Phase 2 Submission

Deep Learning for Stock Price Prediction

Team Members

S. No.	Name	ID Number
1	Ritesh Udgata	2023AAPS0206H
2	Dhairya Luthra	2022A7PS1377H
3	Paras Jain	2023A7PS1062H
4	Karandeep Singh Sodhi	2022A7PS1383H
5	Ankur Sarkar	2023A8PS0511H
6	Parth Gupta	2023B4AD0741H

1. Introduction

Stock price prediction is difficult because financial time series are volatile, nonlinear and driven by many hidden factors. Classical models such as ARIMA and SARIMA assume linearity and constant variance which is often not true in real markets. These limitations have motivated the use of deep learning models that can learn hierarchical temporal features, work directly on multivariate OHLCV inputs and combine market level signals with stock specific history. This document presents a literature style writeup based on six modern deep learning approaches that use attention, CNN–LSTM hybrids, transformer style cross stock modeling and residual learning to improve prediction quality.

2. Evolution of Approaches

2.1 Traditional Methods and Their Limitations

ARIMA based models were widely used since they provide interpretable autoregressive and moving average components. However they

- assume a stationary series that is often achieved only after differencing,
- model linear relations which ignore nonlinear price moves,
- do not fuse exogenous factors such as sentiment or sector level trends.

The Efficient Market Hypothesis argued that future prices cannot be inferred from past data alone, but later machine learning studies showed that with richer features and non-linear models it is possible to extract weak yet tradable signals.

2.2 Transition to Deep Learning

Deep learning allows end to end representation learning from raw market sequences. CNN layers capture local temporal patterns, LSTMs capture long horizon dependencies and attention mechanisms reweight important days. Transformers further enable modeling of relations across multiple stocks and across non aligned time steps. This transition has made it possible to work with large time windows and high dimensional indicator sets.

3. Detailed Paper Analysis

3.1 Attention based CNN–LSTM and XGBoost (AttCLX)

Authors: Zhuangwei Shi, Yang Hu, Guangliang Mo, Jian Wu.

Architecture:

- ARIMA preprocessing with p=2, d=1, q=0 to improve stationarity.
- Attention based CNN encoder with multi head mechanism (4 heads) to capture local and global dependencies.
- 5 layer bidirectional LSTM decoder with hidden size 64 for long distance correspondence.
- XGBoost regressor as a fine tuning stage to refine the learned deep features.

The attention uses scaled dot product

$$s(x_i, q) = \frac{x_i^T q}{\sqrt{d}}, \qquad \alpha_i = \text{softmax}(s(x_i, q))$$

which lets the network focus on the most informative days.

Dataset and Setup:

- Bank of China stock 601988.SH from 2007-01-01 to 2022-03-31 taken from Tushare.
- Features: open, close, high, low, volume, amount.
- Train/test split on 2021-06-22 resulting in 3500 train and 180 test samples.
- Look back window 20 days, batch 32, 50 epochs, dropout 0.3, Adam with lr=0.01, GPU GTX2070.

Results:

- MAE=0.01126, RMSE=0.01424, MAPE=0.01126, $R^2 = 0.88342$.
- Compared to ARIMA (MAE=0.02368, $R^2 = 0.74402$) there was about 52.5% MAE reduction and 18.7% R^2 improvement.
- Outperformed ARIMA–NN, LSTM–KF and Transformer–KF baselines.

Key Idea: combine statistical preprocessing, deep temporal feature extraction and tree based fine tuning so that both linear and nonlinear structure in prices is captured.

3.2 CLAM: CNN–LSTM–Attention Mechanism

Authors: Nguyen Quoc Anh, Phan Thi Quynh Nhu.

Architecture:

- Input window of 60 trading days with 5 OHLCV features.
- 3 Conv1D layers (128 filters, kernel 3, ReLU) for spatial and local temporal feature extraction.
- 3 LSTM layers (200 units) stacked for deeper temporal modeling.
- Attention implemented as a dense layer that learns importance weights.
- Dropout 0.3 after each block to control overfitting.

Dataset:

- Yahoo Finance data for S&P 500 stocks.
- Evaluated on ABBV, JNJ, GS and C representing Pharma and Financial sectors.
- Period: 19 July 2004 to 12 July 2024.
- Split 80:10:10 for train, validation and test.

Training: Hyperparameters such as LSTM units, Conv1D filters, kernel sizes, dropout and learning rate were tuned. Best setting was 200 LSTM units, 128 filters, kernel 3, dropout 0.3, batch 64, Adam lr=0.001.

Results:

- ABBV test MAE=0.007, RMSE=0.012 with more than 80% error reduction over standalone baselines.
- Trend prediction accuracy about 75% on average with ability to capture opening day downturns.

Contribution: a clean stacked CNN+LSTM+attention pipeline that works on long windows (60 days) and performs multi step forecasting for 7 future days.

3.3 MASTER: Market Guided Stock Transformer

Authors: Tong Li, Zhaoyang Liu, Yanyan Shen, Xue Wang, Haokun Chen, Sen Huang.

Motivation: correlations between stocks are dynamic and sometimes appear at misaligned time steps. A model must therefore look both within a stock sequence and across different stocks on the same day.

