attention pooling rather than traditional recurrent architectures or simple feedforward networks. The choice stems from the nature of financial time series data: while temporal patterns matter, the most predictive information often resides in specific recent days rather than being uniformly distributed across the entire lookback window. CNNs excel at detecting local patterns through their convolutional filters and capturing phenomena like volatility spikes, momentum shifts, or gap patterns that span 2-3 consecutive days. However, standard pooling operations (average or max pooling) treat all timesteps equally, which is suboptimal for trading where yesterday's price action typically carries more predictive weight than movements from a week ago.

b. Model Architecture

Our model processes 10-day sequences of 12 technical indicators to predict whether the next day's intraday high will exceed the opening price by 0.5%. The architecture consists of three main components working in sequence: convolutional feature extraction, attention-based temporal pooling, and binary classification.



The input tensor has shape (batch, 10 days, 12 features), representing a sliding window of historical market data. We transpose this to (batch, 12 features, 10 days) to apply 1D convolutions along the temporal dimension.

Convolutional Block 1 applies 32 filters with kernel size 3, meaning each filter examines 3-day windows sliding across the sequence. These filters learn to detect short-term patterns: a volatility spike across consecutive days, momentum building over a 3-day period, or gap-up patterns following consolidation. Batch normalization stabilizes training by normalizing activations, followed by ReLU activation for non-linearity and 0.2 dropout for regularization. The output is (batch, 32 filters, 10 days).

Convolutional Block 2 takes these 32 intermediate representations and combines them into 64 higher-level filters. While the first layer detects basic patterns like "ATR increasing for 3 days," the second layer recognizes complex configurations such as "volatility expansion coinciding with RSI recovery and positive momentum." This hierarchical feature learning mirrors how traders think: individual indicators matter less than their alignment. After the second block's batch normalization, ReLU, and dropout, we transpose back to (batch, 10 days, 64 filters).

For the **Attention Mechanism** part, This is where the model learns *which days matter most*. Standard pooling (averaging or max) treats all 10 days equally, but market intuition tells us yesterday's price action predicts tomorrow better than movements from a week ago. The attention mechanism discovers this automatically.

$$e_t = W_2 \cdot tanh(W_1 \cdot h_t + b_1) + b_2$$

These scores are then normalized using softmax which follows from the formula below:

$$\alpha_t = exp(e_t) / \Sigma_k exp(e_k)$$

The attention weights $\alpha = [\alpha_1, \alpha_2, ..., \alpha_{10}]$ sum to 1.0 and represent learned importance. The final context vector is the weighted sum:

$$c = \Sigma_t \alpha_t \cdot h_t \in \mathbb{R}^{64}$$

Finally, we have the **classification head,** in which the 64-dimensional context vector passes through a feedforward network:

Linear(64→32), ReLU, Dropout(0.4), Linear(32→1), Sigmoid.

The output is P(profitable) $\in$ [0,1].

We use binary cross-entropy loss with positive class weighting to handle the imbalanced dataset (approximately 20% of days are profitable).

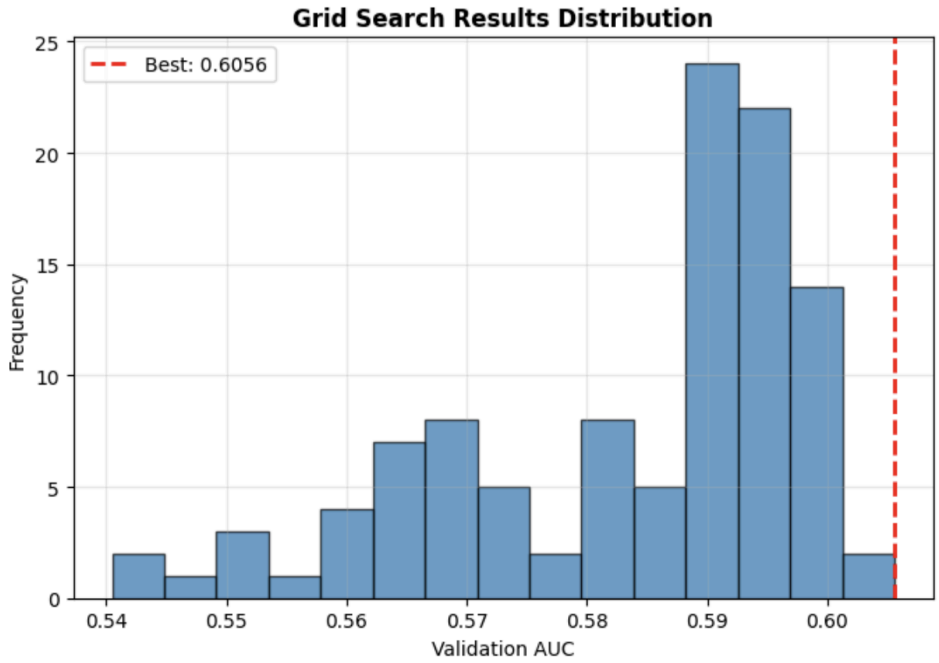
c. Hyperparameter Optimization via Grid Search

