Wavelet-Based Short-Horizon Forecasting of BTCUSDT Returns

Jashua Luna

March 28, 2026

Abstract

This report presents a MATLAB research prototype for short-horizon BTCUSDT return forecasting. The pipeline loads and cleans 1-minute OHLCV data, computes log returns, creates chronological train/validation/test splits, builds overlapping lookback and forecast windows, constructs raw and wavelet-based feature variants, trains baseline and residual TCN models, and evaluates all forecasts in the original return domain. The central question is whether a multiscale wavelet representation improves short-horizon forecasts relative to raw-return inputs under a matched experimental design. For the single reported run, recorded run metadata indicate that zero and mean baselines achieve the best MAE and RMSE, ridge achieves the strongest directional accuracy and Pearson correlation, and none of the residual TCN variants surpass these simpler baselines. Among the neural models, the raw TCN is the strongest, while the wavelet-only and dual-branch raw+wavelet models remain less accurate and less reliably calibrated. These conclusions should be read as specific to this dataset, feature set, and target design rather than as a general verdict on wavelet methods in financial forecasting.

1 Introduction

One-minute financial time series are noisy, nonstationary, and sensitive to regime changes [2]. In that setting, the question of interest is not whether BTCUSDT is broadly predictable, but whether a multiscale preprocessing step can improve short-horizon return forecasts under a controlled comparison.

This report documents a config-driven MATLAB R2021b pipeline built around fixed chronological splits, train-only normalization, matched evaluation metrics, and reproducible run artifacts. The study compares three input modes—normalized raw return windows, normalized wavelet-derived features built from the same windows, and a combined raw + wavelet representation—under a shared forecasting and evaluation setup. The emphasis is therefore on a narrow, internally consistent model comparison rather than on broad claims about market predictability.

2 Problem Definition

Let c_t denote the BTCUSDT close price at minute t. The primary supervised series is the one-minute log return

$$r_t = \log(c_t) - \log(c_{t-1}).$$

For a lookback length $L = 256$, the input at anchor time t is

$$\mathbf{x}_t = [r_{t-L+1}, r_{t-L+2}, \ldots, r_t].$$

For forecast horizon $H \in \{30, 60, 128\}$, the implemented target is direct multi-step forecasting of future one-minute returns:

$$\mathbf{y}_t = [r_{t+1}, r_{t+2}, \ldots, r_{t+H}].$$

This is the target used throughout the pipeline. The direct multi-step framing follows standard multi-horizon forecasting practice [3]. The preprocessing stage also computes aggregate H-step log returns such as r_{30}, r_{60}, and r_{128} for metadata and diagnostic use, but model training and evaluation are carried out on the direct multi-step return sequence above.

Leakage control is handled in two steps: the chronological train/validation/test split is created before window construction, and windows are then generated separately within each split so that no sample crosses a split boundary, consistent with standard time-series evaluation practice [3, 4].

3 Data

3.1 Source

The repository expects a CSV file with the columns `timestamp`, `open`, `high`, `low`, `close`, and `volume`. This schema matches Binance kline market-data conventions, where candlestick bars are indexed by open time [1]. The checked-in local dataset is `root/project_root/data_raw/btcusdt_1m.csv`. The project also includes a helper function, `download_binance_klines_csv`, for downloading Binance 1-minute klines directly into the configured raw-data path.

For the current local repo state, the available BTCUSDT CSV spans 2021-01-01 00:00:00 UTC through 2021-02-01 00:00:00 UTC and contains 44,641 one-minute rows.

3.2 Cleaning and integrity checks

The loading stage standardizes column names, parses timestamps with several fallbacks, converts the data into a MATLAB timetable, sorts rows chronologically, removes duplicate timestamps, and reports missing-minute gaps. Timestamp handling is normalized to UTC in the default configuration.

For the current local dataset, the saved cleaning report is summarized in Table 1. The checks indicate that the sample remained unchanged after cleaning, with no duplicate timestamps and no detected missing-minute gap segments.

Check	Result
Original rows	44,641
Cleaned rows	44,641
Duplicate timestamps removed	0
Missing-minute gap segments	0

Table 1: Cleaning and integrity summary for the local BTCUSDT sample used in the reported run.

3.3 Chosen variable

The predictive target is built from close-to-close log returns. Although the cleaned timetable preserves OHLCV fields, the current modeling pipeline uses only lagged one-minute return histories as model input. Auxiliary covariates such as volume, range, or realized volatility are natural extensions, but they are not yet part of the implemented feature set and therefore are outside the scope of the current comparison.