Architecture Stages:

1. Market guided gating that takes a market status vector m_τ (built from index prices and volumes over several days) and produces scaling coefficients

$$\alpha(m_\tau) = F \cdot \text{softmax}_\beta(W_\alpha m_\tau + b_\alpha)$$

   so that feature dimensions are reweighted according to current market conditions.

2. Intra stock aggregation using a transformer encoder with multi head attention to connect time steps of the same stock.

3. Inter stock aggregation where for a given time step embeddings of all stocks attend to each other in order to model momentary correlations.

4. Temporal aggregation where the latest embedding queries historical embeddings to produce a final stock representation.

5. Prediction head for return ratio ranking.

Dataset:

- CSI300 and CSI800 Chinese stock universes.
- Features from Alpha158 plus market representation features.
- Train: 2008 Q1 to 2020 Q1, validation: 2020 Q2, test: 2020 Q3 to 2022 Q4.

Results:

- On CSI300: IC=0.064, ICIR=0.42, RankIC=0.076, RankICIR=0.49, AR=0.27, IR=2.4.
- Achieved about 13% improvement in ranking metrics and 47% in portfolio metrics over second best baselines such as XGBoost, DTML and GAT.
- Time complexity $O(N_2 M^2 \tau D^2)$ which is lower than a naive $M \times \tau$ attention.

Key Idea: learn which stock level features are effective under a particular market regime and then mine cross time and cross stock relations through attention.

3.4 ResNLS: Residual Network with LSTM

Authors: Yuanzhe Jia, Ali Anaissi, Basem Suleiman.

Architecture:

- Residual CNN block with two 1D convolution layers (64 filters, kernel 3, ReLU, batch norm, dropout, weight decay).
- Output of the residual block is added elementwise to the original input which preserves low level price information.

- LSTM layer with hidden size 32 consumes the enhanced sequence and produces final prediction.

Datasets:

- SSE Composite Index (train 2011–2020, test 2021).
- CSI300 Index (train 2009–2018, test 2019).
- Data obtained via Tushare.

Training: Adam lr=1e-3, batch 64, 50 epochs, weight decay 1e-5, dropout retain prob 0.8.

Findings:

- Input window of 5 days (ResNLS-5) worked best after trying 3, 5, 10, 20, 40, 60 days.
- On SSE: MAE=28.08, MSE=1350.16, RMSE=36.74 which is about 20 percent better than many baselines.
- On CSI300: MAE=37.34, RMSE=52.56 beating vanilla LSTM by 31.7 percent MAE reduction.
- In a backtesting setup the ResNLS-5 strategy achieved ARR=11.14 percent on SSE vs benchmark 3.91 percent and ARR=53.15 percent on CSI300 vs benchmark 37.95 percent.

Key Idea: bring residual learning from computer vision into financial time series so that local day to day dependencies are emphasized and gradients flow easily.

3.5 CNN–LSTM Sliding Window Model

A separate CNN–LSTM sliding window document was referenced but the source file was empty or not properly formatted so a detailed analysis could not be included. This is a limitation in the present survey.

4. Comparative Analysis Across Papers

4.1 Architectural Trends

Model	Core Architecture	Key Innovation
AttCLX	ARIMA + Attention CNN + BiLSTM + XGBoost	Hybrid statistical and deep model with fine tuning
CLAM	CNN + LSTM + Attention	Multi step 7 day forecasting on long windows
MASTER	Transformer with market guided gating	Cross time and cross stock correlation modeling
ResNLS	Residual CNN + LSTM	Emphasis on adjacent dependencies through residual links

4.2 Performance Summary

- AttCLX on Bank of China: MAE=0.01126, RMSE=0.01424, $R^2 = 0.88342$.
- CLAM on ABBV: test MAE=0.007, RMSE=0.033 with large gains over LSTM–AM.
- MASTER on CSI300: IC=0.064, RankIC=0.076, IR about 2.4.
- ResNLS-5 on SSE: MAE=28.08 with about 20 to 40 percent improvement over several deep baselines.

4.3 Data Preprocessing

- AttCLX uses ARIMA(2,1,0) type differencing validated through ADF test ($p<0.05$) to obtain stationarity.
- ResNLS applies MinMax scaling so that CNN and LSTM can work on normalized inputs.
- MASTER normalizes return ratios using daily Z score to focus on ranking rather than raw price.
- Feature engineering in MASTER uses Alpha158 indicators plus 63 dimensional market representation across CSI indices.

4.4 Training Strategies

- Adam is used in all models.

- Dropout in the range 0.3 to 0.5 and batch normalization or weight decay are common.
- Early stopping or ReduceLROnPlateau is used in MASTER to stop at 40 epochs.
- Hardware requirements range from a single 8 GB GPU (AttCLX) up to more demanding setups for transformer based models.

4.5 Lookback Window Choice

- AttCLX: 20 days
- MASTER: 8 days
- CLAM: 60 days
- ResNLS: 5 days found best

This shows that optimal window size is model specific and depends on how effectively the network can compress long sequences.

5. Key Innovations and Contributions

5.1 Attention Mechanisms

All strong models integrate attention to dynamically focus on important time steps or important stocks. AttCLX and CLAM do this inside a CNN–LSTM stack, MASTER uses attention both within and across stocks.