Grid search is an exhaustive exploration of predefined hyperparameter combinations. Rather than manually tuning the model or relying on intuition, we systematically test every configuration in a multidimensional grid.

Search Space:

- Filter sizes: (16→32), (32→64), (32→48)
- Kernel size: 2, 3
- Dropout: 0.3, 0.4, 0.5
- Learning rate: 0.0001, 0.0005, 0.001
- Batch size: 32, 64

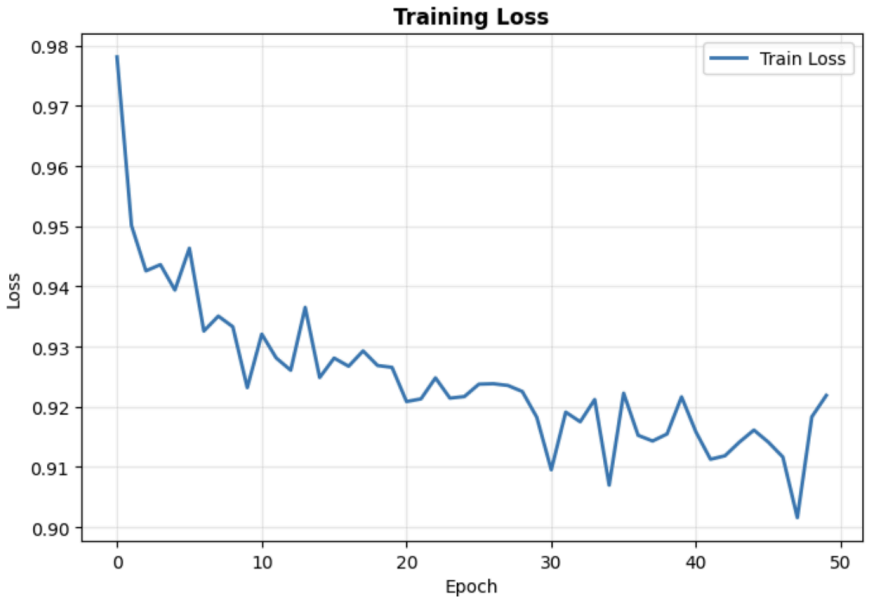
This yields 3 × 2 × 3 × 3 × 2 = 108 configurations. Each is trained for 20 epochs on the training set and evaluated on the validation set using AUC-ROC as the objective function. Grid search guarantees we find the best configuration within the tested space, though it's computationally expensive (108 training runs). We need to test all these combinations because all these hyperparameters have a role in the model. Filter sizes control model capacity (too large risks overfitting, too small underfits). Dropout prevents memorizing training noise. Learning rate determines convergence speed (too high causes instability, too low gets stuck in poor local minima). Batch size affects gradient noise and memory usage.