3.4 Chronological split

The default experiment uses proportion-based chronological splitting with

$$[0.70,\ 0.15,\ 0.15]$$

for train, validation, and test respectively, and a split buffer of 0 minutes. For the current local dataset, the split report is summarized in Table 2. No random shuffling is used before splitting [3,4].

Split	Rows	Start	End
Train	31,249	2021-01-01 00:00	2021-01-22 16:48 UTC
Validation	6,696	2021-01-22 16:49	2021-01-27 08:24 UTC
Test	6,696	2021-01-27 08:25	2021-02-01 00:00 UTC

Table 2: Chronological split ranges used in the reported experiment.

4 Representation

4.1 Raw representation

The raw-input representation is a length-256 vector of one-minute returns. A column-wise z-score normalizer is fit on the training split only and then reused for the validation and test splits.

4.2 Wavelet representation

Wavelet features are constructed one lookback window at a time. The feature builder follows a staged backend preference: use a dual-tree complex wavelet transform via `dtwavexfm` when available on the MATLAB path, otherwise fall back to a standard discrete wavelet decomposition via `wavedec`, and finally fall back to the identity window itself if neither backend is available. The DTCWT is used here for its approximate shift-invariance and complex coefficient representation, while the DWT provides the simpler multilevel decomposition fallback [6–8].

Under the default configuration, the wavelet settings are summarized in Table 3.

Setting	Value
Preferred method	`dtcwt`
Fallback method	`dwt`
Wavelet family	`db4`
Decomposition levels	4
Boundary handling	`per`
Output mode	flattened feature vector

Table 3: Default wavelet configuration used in the reported run.

If the dual-tree backend is available, the feature vector concatenates real and imaginary coefficient components. In all cases, the code records a feature schema so that downstream dimensions remain fixed and reproducible for a given run.

4.3 Input variants

The implemented experiment supports three feature modes: raw returns only, wavelet features only, and raw + wavelet combined.

For the combined mode, the pipeline concatenates normalized raw and normalized wavelet features and then fits an additional column-wise normalizer on the training split before neural training.

5 Models

The baseline set combines simple benchmark forecasts with linear and neural sequence models. Standard benchmark methods remain essential for contextualizing forecast skill, while ridge regression and residual TCNs provide linear and nonlinear comparators [3, 9, 10].

Model	Description
Zero baseline	Predicts zero for every future return step across the horizon.
Last baseline	Repeats the last observed return in the input window across the entire horizon.
Mean baseline	Predicts the mean training return repeated across all forecast steps.
Ridge baseline	Multivariate ridge regression trained on normalized raw return windows (intercept added, ridge penalty $\lambda = 1.0$).
Raw TCN	Residual temporal convolutional network on raw return windows (see Table 5 for default architecture).
Wavelet TCN	Same residual TCN backbone trained on normalized wavelet-derived features.
Dual-Branch TCN	Dual-branch raw + wavelet encoders with late fusion before the forecast head.

Table 4: Summary of model families evaluated in the experiment.

Component	Configuration
Input	Image-style input layer with shape [input_dim, 1, 1] and no built-in normalization
Backbone	Stem convolution plus four residual temporal blocks
Dilations	$[1, 2, 4, 8]$ with kernel width 3
Channels	$[24, 32]$
Pooling	Global average pooling over the temporal axis
Head	Fully connected forecast head of size $\max(2H, 64)$
Output	Fully connected layer of size H plus regression output

Table 5: Default raw TCN architecture. The wavelet TCN uses the same backbone, while the combined model duplicates the branch encoder and fuses the embeddings before the forecast head.

6 Training Setup

6.1 Normalization

All normalization statistics are fit on the training split only. The default normalization method is z-score scaling with a small numerical safeguard $\epsilon = 10^{-8}$ to avoid division by very small standard deviations [4, 5].

6.2 Loss

The current default implementation uses mean squared error through a standard regression output layer [12]. A more elaborate mixed Huber objective with aggregate and directional penalties was prototyped in this repository, but it is not the default report configuration because it produced