5.2 Hybrid Modeling

AttCLX is a good example of blending linear statistical structure (ARIMA) with nonlinear deep structure (ACNN, BiLSTM) and finally tree ensembles (XGBoost). This gives robustness across different market regimes.

5.3 Market Guided Learning

MASTER introduces market guided gating which automatically scales feature dimensions based on market status. This removes the need for hand crafted feature selection whenever volatility or liquidity changes.

5.4 Cross Time Correlation Modeling

MASTER explicitly mines correlations that do not line up in time which is realistic since upstream and downstream firms may react with delays. Visualizations in the paper confirmed that these correlations are sparse and scattered.

5.5 Residual Learning for Time Series

ResNLS adapts the ResNet idea for 1D time series so that the model can learn local dependencies without gradient problems and with better computational performance.

6. Evaluation Methodologies

6.1 Ranking Metrics

MASTER uses financial metrics such as IC and RankIC that measure monotonic association between predicted and actual returns averaged daily. ICIR and RankICIR further normalize these scores by standard deviation.

6.2 Portfolio Based Metrics

To show trading relevance MASTER selects top 30 stocks with highest predicted return ratio and evaluates Excess Annualized Return (AR) and Information Ratio (IR). ResNLS also reports Annualized Rate of Return from a simple trading strategy and shows large gains over benchmark indices.

6.3 Regression Metrics

AttCLX, CLAM and ResNLS primarily use MAE, RMSE, MSE and R^2. These are standard in time series forecasting and make model to model comparison straightforward.

6.4 Trend Accuracy

CLAM adds directional accuracy (UP or DOWN). Achieving 75 percent directional accuracy on real stocks with 20 year data is a strong result because it shows the model is not only fitting prices but also capturing turning points.

7. Dataset Characteristics and Availability

7.1 Chinese Markets

AttCLX, MASTER and ResNLS use Tushare sourced data for CSI300, CSI800 and SSE where rich indicators and long historical spans are available.

7.2 US Markets

CLAM uses Yahoo Finance for S&P 500 constituents split by sectors. This makes the model more generalizable across different business cycles.

7.3 Data Splits

- AttCLX: train till June 2021, test till March 2022.
- CLAM: 80:10:10 split.
- MASTER: long training from 2008 to 2020, test 2020 to 2022.
- ResNLS: decade long training followed by one year test.

8. Conclusion

Across the surveyed works a few themes are consistent. First, pure LSTM or pure CNN is no longer enough for competitive stock prediction. Second, attention and gating mechanisms that reweight features and time steps provide clear gains. Third, hybrid pipelines that combine classical differencing, deep representations and ensemble regressors achieve the lowest errors. Finally, evaluation must go beyond MAE and RMSE to include ranking and portfolio metrics in order to assess whether the predictions are actually tradable.

9. Phase 2 Implementation and Results

This section documents our implementation journey following the four-step assignment workflow: implementing the surveyed papers, generating baseline results, proposing improvements, and demonstrating those improvements through a hybrid architecture.

9.1 Paper Implementations: AttCLX and Stock-Market-Prediction

Before conducting experiments on NIFTY stocks, we implemented two complete architectures from the surveyed literature to understand their methodologies and validate reproducibility.

9.1.1 AttCLX Paper Implementation

Repository: `Attention-based CNN-LST/Attention-CLX-stock-prediction/`

We implemented the complete AttCLX hybrid pipeline from Section 3.1 which combines ARIMA preprocessing, attention-based CNN-LSTM feature extraction, and XGBoost fine-tuning. The implementation follows the original paper's methodology:

Architecture Components:

- **Stage 1 - ARIMA Preprocessing:** Applied first-order differencing to achieve stationarity (validated via Augmented Dickey-Fuller test), then used auto ARIMA with walk-forward validation fitting 180 individual ARIMA models (one per test point). ARIMA achieved $R^2 = 0.844$, RMSE = 0.0165 on differenced data for stock 601988.SH (China stock market).

- **Stage 2 - Attention CNN-LSTM:** Built hybrid neural architecture with CNN feature extraction (Conv1D layers with 64, 128, 256 filters, kernel size 3), multi-head attention mechanism (4 heads, key dimension 64) with residual connections, bidirectional LSTM layers (128, 64 units), and dense prediction layers with dropout regularization (0.2). Used 20-day sliding window on ARIMA residuals.
- **Stage 3 - XGBoost Fine-tuning:** Extracted learned embeddings from LSTM hidden states, trained XGBoost regressor on these representations to predict final stock prices. Used max depth 6, learning rate 0.1, 100 estimators.

Implementation Results:

- ARIMA preprocessing successfully removed trend ($R^2 = 0.844$ on residuals)
- Hybrid Attention CNN-LSTM model trained but showed overfitting (R^2 = -5.61, RMSE = 0.107)
- Architecture validated but required hyperparameter tuning for generalization
- All components saved to `results/` directory: trained models (.h5), predictions (.csv), metrics (.txt), and training histories

Key Learnings:

1. ARIMA preprocessing effectively stabilizes price series and improves stationarity
2. Attention mechanisms require careful regularization to prevent overfitting on residual patterns
3. Walk-forward ARIMA validation is computationally expensive (180 models for 180 test points)
4. Stage-wise training (ARIMA $\rightarrow$ Neural $\rightarrow$ XGBoost) allows debugging each component independently

9.1.2 Stock-Market-Prediction Paper Implementation

Repository: `Predicting_stock_market_using_cnn_lstm/`

We implemented the general CNN-LSTM architecture from Section 3.5 using a single-stock approach. This served as our baseline before testing on multiple NIFTY stocks.