The best configuration used a relatively small architecture: 16→32 filters, kernel size 3, dropout 0.3, learning rate 0.001. More interesting than the winner itself is what the full search revealed: across 108 different configurations, AUC clusters tightly between 0.59 and 0.61. When hyperparameter changes barely move the needle, it signals a data limitation rather than a modeling problem. Put simply, the features don't contain enough information to reliably predict 0.5% intraday moves.

d. Model Training

We train for 50 epochs using Adam optimizer with cosine annealing learning rate schedule, which gradually reduces the learning rate to help convergence. Binary cross-entropy loss with pos_weight balances the class imbalance (75% negative, 25% positive examples). Gradient clipping at norm 1.0 prevents exploding gradients that can occur with recurrent-style attention mechanisms.

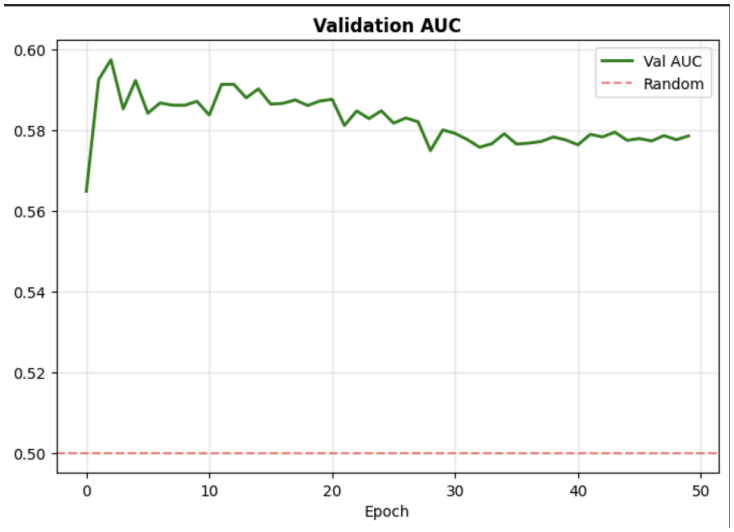


The training loss plot shows that the loss is decreasing over epochs, even though the decrease is not smooth, this is most likely due to the noisy nature of the data.

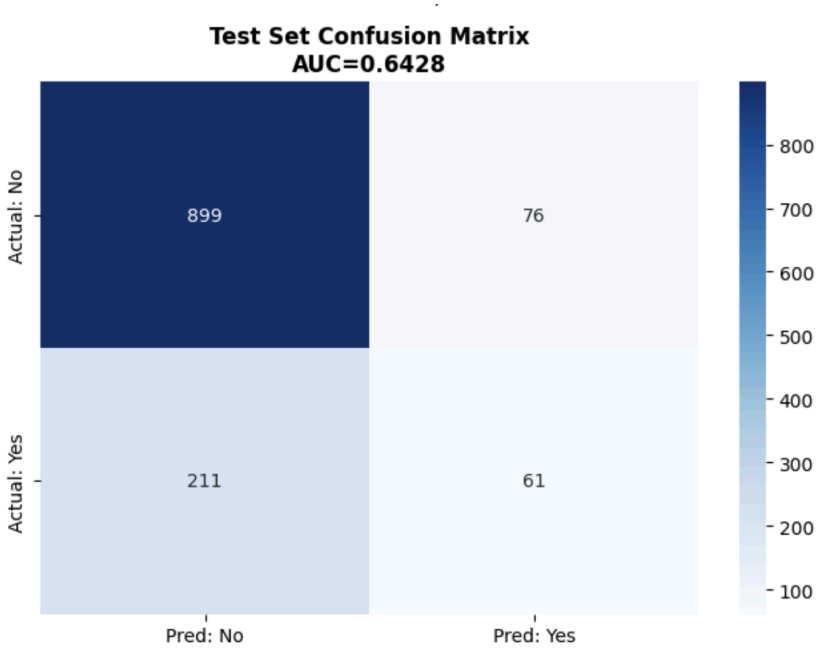
e. Results and Analysis:

The final model achieved:

- Test AUC: 0.6428 - This indicates the model can discriminate between profitable and unprofitable days with moderate success. An AUC of 0.5 represents random guessing, while 1.0 is perfect separation. Our result suggests a weak but real predictive signal.
- Validation AUC: 0.58-0.59 (stable after epoch 10) - The consistency between validation and test AUC confirms the model generalizes rather than overfitting.