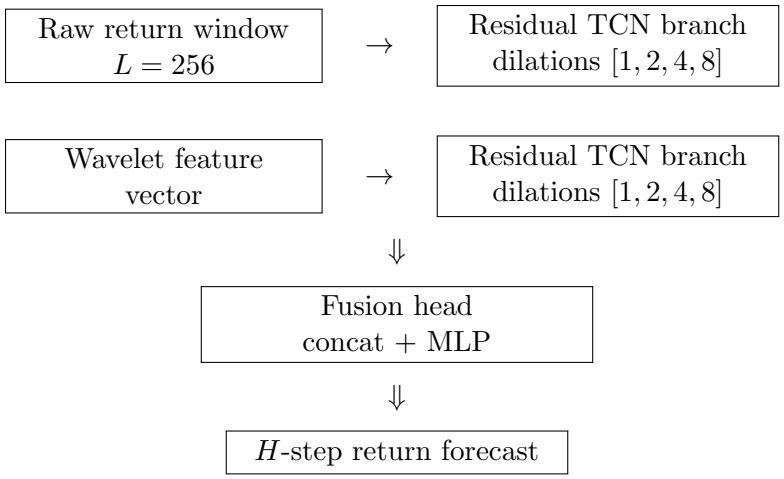

Figure 1: Residual dual-branch TCN used for the combined raw + wavelet model. Each branch learns a separate temporal embedding before late fusion. The current default neural objective is plain mean squared error for stable magnitude calibration.

unstable forecast magnitudes in practice. The simpler MSE objective is therefore the safer baseline for the current residual TCN comparison.

6.3 Optimization

Neural models are trained with Adam using MATLAB's `trainingOptions` [11,13]. Default settings are summarized in Table 6.

Setting	Value
Initial learning rate	10^{-3}
Batch size	128
Maximum epochs	12
Validation patience	5
Execution environment	`auto`
Shuffle	`never`

Table 6: Default neural optimization settings.

The no-shuffle setting is not meant to claim any theoretical advantage; it simply matches the current implementation and keeps runs deterministic given a fixed seed and environment.

6.4 Model selection and repeated runs

The default repeated neural seeds are

$$\{42,\ 43,\ 44\}.$$

Metrics are saved per seed and then aggregated at the reporting stage. The repository also includes an optional validation-based tuning harness for the raw, wavelet, and combined neural models. The default tuning search covers learning rate, filter configuration, dropout, and epoch budget on a fixed wavelet representation.

7 Evaluation

7.1 Primary metrics

All scoring is performed in the original return domain. The evaluator records the metrics in Table 7.

Metric	Interpretation
MAE	Mean absolute error over the direct multi-step return path
RMSE	Root mean squared error over the direct multi-step return path
Stepwise directional accuracy	Fraction of forecast steps with correct sign
Cumulative directional accuracy	Sign accuracy of the full-horizon aggregate return
Pearson correlation	Linear association between predicted and realized returns

Table 7: Primary evaluation metrics.

7.2 Aggregation protocol

Metrics are recorded separately for train, validation, and test splits. The final summary stage aggregates test metrics by model and horizon, reporting mean and standard deviation across repeated neural seeds when applicable.

7.3 Ablation studies

The current codebase directly supports ablations over input mode (raw, wavelet, or raw + wavelet), forecast horizon $H \in \{30, 60, 128\}$, modest neural hyperparameter sweeps, and alternate wavelet configurations when feature artifacts are rebuilt.

Auxiliary-feature ablations are not yet implemented in the released pipeline and are therefore not claimed here.

7.4 Qualitative plots

The figure-generation stage exports summary heatmaps, relative-improvement charts, training-curve grids, stepwise profiles, representative forecasts, aggregate calibration figures, and wavelet showcase panels. These figures are part of the experiment bundle, and their interpretation is used directly in the results discussion below.

8 Results

The reported experiment run is `results/report_residual_tcn_mse_20260327_232949`. According to the recorded run metadata, all three horizons used the preferred DTCWT path with the `dualtree` backend, no fallback, and 544-dimensional wavelet feature vectors. That provenance statement is run-specific and should be interpreted as a property of the saved experiment bundle rather than as an independently audited external claim.

8.1 Main comparison

Table 8 summarizes the main quantitative comparison in the test split using MAE in basis points and Pearson correlation in return units. The overall pattern is stable across all three horizons:

zero and mean baselines are best on MAE and RMSE, ridge is strongest on the nontrivial signal metrics, and none of the residual TCN models close the gap.

Model	H=30 MAE (bps)	H=60 MAE (bps)	H=128 MAE (bps)	H=30 Corr.	H=60 Corr.	H=128 Corr.
Zero	13.88	13.89	13.91	n/a	n/a	n/a
Mean	13.88	13.89	13.91	0.000	0.000	0.000
Last	20.02	20.11	20.26	0.001	-0.000	-0.000
Ridge	13.96	13.98	14.02	0.033	0.026	0.019
Raw TCN	68.57	45.58	29.60	-0.001	-0.000	-0.000
Wavelet TCN	106.43	70.52	44.43	-0.002	-0.000	-0.000
Dual-Branch TCN	99.11	77.03	49.29	0.002	0.000	-0.000

Table 8: Main test-set comparison for the reported run. MAE values are shown in basis points for interpretability, while correlation remains in the original return domain.

