BackToTheAttention

Stochastic Batman

December 2025

Problem Statement

- Goal: Apply attention mechanisms to time series forecasting, specifically stock price prediction.
- Target: Predict closing prices for Pinterest (PINS) using historical data.
- Why? Attention excels at capturing dependencies in sequences; time series like stocks have temporal patterns that can benefit from this.
- Approach: Code-first implementation in Python (src/), demonstrated via notebooks.
- Data Source: Yahoo Finance API (yfinance) for real-time stock data.
- Note: Results are non-reproducible due to live data fetching; focus on trends and relative performance.

Project Structure and Setup

- **Structure**:
  - src/: Core logic (DeTention.py, train.py, predict.py).
  - notebooks/: Walkthroughs (data_and_training.ipynb, inference.ipynb).
  - models/: Trained weights and plots.
- **Setup**:
  - Python 3.14 virtual environment.
  - Used packages: matplotlib, scikit-learn, statsmodels (for ARIMA), torch, yfinance.
  - Requires internet for data fetching.
- **Usage Example**:

```
python src/train.py --ticker "PINS"
python src/predict.py
```

Data Preparation Steps

- Fetch historical data with Yahoo Finance API.
- Preprocess:
  - Focus on 'Close' prices (target).
  - Normalize using MinMaxScaler to $[0, 1]$ (scaling prevents gradient issues).
  - Create sequences: Window size L (e.g., 30 days) to predict next day.
  - Split: Train (80%), Test (20%).
- Dataset: TensorDataset for PyTorch DataLoader (batch size 32).

Model Architecture: DeTention Overview

- Transformer-inspired architecture for time series forecasting.
- Intuition: Unlike RNNs/LSTMs with sequential processing, attention allows parallel computation and direct focus on relevant past time steps, capturing long-range dependencies in stock prices (e.g., market cycles).
- Components:
  - Input: Sequence of L time steps, each a scalar (closing price) or vector.
  - Embedding: Linear layer to project to d_{model} (e.g., 64).
  - Positional Encoding: RoPE (Rotary Position Embeddings) for relative positioning.
  - Encoder: Stack of n_{layers} (e.g., 2) DeTentionBlocks.
  - Output: Pooling (avg or last) + Linear to predict next value.
- Advantages: Scalable, handles variable-length sequences, focuses on important historical patterns.

DeTention: RoPE Positional Encoding

- Intuition: Time series are ordered; RoPE encodes positions via rotations in embedding space, preserving relative distances (better than absolute sinusoids for sequences).
- Math:
  - For dimension pairs $(2i, 2i+1)$: $\theta_i = base^{-2i/d}$ (base=10000).
  - For position m, vector $x = [x_0, x_1, ..., x_{d-1}]$, rotate:

$$x'_{2i} = x_{2i} \cos(m\theta_i) - x_{2i+1} \sin(m\theta_i)$$

$$x'_{2i+1} = x_{2i} \sin(m\theta_i) + x_{2i+1} \cos(m\theta_i)$$

- Applied to queries and keys in attention for relative positional awareness.
- Why? Enables model to generalize to longer sequences than trained on; intuitive for time shifts in stocks.

DeTentionBlock: Attention and Feed-Forward

- Intuition: Core building block; self-attention lets each time step "attend" to others, weighing importance (e.g., recent volatility influences prediction more). FFN adds non-linearity for complex patterns.
- Structure:
  - LayerNorm before attention and FFN for stable training.
  - Multi-Head Self-Attention: $n_{heads} = 4$, each head computes attention in subspace.
  - Recall attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

    where Q, K, V are linear projections of input + RoPE.
  - Feed-Forward: Two linear layers with ReLU, hidden size 256.
  - Dropout (0.2) for regularization.
  - Residual connections: output = input + sublayer(output).
- Overall: Allows model to learn which past prices matter most for forecasting next close.

