[DRAFT] Temporal Convolutional Networks for Binary Option Forecasting in Systematic Trading

Andrea Landini

University of Liechtenstein
e-mail: andrea.landini@uni.li
https://andrealandini.info

September 29th, 2025

ABSTRACT

This document presents the mathematical formulation and the architectural summary of a Temporal Convolutional Network (TCN) trained for multi-horizon binary classification on ETHUSDT data. The model predicts the probability that the close price exceeds a strike (defined as the hourly open) across multiple horizons: 5, 10, 20, 30 minutes, and at the hourly expiration.

Key words. machine learning - deep learning - convolutional networks - option pricing

1. Introduction

Machine learning has advanced tremendously since Frank Rosenblatt's introduction of the first perceptron in 1957. By 2025, its impact is so pervasive that it is difficult to identify any field left untouched. With the aim of pursuing a career in quantitative finance, I focus on stochastic differential equations (SDEs) and machine learning as foundational tools. As an exercise, I applied them to pricing options on cryptocurrencies, starting with ETH, in order to identify potential mispricings. The inherently high risk of such contracts is a feature I specifically targeted, since their prices range between 0 and 1 (as will be explained later), and at expiration the payoff collapses to one of these two extremes.

Thus, it is likely that the price will float enough to be mispriced. I planned to have a signal that could guide me to build a trading logic entirely based on ML. In the next section, I describe the one-hour binary option market that serves as benchmark; Section 3 introduces the model; Section 4 presents the results; and the Appendix provides further details on the model, the outcomes, and the project structure.

2. Market Benchmark: One-Hour Cash-or-Nothing Binary

I became interested in cryptocurrencies due to the abundance of readily available data, which is crucial for training machine learning models. With increasing attention on these markets, a segment has emerged that revolves around contracts allowing participants to bet on whether the price will be higher or lower than its opening level one hour earlier. Such contracts can be viewed as cash-or-nothing binary call options with maturity $T = 1$ hour and strike K, whose payoff is $Q\mathbf{1}_{\{S_T > K\}}$, where $Q > 0$ is the fixed cash amount paid if the event $\{S_T > K\}$ occurs.

Under the Black–Scholes model and the risk-neutral measure $\mathbb{Q}$, the time–0 value is

$$V_0^{\mathrm{BS}} \;=\; Q\, e^{-rT}\, \Phi(d_2), \qquad d_2 \;=\; \frac{\ln(S_0/K) + (r - q - \tfrac{1}{2}\sigma^2)T}{\sigma\sqrt{T}},$$

where S_0 is the spot price, r is the risk-free rate, q is the continuous dividend (or convenience yield) rate, σ is the volatility, T is the time to maturity, and Φ denotes the standard normal CDF.

In our setting the market behaves as a binary whose resolution horizon is exactly one hour; we therefore take $r = 0$ and $q = 0$, and set the final prize to $Q = 1$. With these conventions the price simplifies to the risk-neutral probability that the event occurs:

$$V_0^{\mathrm{BS}} \;=\; Q\,\Phi(d_2) \;=\; \Phi(d_2), \qquad d_2 \;=\; \frac{\ln(S_0/K) - \tfrac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}.$$

Thus, the market quote coincides with $\mathbb{Q}(S_T > K)$. Throughout we work in native (one-hour) units: T is one hour and σ is the corresponding one-hour volatility (not annualized), so that the product $\sigma\sqrt{T}$ is dimensionally consistent.

Option prices in this market lie within the interval $[0, 1]$, directly reflecting the probabilities of the underlying finishing above or below the strike. To evaluate trading outcomes, I report profits and losses as raw deltas, since both a trader and a gambler would prefer thinking in terms of changes to their bankrolls. This representation makes the implications more transparent. For example, buying 10 shares at a price of 0.5 requires a capital outlay of 5; selling them later at 0.6 returns 6, for a net profit of 1, i.e., 20%. By contrast, purchasing 10 contracts at 0.05 and selling at 0.06 also corresponds to a 20% relative gain, but the actual profit is only 0.1 on a bankroll of 10, that is, 1%. In other words, identical percentage returns can correspond to very different impacts on capital, which is why raw deltas provide a clearer picture.