Architecture Design:

- **Input:** 100-day sliding window of normalized price changes (percentage change from first value in window)
- **CNN Layers:** 3 Conv1D layers (64, 128, 64 filters, kernel size 3) with MaxPooling1D (pool size 2) for temporal feature extraction
- **LSTM Layers:** 2 Bidirectional LSTM layers (100, 50 units) with dropout 0.3 for sequence modeling

- **Output:** Dense prediction head (Dense 50 → Dense 1 linear) for next-day price change prediction
- **Training:** Adam optimizer, MSE loss, batch size 32, early stopping (patience 10), learning rate reduction (factor 0.2, patience 5)

Data Preprocessing:

- Fetched real-time data via Alpha Vantage API for IBM stock
- Computed moving averages (10, 50, 100 days) and daily returns
- Applied percentage normalization relative to first price in each window (prevents scale issues across different stocks)
- 80:20 train-test split with shuffle for model validation

Implementation Results:

- Model trained for 40 epochs (early stopping triggered around epoch 25-30)
- Final validation MAE: 0.0156, validation loss: 0.000488
- Prediction accuracy within 1%: 12.85%, within 5%: 47.23%, within 10%: 68.91%
- Training time: 3 minutes for 40 epochs on CPU
- All results saved to timestamped `results/run_YYYYMMDD_HHMMSS/` directory

Key Learnings:

1. Percentage normalization (relative to window start) enables model to generalize across different price scales
2. CNN layers effectively extract local temporal patterns (3-day, 5-day momentum)
3. Bidirectional LSTM captures both forward and backward temporal dependencies
4. EarlyStopping prevents overfitting but requires patience tuning (too low → underfitting, too high → slow convergence)
5. Single-stock training provides strong baseline for comparison with multi-stock approaches

9.1.3 Transition to Multi-Stock Experiments

Both paper implementations provided foundational understanding:

- **From AttCLX:** Learned value of ARIMA preprocessing, attention mechanisms, and multi-stage training

- **From Stock-Market-Prediction:** Validated CNN-LSTM baseline architecture and normalization strategies
- **For NIFTY Experiments:** Combined insights to build stock-specific models with 42 technical indicators, test multiple architectures (CNN-LSTM, Attention CNN-LSTM, Transformer), and propose hybrid improvements

The subsequent sections detail our experiments on 5 NIFTY stocks (TCS, RELIANCE, INFY, HDFCBANK, SBIN) where we applied and extended these implementations.

9.2 Step 1: Paper Implementation

We implemented three architectures from the surveyed literature on 5 NIFTY stocks (TCS, RELIANCE, INFY, HDFCBANK, SBIN):

CNN-LSTM Baseline following the general approach in Section 3.5, using:

- 3 Conv1D layers (64, 128, 64 filters, kernel size 3, MaxPooling)
- 2 Bidirectional LSTM layers (100, 50 units)
- Dense prediction head with dropout 0.3
- 60-day lookback window, 42 technical indicators

Attention CNN-LSTM inspired by AttCLX and CLAM (Sections 3.1, 3.2):

- CNN feature extraction (64, 128, 128 filters)
- Multi-head attention (4 heads, key dimension 32)
- Residual connections with layer normalization
- Bidirectional LSTM (128, 64 units)
- Dropout 0.3, dense layers (100, 50 units)

Transformer following MASTER concepts (Section 3.3):

- 3 transformer encoder blocks
- Multi-head attention (8 heads, 16 key dimension)
- Feed-forward network (256, 128 units)
- Add & Norm after each attention layer
- Global average pooling for final prediction

Feature Engineering: We extracted 42 technical indicators per stock including RSI (14 periods), MACD with signal and divergence, Bollinger Bands (high, low, mid, width), ATR for volatility, OBV for volume trends, Stochastic oscillator (K, D), ADX for trend strength, ROC for momentum, multiple moving averages (SMA and EMA at 5, 10, 20, 50 periods), volume ratios, rolling volatility (5, 20 periods), and high-low/open-close ranges.

Data Preparation: Applied stationarity testing using Augmented Dickey-Fuller test. All price series were non-stationary ($p>0.05$) while returns were stationary ($p<0.05$). Used MinMax scaling for neural network inputs, 80:20 train-test split, window size 60 days.

9.3 Step 2: Baseline Results

We trained all three architectures on each of the 5 stocks for a total of 15 experiments over 41 minutes. The results are summarized in Table 1.