Training Process

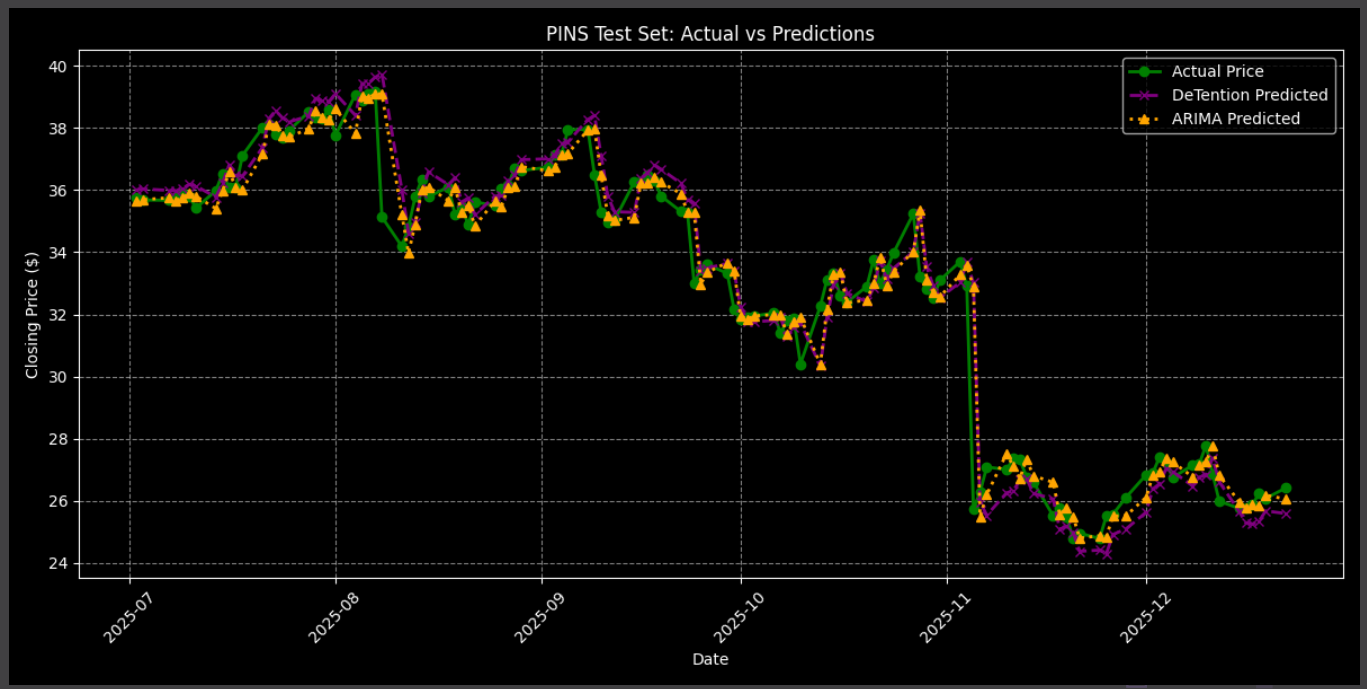
- Hyperparameters: LR=0.0001, Epochs=100, Patience=15 (early stopping).
- Optimizer: Adam; Loss: MSE.
- Process:
  - Train loop with validation.
  - Monitor val loss; save best model.
  - Device: GPU if available.
- Baseline: ARIMA (`p=5, d=1, q=0`) for comparison.
- Plots: Training curves, test predictions vs actuals.

Training Results Interpretation

- Model converges with decreasing loss (train and val).
- Test Performance: Tracks trends well, captures general movements.
- Metrics: Low MSE/MAE indicate reasonable accuracy for volatile stocks.
- Visualization: Prediction plot shows alignment with actual prices, though some deviations in peaks/troughs.
- Insights: Attention helps model long-range dependencies, but stock noise limits perfect prediction.
- Note: Numbers vary with data; focus on relative improvements and trends.

DeTention vs ARIMA

- Comparison on Test Set:
  - ARIMA: Statistical baseline, good for linear autoregressive patterns.
  - DeTention: Handles non-linear dependencies via attention.
- Results: ARIMA often shows slightly lower errors (e.g., 20-25% better in some runs), but DeTention competitive.
- Why ARIMA Wins Sometimes? Stocks can be near-random; simple models suffice.



References

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need.
2. Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., & Liu, Y. (2023). RoFormer: Enhanced Transformer with Rotary Position Embedding.
3. Shazeer, N. (2020). GLU Variants Improve Transformer.