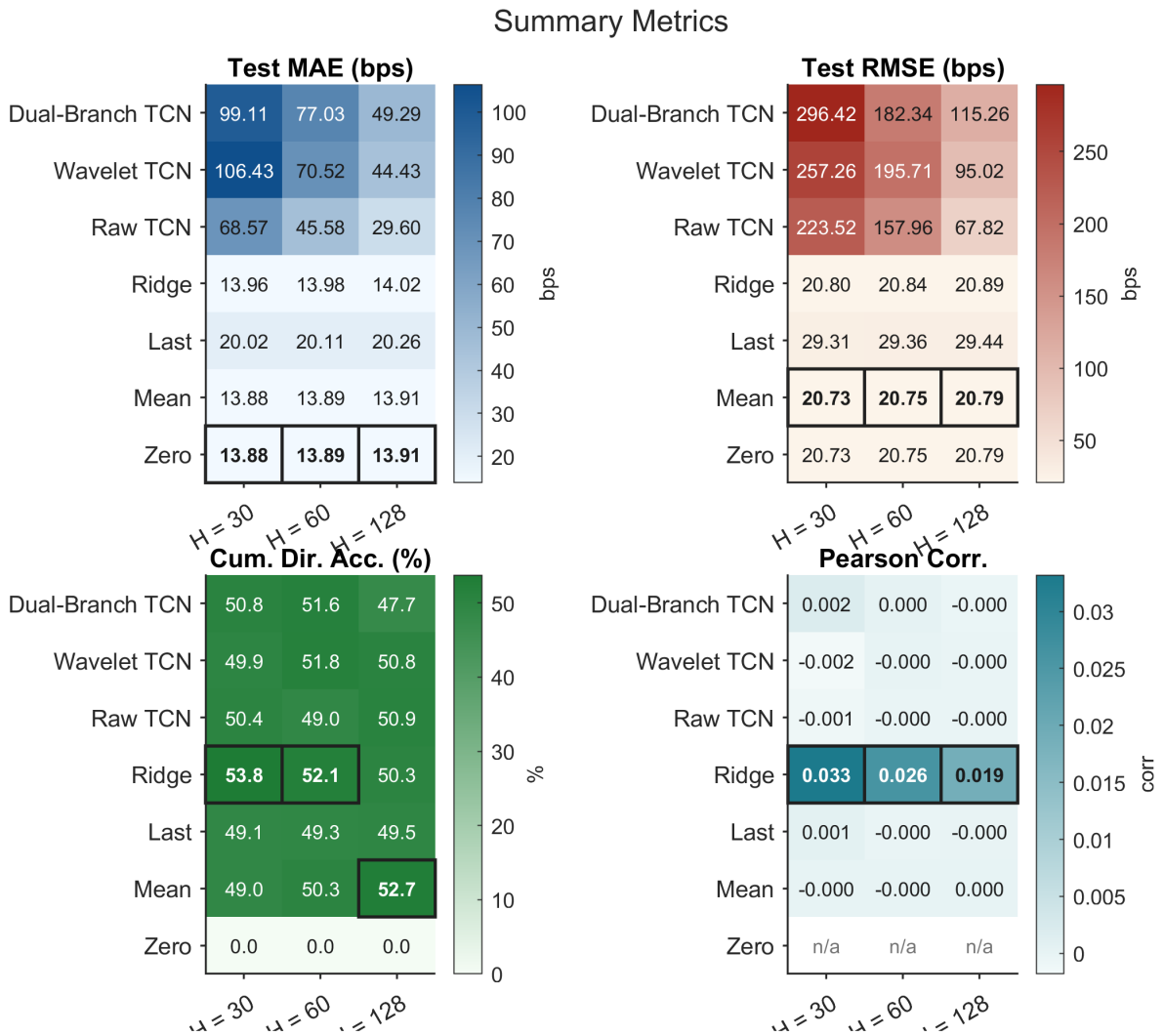

Figure 2: Summary heatmaps for the reported run. The page-7 heatmap matrix shows that zero and mean baselines dominate MAE and RMSE, ridge dominates correlation and much of the directional-accuracy comparison, and the residual TCN variants remain clearly behind on the primary error metrics.