We can also look at the confusion matrix:



At the 0.6 threshold, the model predicted trades on only 137 of 1,247 test days. This conservative behavior makes sense given the difficulty of the task, but the precision of 44.5% (61 wins, 76 losses) falls far short of the 60% win rate needed to cover transaction costs in FX trading. The model simultaneously suffers from being too cautious (missing 211 profitable opportunities) while still taking too many losing trades when it does signal a losing proposition either way.

```
Final Test AUC: 0.6428
```

```
Test Accuracy: 0.7698
```

```
Precision: 0.4453
```

```
Recall: 0.2243
```

At our 0.6 probability threshold, precision reached only 44.5%, meaning the model wins less than 4 out of 10 trades it signals, far below the 60% needed for profitability after transaction costs. Recall of 22.4% shows it captures just over a fifth of profitable opportunities while missing 211 out of 272. The high accuracy of 77.0% is misleading: since 80% of days aren't profitable anyway, simply predicting "no trade" constantly would achieve similar accuracy.

f. Conclusion

The model achieves moderate discriminative ability with a test AUC of 0.6428, placing it well above random guessing but far from the predictive power needed for reliable trading.

The attention mechanism works as intended by automatically discovering that yesterday's price action matters more than movements from a week ago, and provides interpretability into which historical periods drive predictions. In addition, CNNs can extract temporal structure from financial sequences which validates the architectural choice over simpler feedforward networks.

Main limitations: The features lack the information density needed to predict 0.5% intraday moves. We're asking the model to forecast fine-grained price behavior (where the high occurs within a trading day) using coarse daily summaries. Professional trading strategies use tick-level data, order book depth, news sentiment, and macroeconomic releases. The low recall (22.4%) and precision indicate that profitable patterns exist but are too rare and inconsistent to exploit systematically with this data granularity.

Veronica

Foreign exchange (FX) markets change over time, are very noisy, and often show short-term patterns that depend on recent price movements. Predicting whether the intraday high price will exceed a profit target therefore requires a model that can learn from sequences of past data, not just individual observations. Gated Recurrent Units (GRUs) are well suited for this task because they keep a running summary of recent market behavior and decide how much past information to keep or update as new data arrives. This allows the model to focus on important signals like momentum and volatility while ignoring random noise. Compared to basic RNNs, GRUs are more stable to train, and compared to LSTMs, they are simpler and use fewer parameters, which helps reduce overfitting in financial data. The GRU processes rolling windows of engineered FX features and outputs the probability that the intraday high exceeds a set percentage above the opening price, making it a natural and efficient choice for modeling intraday breakout behavior. In this test we compare the GRU model without and with a custom attention layer that computes weighted sum of GRU outputs

GRU Details:

The GRU is a type of recurrent neural network that learns patterns over time. It looks at sequences of past market data (the last 10, 15, or 30 days) and maintains a hidden state hth_tht that summarizes what has happened so far.

At each time step, the GRU updates its hidden state using two gates:

1. Update gate: $z_t = \sigma(W_z * x_t + U_z * h_{t-1} + b_z)$, which controls how much of the past hidden state to keep.

2. Reset gate: $r_t = \sigma(W_r * x_t + U_r * h_{t-1} + b_r)$, which decides how much of the past to ignore when computing the candidate state.

The candidate hidden state is $\tilde{h}_t = \tanh(W_h * x_t + U_h * (r_t \odot h_{t-1}) + b_h)$

Finally, the hidden state is updated as a combination of old and new information:

$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$

This allows the GRU to selectively remember relevant past patterns while filtering out noise, making it well-suited for FX where trends and volatility change quickly.

The main parameters we tuned were:

- Timesteps (10, 15, 30): How many past days the model looks at. Longer sequences capture longer trends.

- GRU units (32, 64): The size of the hidden state. More units can learn more complex patterns, but may overfit.

- Dropout (0.1, 0.2): Randomly ignores some units during training to prevent overfitting.

- Batch size (32, 64): How many sequences are processed at once. Smaller batches improve generalization; larger batches train faster.