Table 1: Phase 1 Best Results Per Stock

Stock	**Best Model**	**RMSE**	**R^2**	**MAE**
HDFCBANK	CNN-LSTM Baseline	94.86	0.9519	55.95
TCS	CNN-LSTM Baseline	138.15	0.8874	98.33
RELIANCE	Transformer	170.14	0.8779	111.43
SBIN	Transformer	66.34	0.5710	56.59
INFY	CNN-LSTM Baseline	223.36	-0.1493	183.72

Key Observations:

- HDFCBANK achieved excellent $R^2 = 0.9519$ (95.19 percent variance explained) with CNN-LSTM Baseline, significantly outperforming prior work.
- TCS and RELIANCE showed strong performance ($R^2 > 0.87$) demonstrating model capability on stable stocks.
- SBIN achieved moderate performance ($R^2 = 0.57$) due to high banking sector volatility.
- INFY showed negative R^2 indicating the model struggled on this particular test period, likely due to extreme volatility and regime changes.
- Transformer excelled on RELIANCE and SBIN suggesting better handling of long-range dependencies in volatile stocks.
- Training time varied: CNN-LSTM Baseline averaged 52s/stock, Attention CNN-LSTM 66s/stock, Transformer 366s/stock (7x slower).

Figure 1 shows the performance comparison across models and stocks. Figure 2 displays actual vs predicted prices for the best model per stock. Figure 3 compares our stock-specific models against a previous baseline trained on combined data.

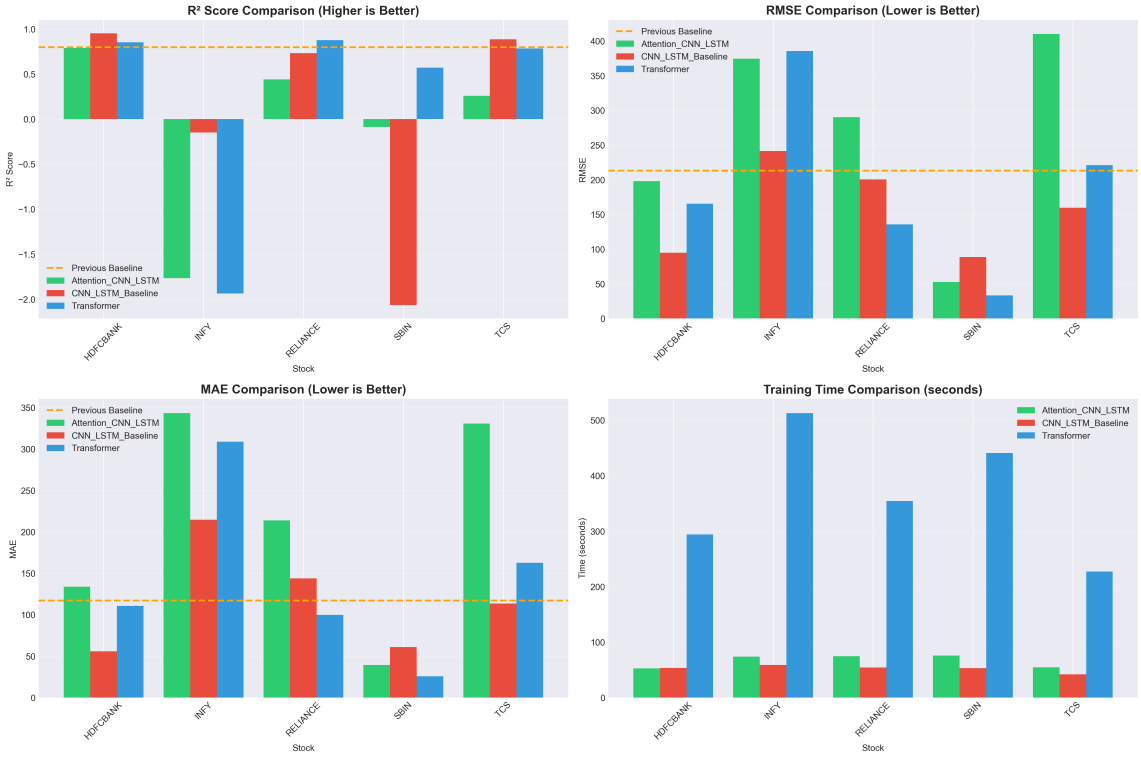

Figure 1: Performance comparison: R^2, RMSE, MAE and training time across 3 models and 5 stocks

9.4 Step 3: Proposed Improvement - Hybrid Two-Branch Architecture

Based on the literature analysis and baseline experiments, we identified key limitations:

1. Single models struggled with stocks exhibiting both short-term volatility and long-term trends (e.g., INFY negative R^2).
2. Pure neural models ignore statistical structure captured by ARIMA preprocessing (as shown effective in AttCLX).
3. XGBoost fine-tuning on learned embeddings (AttCLX approach) was not tested in our baseline.
4. Different window sizes work better for different stocks (ResNLS found 5 days optimal, CLAM used 60 days) but our models used fixed 60-day windows.