8.2 Interpretation

In this single reported run, the empirical conclusion is that wavelet-enhanced residual TCNs do not improve short-horizon BTCUSDT return forecasting under the current direct multi-step setup. The main outcomes are summarized in Table 9. Zero and mean baselines win on MAE and RMSE, ridge dominates the nontrivial signal metrics, raw TCN is the strongest neural model, and the additional wavelet representation does not improve the neural ranking.

Finding	Summary
Error metrics	Zero and mean baselines achieve the best MAE and RMSE at all horizons, at about 13.9 bps MAE and 20.7–20.8 bps RMSE.
Signal metrics	Ridge reaches 51.34%, 51.14%, and 50.61% stepwise directional accuracy and 0.033, 0.026, and 0.019 Pearson correlation for $H = 30$, 60, and 128.
Best neural model	Raw TCN is the strongest neural variant, but it remains materially worse than the baselines on the primary error metrics.
Wavelet effect	Wavelet-only and dual-branch models are weaker than raw TCN, so the additional wavelet representation does not help in this run.

Table 9: Compact summary of the main empirical findings.

The rollback to MSE did, however, improve the stability of the neural outputs relative to the earlier mixed-loss prototype. In the saved diagnostics for this run, no extreme prediction blow-ups are evident and the best validation losses fall on a sensible scale. That is a meaningful engineering improvement even though the predictive ranking itself did not change in favor of the neural models.

Aggregate calibration gives the neural models a slightly more nuanced reading. At $H = 60$, the dual-branch TCN reaches aggregate correlation 0.121 and cumulative directional accuracy 55.75%, which indicates some usable aggregate structure. But that partial aggregate signal is not enough to offset its much larger stepwise MAE and RMSE. At $H = 128$, calibration deteriorates again, especially for the dual-branch model, which has aggregate correlation -0.094 and an over-dispersed prediction scale.

The representative examples at $H = 128$ show the same pattern. The dual-branch model is nearly exact on one benign window (33.86 bps actual versus 33.79 bps predicted aggregate return), but it fails badly on harder cases. In the selected worst example the actual aggregate return is 360.98 bps while the prediction is -5422.83 bps, and in the selected high-volatility example the actual value is 1189.01 bps while the prediction is only 74.44 bps. This is consistent with a model family that can fit some typical windows but does not generalize robustly to the extreme or high-volatility cases that matter most.

9 Discussion

Several design constraints and limitations matter for interpreting the reported negative result; Table 10 collects the main ones.

Within those constraints, the current evidence supports a narrow claim: for this one-month BTCUSDT 1-minute direct multi-step forecasting problem, wavelet-informed residual TCNs do not outperform simple baselines, and the raw TCN is the only neural variant that remains even moderately competitive. That narrow framing matters because published studies have reported better results for wavelet-enhanced models in other financial or crypto forecasting settings, and because richer behavioral or market covariates can improve crypto return predictability [14,15]. The reported run therefore argues more strongly for target and objective redesign than for additional