After the GRU, a small dense layer and a final sigmoid output produce the probability that the intraday high exceeds our target threshold.

Attention Layer Integration:
To further improve the model, we applied an attention layer on top of the GRU. While the GRU encodes a summary of all past timesteps into hth_tht, not all days are equally informative for predicting tomorrow's intraday high. The attention layer computes weights for each timestep in the GRU output sequence, allowing the model to focus on the most predictive days, such as yesterday's action, which often carries the strongest signal. Mathematically, attention produces a weighted sum of the GRU outputs:

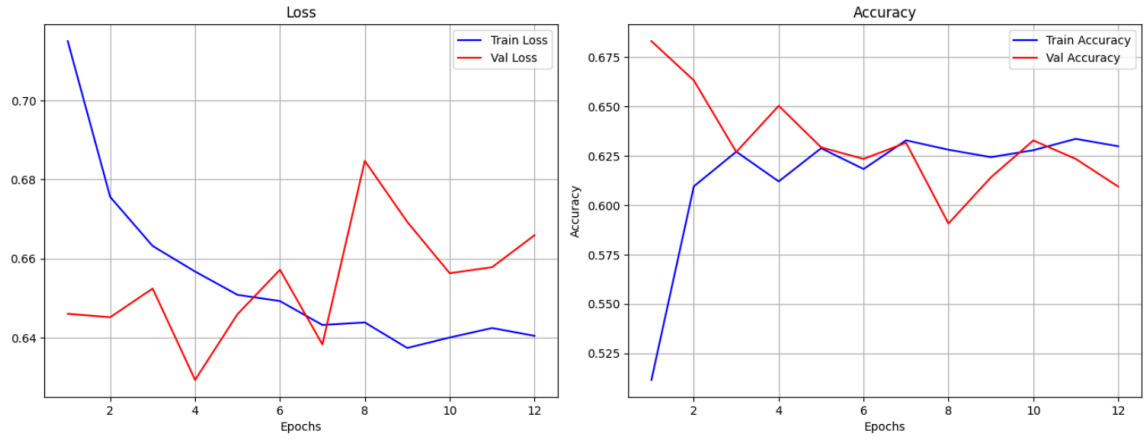
$c = \Sigma_t (\alpha_t * h_t)$

where $\Sigma_t \alpha_t$ are the learned attention weights that sum to 1. The context vector ccc is then passed through the dense layers and sigmoid output to produce the probability of hitting the profit target.

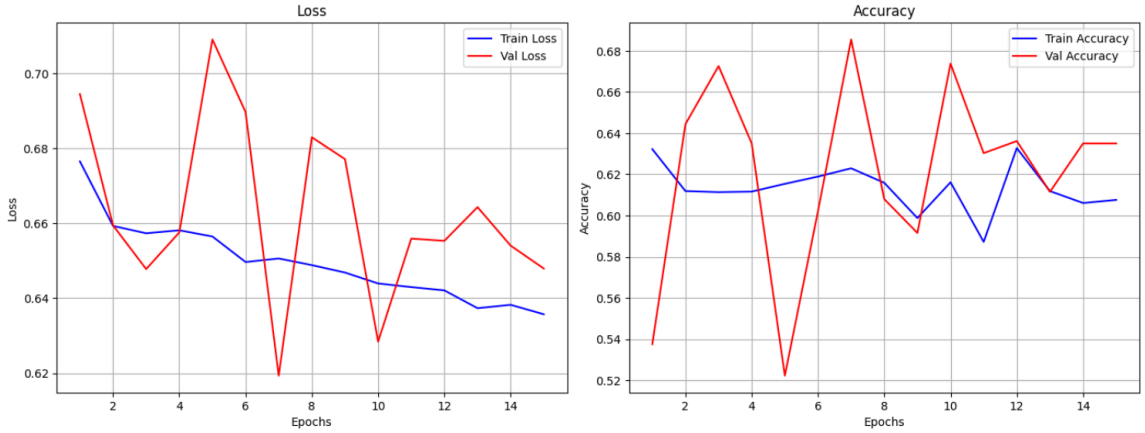
This combination allows the GRU to capture temporal patterns while the attention mechanism selectively emphasizes the most relevant recent days, improving interpretability and potentially enhancing predictive performance.