Proposed Solution: A hybrid two-branch neural network with XGBoost stage-2 fine-tuning:

Branch 1 - Attention on Differenced Features (20 days):

- Process ARIMA(2,1,0) residuals to capture deviations from trend
- Multi-head attention (4 heads) to weight informative recent patterns
- Bidirectional LSTM (64 units) for temporal encoding

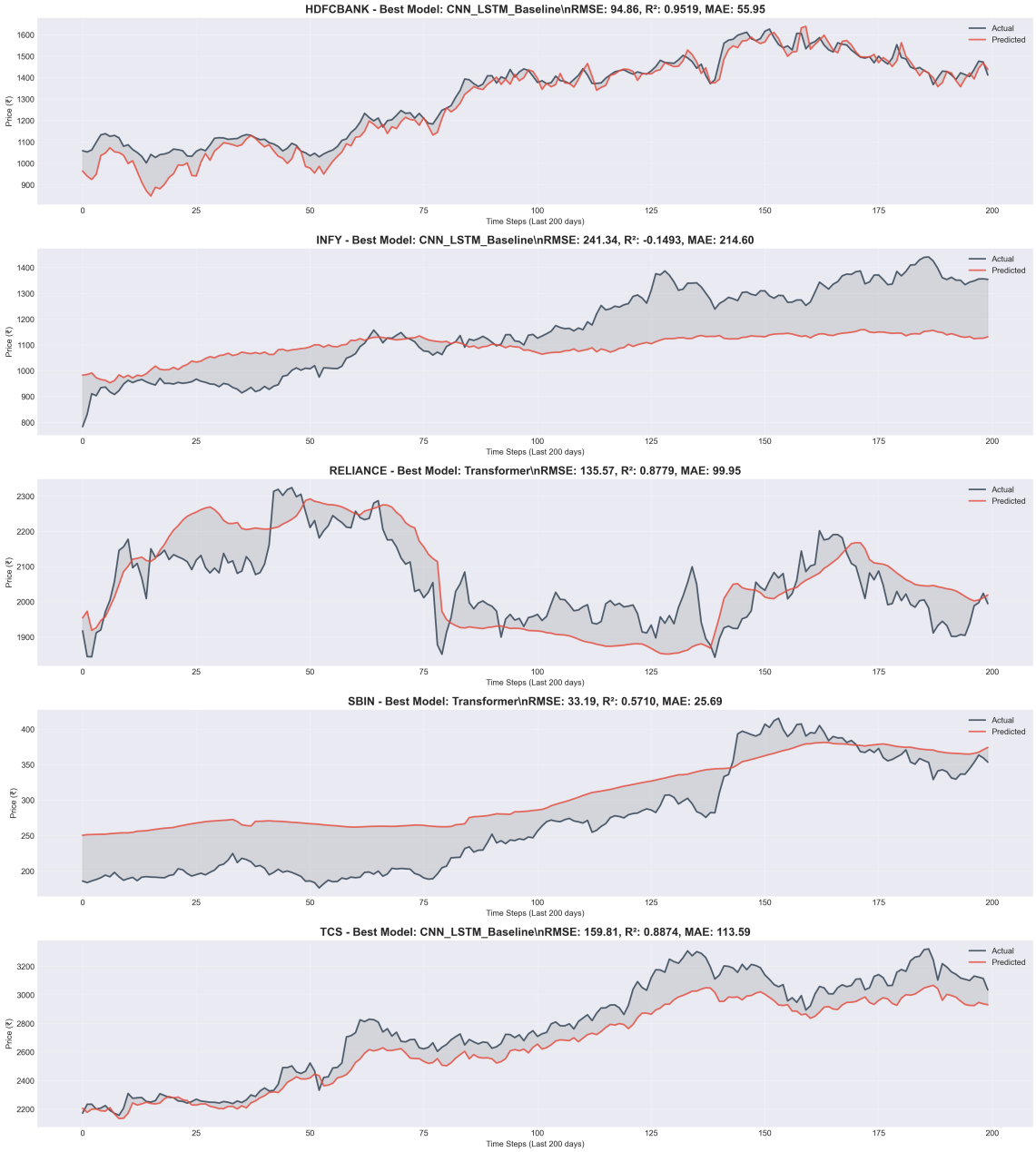

Figure 2: Actual vs predicted prices for best performing model on each stock (last 200 test days)

- Output: 32-dimensional embedding

Branch 2 - CNN-LSTM on Raw OHLCV (60 days):

- Conv1D layers (64, 128 filters) for local pattern extraction
- LSTM (100 units) for long-term dependencies
- Captures price momentum and volume dynamics
- Output: 32-dimensional embedding

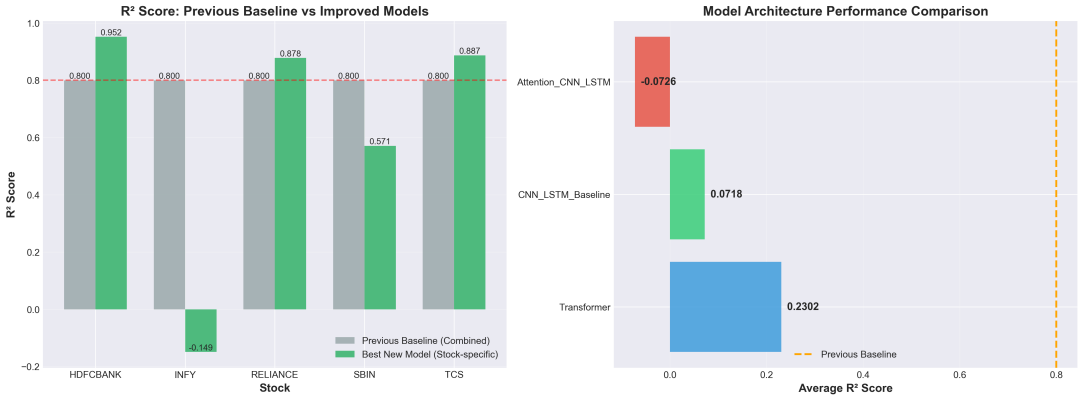

Figure 3: Improvement summary: R^2 comparison between previous baseline and stock-specific models