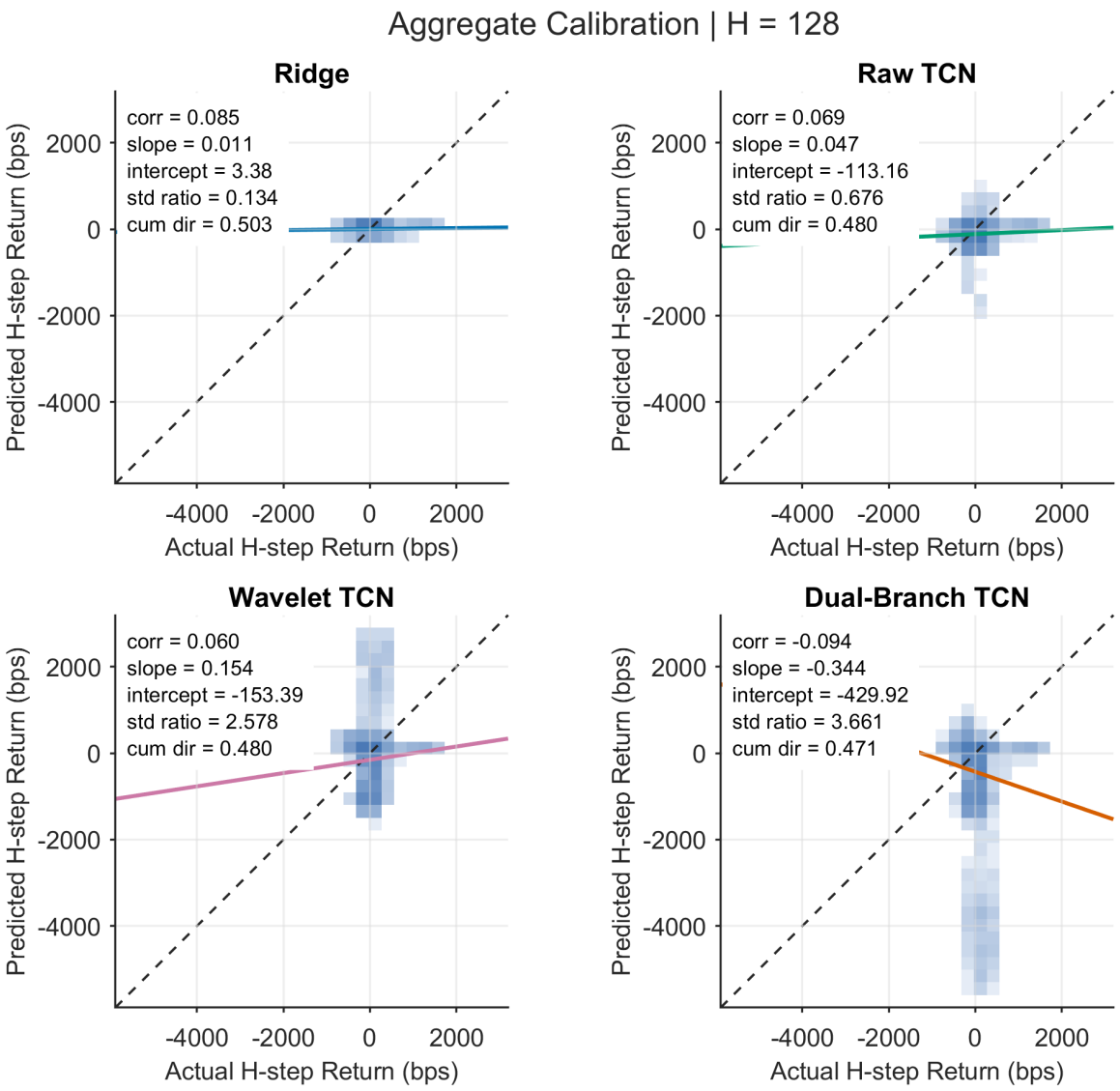

Figure 3: Aggregate calibration at $H = 128$. The page-9 calibration plots show that the raw TCN exhibits a small amount of aggregate correlation, but ridge remains better calibrated overall. The wavelet-only and dual-branch models remain poorly calibrated at this horizon, especially in output dispersion and slope.