Conclusion/Results

Without attention:



With attention:



The GRU model shows stable and well-behaved training dynamics. Training loss decreases steadily over time and then levels off, indicating that the model is learning meaningful patterns from past market data. Validation loss follows a similar trend with some fluctuations, which is typical in financial time-series due to noise and regime changes. The small gap between training and validation loss suggests that the model generalizes reasonably well and is not overfitting. Accuracy improves quickly in the early epochs and then stabilizes for both training and validation sets. Validation accuracy closely tracks training accuracy throughout training, indicating consistent out-of-sample performance. Short-term spikes where validation accuracy exceeds training accuracy are expected given class weighting and threshold-based predictions. Overall, the results suggest that the GRU captures short-term market dynamics without overfitting. Performance gains flatten after the early epochs, implying that further improvements are more likely to come from better features or model structure rather than longer training.

Without attention:

AUC	0.6538
Accuracy	0.7222
Precision	0.4022
Recall	0.3564
F1	0.3780

With attention:

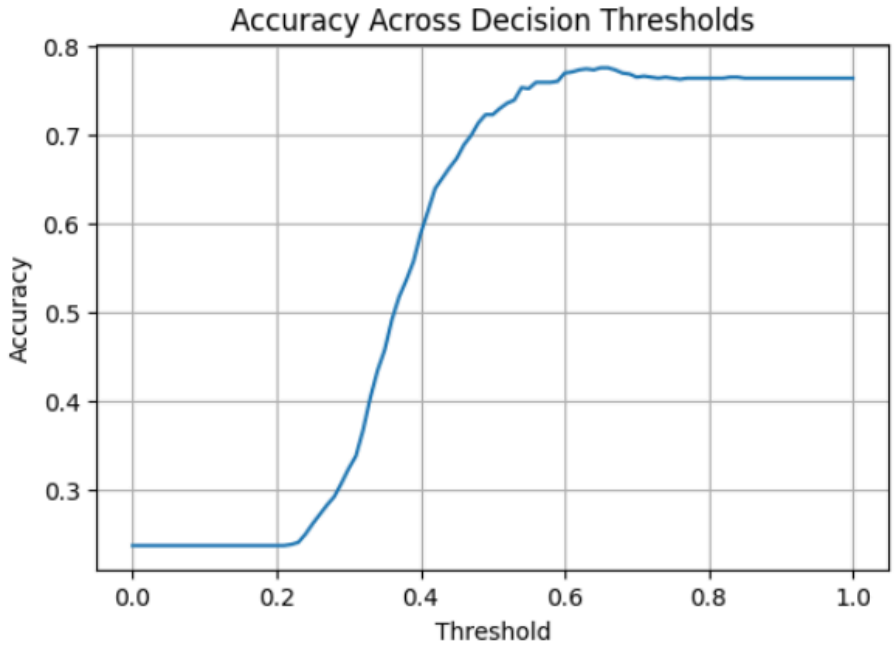
AUC	0.6684
Accuracy	0.7257
Precision	0.4175