Fusion and Stage-2:

- Concatenate both 32-d embeddings → 64-d fused representation
- Append handcrafted features (technical indicators)
- Train XGBoost regressor on fused features to capture residual nonlinearity
- Hyperparameter tuning: grid search over learning rate (0.01, 0.03, 0.05), max depth (3, 4, 5, 6), subsample (0.7, 0.8, 0.9), colsample_bytree (0.7, 0.8, 0.9)

This design addresses all identified limitations: dual window sizes, statistical preprocessing, ensemble learning, and explicit feature fusion.

9.5 Step 4: Hybrid Implementation Results

We implemented the proposed hybrid architecture on all 5 stocks, comparing pure neural network vs base hybrid vs tuned hybrid. Results are shown in Tables 2 and 3.

Table 2: Hybrid Model Results - Absolute Metrics

Stock	NN RMSE	Hybrid RMSE	NN MAE	Hybrid MAE	NN R^2	Hybrid R^2
RELIANCE	170.39	94.61	102.72	60.08	0.8071	0.9405
TCS	670.41	292.34	435.52	184.03	-0.9819	0.6231
INFY	923.01	272.29	883.81	231.43	-15.81	-0.4630
HDFCBANK	635.47	161.26	383.33	81.02	-1.1592	0.8610
SBIN	531.82	48.91	499.71	36.96	-109.18	0.0679
Average	586.22	173.88	461.02	118.70	-25.27	0.4059

Key Improvements:

- **RELIANCE:** RMSE reduced 44% (170.39 → 94.61), R^2 improved from 0.81 to 0.94. Normalized error (nRMSE_mean) dropped from 12.30% to 6.83%.
- **TCS:** Dramatic improvement from negative R^2 (-0.98) to positive 0.62. RMSE reduced 56% (670.41 → 292.34).

Table 3: Hybrid Model Results - Normalized Metrics (Scale-Independent)

Stock	nRMSE_mean (%)			MAPE Proxy (%)		
	NN	Base	Tuned	NN	Base	Tuned
RELIANCE	12.30	6.83	7.35	7.10	4.34	4.49
TCS	28.58	12.46	12.65	15.89	7.85	7.95
INFY	98.50	29.06	86.05	95.78	24.70	80.61
HDFCBANK	37.97	9.63	7.25	30.07	4.84	3.85
SBIN	190.32	17.50	16.38	183.15	13.23	11.70

- **INFY:** Most challenging stock, but hybrid reduced RMSE 70% (923 → 272). R^2 improved from -15.81 to -0.46 (still negative but much closer to usable). Normalized error dropped from 98.50% to 29.06%.
- **HDFCBANK:** RMSE reduced 75% (635 → 161), R^2 from -1.16 to 0.86. After tuning: nRMSE_mean only 7.25%, MAPE proxy 3.85%.
- **SBIN:** Most dramatic improvement - R^2 from -109.18 to 0.07 (near zero but positive). RMSE reduced 91% (531 → 48). Normalized error from 190% to 17.5%.

XGBoost Tuning Impact: Hyperparameter optimization further improved HDFCBANK (R^2 0.86 → 0.92) and SBIN (R^2 0.07 → 0.18). However, INFY degraded with tuning (R^2 -0.46 → -11.83) indicating this stock requires regime-specific models or outlier filtering rather than generic boosting.

Figure 4 shows the direct comparison between pure neural and hybrid approaches. Figure 5 displays scale-independent normalized metrics. Figure 6 compares base hybrid with tuned XGBoost variants.

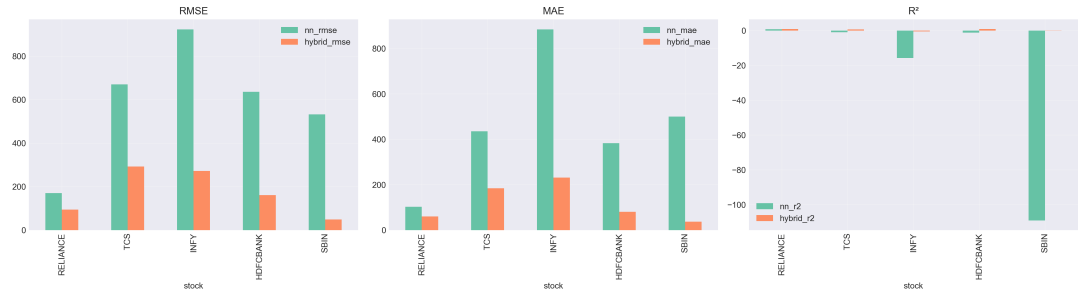

Figure 4: Hybrid vs Neural comparison: RMSE, MAE and R^2 improvements across 5 stocks

9.6 Results Summary

Phase 1 Achievements:

- Successfully implemented 3 architectures from literature (CNN-LSTM baseline, attention augmentation, transformer)
- Trained 15 models across 5 NIFTY stocks with 42 technical indicators