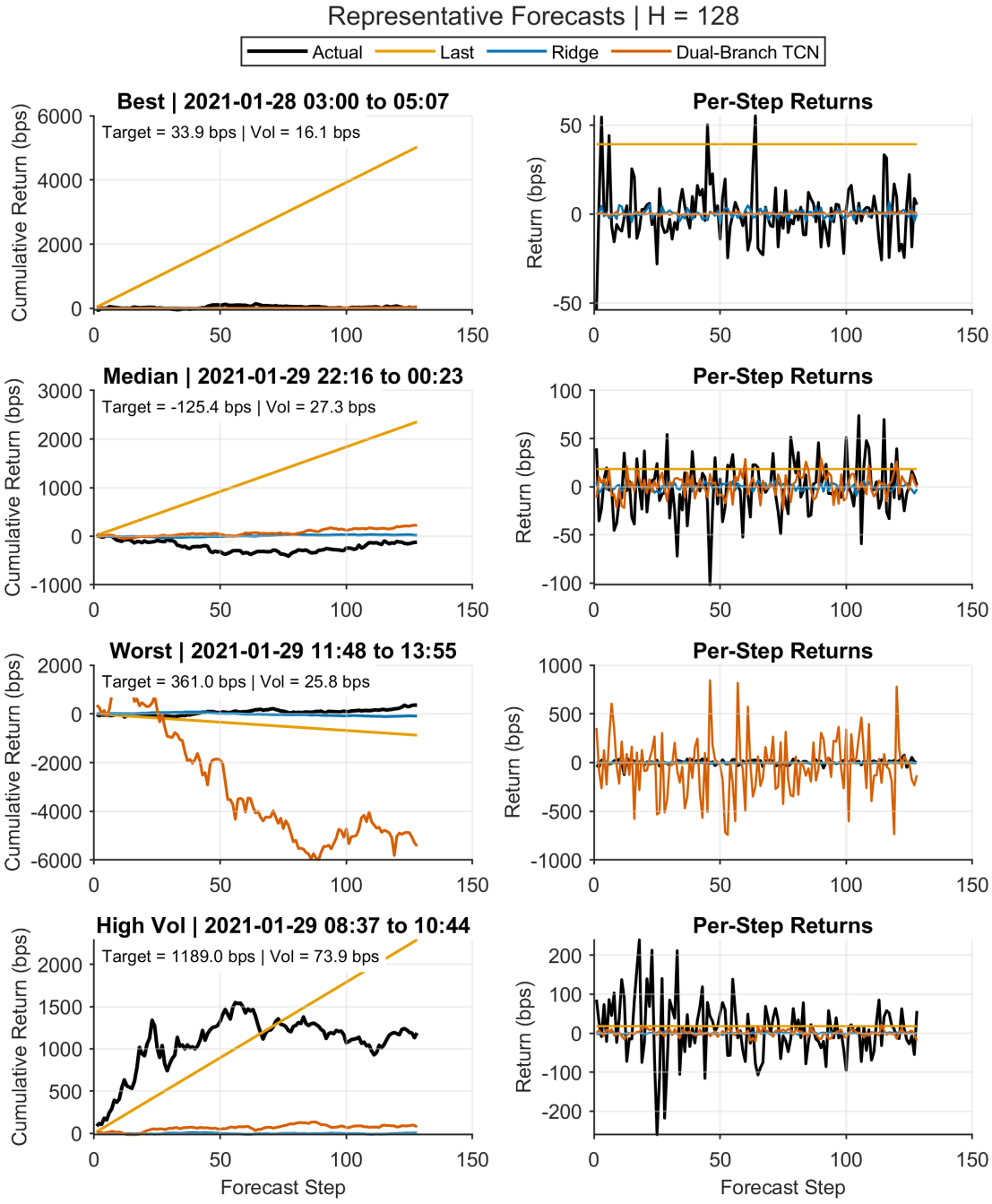

Figure 4: Representative forecast windows at $H = 128$. The page-10 forecast panel shows that the dual-branch TCN can match one low-volatility example closely, but it remains unreliable on difficult and high-volatility windows. Ridge remains conservative but substantially more stable.

Constraint	Implication
Matched evaluation	The comparison is meaningful only because all models share the same splits, targets, and return-domain metrics.
Wavelet provenance	Any wavelet claim should be conditioned on the recorded backend used in the run because fallback remains possible in principle.
Data scope	The dataset covers one asset pair over roughly one month, so external validity is limited.
Covariates	The current predictors use only lagged returns; richer OHLCV-derived covariates remain future work.
Objective	The project evaluates forecast accuracy, not trading profitability, transaction costs, or economic significance.
Generalization	Published results in other forecasting settings may differ, so the present result should be read as sample- and design-conditional rather than as a universal statement about wavelet preprocessing.

Table 10: Interpretation limits for the reported run.

architectural complexity. Plausible next steps would be to train on normalized targets, predict aggregate horizon returns rather than the full direct path, publish the exact run metadata more explicitly, or revisit the neural comparison with richer input covariates.

10 Conclusion

This report documents a completed experiment rather than a placeholder draft. The key empirical result is that wavelet-enhanced residual TCNs do not improve short-horizon BTCUSDT return forecasting in the current direct multi-step setup. Zero and mean baselines are best on MAE and RMSE, ridge is best on correlation and directional accuracy, and the raw TCN is only the best of the neural variants, not the best overall model.

The most useful lesson from the residual TCN iteration is methodological: the MSE rollback stabilized the neural forecasts relative to the earlier mixed-loss prototype, but stable training alone was not sufficient to make the wavelet representations outperform simpler baselines. The conclusion is therefore substantive for this run, but it should still be read as limited to this dataset, feature set, and reporting bundle. Future work should focus on target design, calibration, and additional covariates before investing in further architectural complexity.

A Hyperparameters

Parameter group	Setting
Targets	Lookback length $L = 256$; horizons $H \in \{30, 60, 128\}$; direct multi-step one-minute returns
Seeds	$\{42, 43, 44\}$
Normalization	Raw, wavelet, and combined inputs use z-score normalization with $\epsilon = 10^{-8}$
Wavelets	Mode `auto`; preferred backend `dtcwt`; fallback `dwt`; family `db4`; levels = 4; boundary `per`
Baselines	Ridge penalty $\lambda = 1.0$
TCN backbone	Kernel width 3; channels $[24, 32]$; dilations $[1, 2, 4, 8]$; dropout 0.15; hidden width $\max(2H, 64)$
Loss	Mean squared error by default; optional Huber+aggregate+directional prototype disabled
Optimization	Adam; learning rate 10^{-3}; batch size 128; epochs 12; validation patience 5
Tuning grid	Learning rates $\{3 \times 10^{-4}, 10^{-3}\}$; filter sets $\{[24, 32], [32, 48], [32, 64]\}$; dropouts $\{0.10, 0.15, 0.20\}$; epoch budgets $\{8, 12\}$

Table 11: Compact hyperparameter summary for the reported experiment family.