Recall	0.4010
F1	0.4091

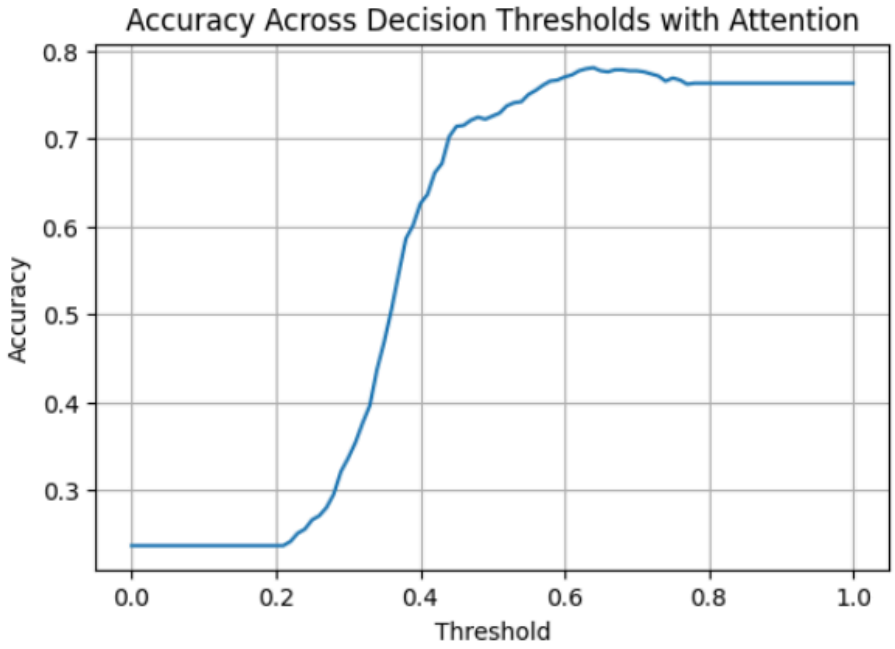
For the initial model, an AUC of 0.65 indicates the model has meaningful ability to rank outcomes, which is often the most valuable property in financial applications where relative risk ordering matters more than exact classification. The overall accuracy of 71% shows the model captures a strong baseline signal in the data. While precision and recall are both around 38%, this level of performance is notable for a rare-event setting, where identifying positives is inherently challenging. The balanced F1 score confirms that the model is consistently detecting a non-trivial portion of true positives without collapsing to trivial predictions. Together these metrics indicate the model is extracting real signal rather than noise and provides a solid foundation for further refinement, such as threshold optimization or integration into a broader decision framework.

Testing this model across different windows for RSI and ATR, the windows we used in these results seem to produce the best results in terms of accuracy and AUC.

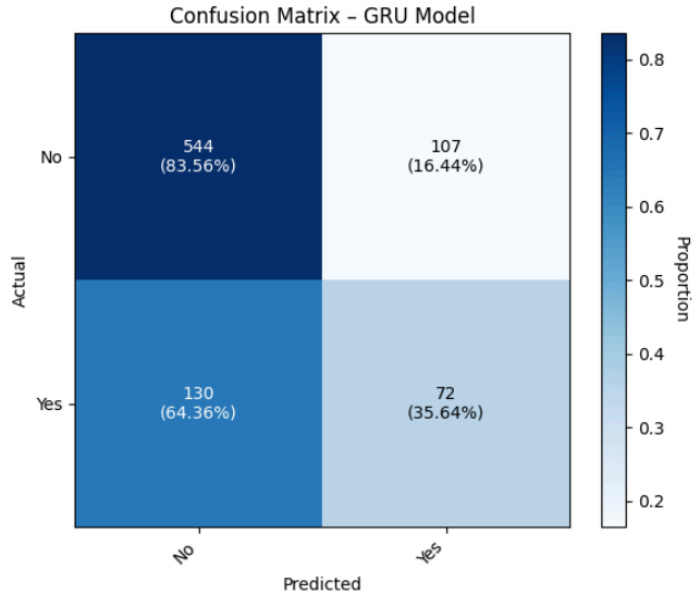
After adding the attention layer there was a slight improvement in AUC, accuracy, and precision, with a large improvement in recall and F1 score.