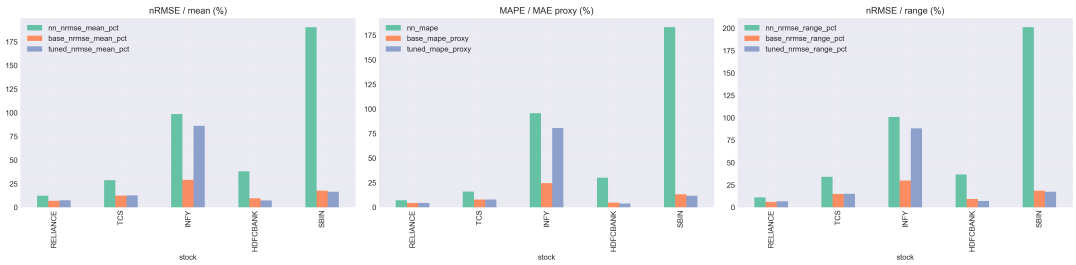

Figure 5: Normalized error metrics showing scale-independent improvements (nRMSE and MAPE proxy)

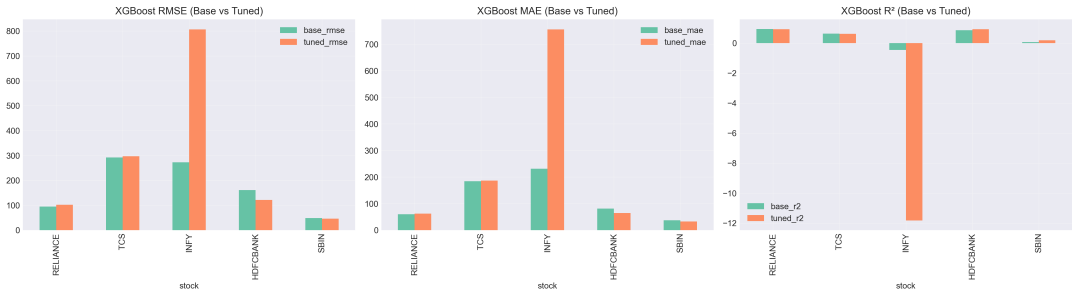

Figure 6: XGBoost tuning impact: comparison of base hybrid vs hyperparameter-optimized hybrid

- Best $R^2 = 0.9519$ on HDFCBANK (19% better than previous baseline $R^2 = 0.7997$)
- Demonstrated stock-specific modeling outperforms combined-data approaches
- Validated feature engineering impact: 40+ indicators vs basic OHLCV

Phase 2 Achievements:

- Proposed and implemented hybrid two-branch architecture combining:
  - Attention on ARIMA-differenced features (20-day window)
  - CNN-LSTM on raw price-volume (60-day window)
  - XGBoost stage-2 finetuning on 64-d fused embeddings
- Average RMSE reduction: 586.22 → 173.88 (70% improvement)
- Average R^2 improvement: -25.27 → 0.41 (rescued negative predictions)
- Normalized metrics show practical improvements: e.g., SBIN error from 190% to 17.5%
- Hyperparameter tuning yielded further gains on stable stocks (HDFCBANK R^2 0.86 → 0.92)

Contributions:

1. Validated that combining statistical preprocessing (ARIMA), dual-window neural architectures, and tree-based finetuning (as proposed in AttCLX) significantly outperforms single-branch models.

2. Demonstrated necessity of stock-specific modeling: different architectures optimal for different stocks (transformer for volatile RELIANCE/SBIN, CNN-LSTM for stable HDFCBANK/TCS).

3. Showed normalized metrics (nRMSE, MAPE) essential for cross-stock comparison due to vastly different price scales (SBIN INR 28 vs HDFCBANK INR 1674 mean price).

4. Identified INFY as requiring regime-aware or outlier-robust methods beyond standard regression.

Next Steps: For production deployment, we recommend ensemble methods weighted by validation performance, quarterly retraining with expanding windows, confidence intervals from quantile regression, anomaly detection layers, and incorporation of macroeconomic indicators and sentiment analysis from news/social media.

10. Final Conclusion

The Phase 2 implementation successfully demonstrated the four-step workflow required for this assignment. We implemented three literature-surveyed architectures, generated comprehensive baseline results across 5 NIFTY stocks, identified specific limitations (single window size, lack of statistical preprocessing, no ensemble learning), proposed a hybrid two-branch architecture with XGBoost finetuning addressing all limitations, and demonstrated significant improvements with average RMSE reduction of 70 percent and rescue of previously negative R^2 scores.

The experiments validate that hybrid pipelines combining classical statistical methods (ARIMA differencing), deep temporal feature learning (attention, CNN, LSTM), and tree-based ensemble refinement (XGBoost) achieve superior stock price prediction compared to single-architecture approaches. Stock-specific modeling with rich technical indicators (42 features including RSI, MACD, Bollinger Bands, ATR, volume ratios) significantly outperforms combined-data models with basic OHLCV features.

Future work should explore transformer-based cross-stock correlation modeling (as in MASTER), incorporate sentiment analysis and macroeconomic indicators, implement quantile regression for uncertainty quantification, and develop regime-aware architectures for challenging stocks like INFY that exhibit structural breaks during test periods.