B Run Reproducibility and Verification

Table 12 summarizes the main reproducibility hooks and the corresponding verification status for the reported run.

Item	Current status
MATLAB target	R2021b
Primary toolboxes	Deep Learning Toolbox; Wavelet Toolbox for non-identity wavelet extraction; Statistics and Machine Learning Toolbox helpful but not required; Parallel Computing Toolbox optional
Dataset path	`root/project_root/data_raw/btcusdt_1m.csv`
Reported run ID	`report_residual_tcn_mse_20260327_232949`
Data span	2021-01-01 00:00:00 UTC through 2021-02-01 00:00:00 UTC
Split dates	Train through 2021-01-22 16:48 UTC; validation through 2021-01-27 08:24 UTC; test from 2021-01-27 08:25 UTC onward
Seeds	42, 43, and 44
Execution hardware	NVIDIA GeForce RTX 4070 Founders Edition GPU; Intel Core i5-11600K CPU; 64 GB DDR4-3600 RAM
Wavelet provenance	Recorded run metadata indicate preferred DTCWT with the `dualtree` backend, no fallback, levels=4, and 544 dimensions at all horizons
Saved artifacts	Config snapshot, run metadata, metrics tables, figures, figure data, and git commit hash when available
Verification	Repository structure supports unit tests, integration tests, a smoke-test runner, and CI hooks; independent verification still requires the repository URL, commit hash, and test or CI logs

Table 12: Reproducibility summary for the reported run.

The reported experiment was run on a workstation with an NVIDIA GeForce RTX 4070 Founders Edition GPU, an Intel Core i5-11600K CPU, and 64 GB of DDR4 memory at 3600 MT/s. Hardware is not fixed by the repository configuration itself, but recording the execution platform improves reproducibility for future reruns.

References

[1] Binance Open Platform. "Market Data Endpoints: Kline/Candlestick data." https://developers.binance.com/docs/binance-spot-api-docs/rest-api/market-data-endpoints. Accessed March 2026.

[2] R. Cont. "Empirical properties of asset returns: stylized facts and statistical issues." *Quantitative Finance*, 1(2):223–236, 2001.

[3] R. J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. 3rd ed. OTexts, 2021. https://otexts.com/fpp3/.

[4] Scikit-learn developers. "Common pitfalls and recommended practices." https://scikit-learn.org/stable/common_pitfalls.html. Accessed March 2026.

[5] MathWorks. "zscore." https://www.mathworks.com/help/stats/zscore.html. Accessed March 2026.

[6] N. G. Kingsbury. "Complex wavelets for shift invariant analysis and filtering of signals." *Applied and Computational Harmonic Analysis*, 10(3):234–253, 2001.

[7] MathWorks. "Dual-Tree Complex Wavelet Transforms." https://www.mathworks.com/help/wavelet/ug/dual-tree-complex-wavelet-transforms.html. Accessed March 2026.

[8] MathWorks. "wavedec: Multilevel 1-D discrete wavelet transform." https://www.mathworks.com/help/wavelet/ref/wavedec.html. Accessed March 2026.

[9] A. E. Hoerl and R. W. Kennard. "Ridge regression: Biased estimation for nonorthogonal problems." *Technometrics*, 12(1):55–67, 1970.

[10] S. Bai, J. Z. Kolter, and V. Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." arXiv:1803.01271, 2018.

[11] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization." In *International Conference on Learning Representations (ICLR)*, 2015. arXiv:1412.6980.

[12] MathWorks. "Regression Output Layer." https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.regressionoutputlayer.html. Accessed March 2026.

[13] MathWorks. "TrainingOptionsADAM." https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.trainingoptionsadam.html. Accessed March 2026.

[14] M. Vogl, P. G. Rötzel, and S. Homes. "Forecasting performance of wavelet neural networks and other neural network topologies: A comparative study based on financial market data sets." *Machine Learning with Applications*, 8:100302, 2022.

[15] K. Dunbar and J. Owusu-Amoako. "Predictability of crypto returns: The impact of trading behavior." *Journal of Behavioral and Experimental Finance*, 39:100812, 2023.