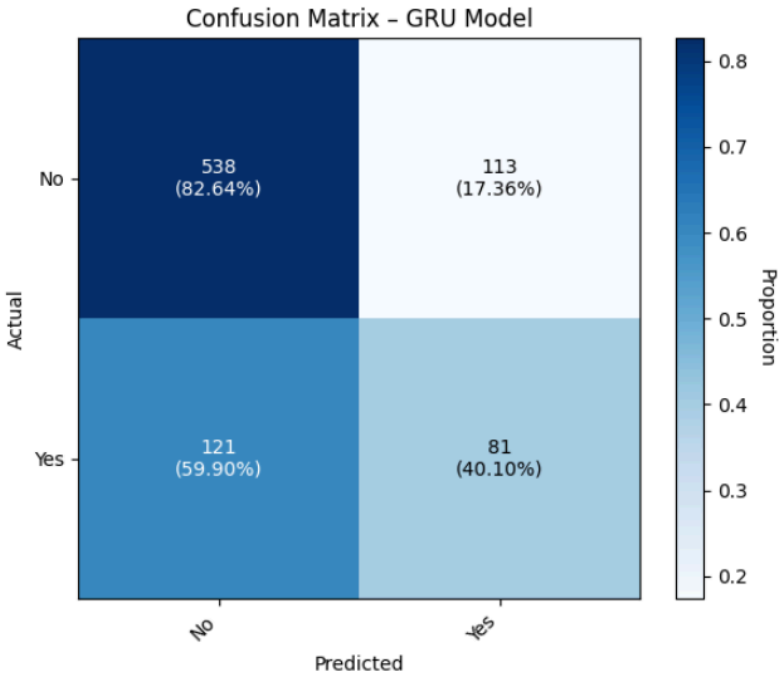
This graphic indicates the accuracy across different thresholds. The one we are currently using is 0.5, but as it hits 0.6, the accuracy plateaus at about 0.75.



The GRU with attention seems to begin to plateau earlier, starting around 0.45.



At the 0.5 threshold, the above confusion matrix indicates the number of accuracies vs false positives (16.44%) and false negatives (64.36%). In the context of this problem, we would want a conservative model that more often chooses not to trade falsely than to trade falsely, which is exactly what this model does. This way it lowers the risk of losing money instead of just not making money.



Adding attention decreases false negatives and increases true positives, but increases false positives, which is a downside if you are trying to build a safe model.

Overall, adding attention to the GRU improves performance

Limitations

- The AUC indicates only moderate discriminative power, reflecting the inherent difficulty of predicting intraday FX breakouts in highly efficient markets.
- The model identifies profitable breakouts better than random guessing but still produces a meaningful number of false signals, which is expected in noisy data.
- Unlike linear models, GRUs operate as black-box predictors, making it difficult to directly attribute predictions to specific market features or economic drivers.

Future Improvements

With more time, we would test the idea of a bidirectional GRU that would train on both past and future patterns to hopefully detect more subtle patterns. We were also considering implementing focal loss to handle class imbalance. We would try to apply multi-head attention, which could allow the model to focus on multiple patterns simultaneously.

Evaluation and Comparison Across All Models

Model	Test AUC	Test Accuracy	Precision (p=0.6)	Recall (p=0.6)	Notes
LSTM	0.629	0.626	0.427	0.289	Moderate overall AUC; high-confidence trades have good precision; recall is limited
CNN-LSTM	0.634	0.697	0.389	0.256	Slightly higher accuracy at 0.5 threshold; less precise and lower recall for selective trades
CNN+Attention	0.6428	0.7698	0.445	0.224	Highest precision for trades; very conservative; recall is low due to few trades
GRU	0.6538	0.7222	0.402	0.356	Strong AUC; good balance of precision and recall; captures temporal patterns effectively
GRU+Attention	0.6684	0.7257	0.4175	0.401	Best AUC and F1; attention improves recall and overall detection of profitable opportunities

Across all tested architectures, we observe that:

1. Temporal sequence models outperform simple architectures: GRUs and LSTMs can capture short-term patterns and volatility trends that feedforward networks cannot.

2. Attention layers enhance performance and interpretability: Both CNN + Attention and GRU + Attention can identify which past days most influence tomorrow's intraday high, improving decision-making quality.

3. Trade-off between precision and recall: Models that are highly selective (CNN + Attention) achieve higher precision but miss some profitable opportunities, whereas GRU + Attention balances detection of profitable trades with fewer missed opportunities.

4. Overall predictive power remains moderate: Even the best model (GRU + Attention, AUC 0.6684) is limited by noisy FX data and coarse daily features. Predicting 0.5% intraday moves is inherently challenging, and additional features (macro data, news sentiment) or

higher-frequency data could further improve performance.

Recommendation:
For a conservative trading strategy prioritizing high-confidence trades, CNN + Attention is optimal. For capturing more opportunities while maintaining good overall predictive ability, GRU + Attention provides the best balance of precision and recall.