Table 1: Performance metrics across NN prediction horizons, rounded to 3 decimals.

Horizon	Total return	Sharpe	Win rate	Max drawdown	Avg win	Avg loss	Win/loss ratio	Expectancy/trade
t+5	2.504	0.302	0.409	-0.640	0.484	0.185	2.612	0.089
t+10	3.166	0.400	0.455	-0.246	0.486	0.141	3.445	0.144
t+20	1.426	0.166	0.455	-0.615	0.443	0.250	1.770	0.065
t+30	1.506	0.177	0.455	-0.755	0.455	0.254	1.794	0.068
T	1.626	0.192	0.455	-0.686	0.458	0.246	1.861	0.074

3. Model

The model is a temporal convolutional network (TCN) that maps the past 60 minutes of normalized features $\{x_{t-59}, \ldots, x_t\}$ into multi-horizon probabilities. A stack of dilated one-dimensional convolutions extracts temporal dependencies, and the representation at the current time t is projected through a fully connected layer. The output is a vector

$$\hat{y}_t = \begin{bmatrix} P(\Delta_{t,5} > 0) \\ P(\Delta_{t,10} > 0) \\ P(\Delta_{t,20} > 0) \\ P(\Delta_{t,30} > 0) \\ P(\Delta_{t,T} > 0) \end{bmatrix}.$$

where $\Delta_{t,h} = \text{close}_{t+h} - \text{strike}_t$ is the price difference between the future close at horizon h minutes and the strike defined by the hour open. Thus, each component of $\hat{y}_t$ is the probability that the price at horizon h ends above the strike.

With these probabilities we can compare the model's forecast to the market-implied price and generate trading signals by taking long positions when the forecast crosses above and short positions when it crosses below.

4. Results

Table 1 summarizes the trading outcomes obtained by applying the signals across horizons. The t+10 horizon delivers the strongest performance, with a cumulative return of 3.166, a Sharpe ratio of 0.4, and the shallowest drawdown. Other horizons are still profitable, but their returns are smaller and the drawdowns more severe. The win rate hovers around 45% for all horizons, yet profits accumulate because winning trades are on average larger than losing trades, producing win/loss ratios between 1.7 and 3.4. This asymmetry highlights that the strategy succeeds not by being frequently right, but by ensuring that gains outweigh losses when it is right.

It is important to emphasize that the reported results are expressed per USD invested, which makes them appear almost too good to be true. Given the structure of the contract, high volatility is expected, and therefore the Sharpe ratios remain low. The key takeaway, however, is not the absolute performance metrics but the fact that directional accuracy can be extracted from the signal (as detailed in Appendix B). The trading outcomes ultimately depend on how effectively this signal is implemented. For the purpose of this study, it is enough to note that the TCN provides a signal with positive growth potential (3.5 at the best horizon t+10, corresponding to a total return of 2.5). Results of this magnitude are extraordinary, which makes it necessary to turn attention to the shortcomings of the current work.

First, the analysis is based on a small sample of only 22 contracts. The encouraging part is that even on such limited data the model shows an edge, but proper evaluation requires testing on thousands of contracts. By the time this paper is read, additional time series will already have been collected, and scaling up the dataset is only a matter of time. Second, on a practical level, I did not save bid quotes or the ask side for the binary put option (the only mechanism to short in this market), and instead computed everything on the ask price. This is a significant limitation, as the spread in these contracts can be wide, though the model is primarily designed to capture directional moves for hedging rather than to scalp small price differences. Future iterations will correct this and incorporate both sides of the market.

The largest limitation, however, lies in risk management. The current signal strategy has no explicit take-profit or stop-loss beyond reacting to whether the market price crosses the prediction line. This means the model could enter at 0.05 and be forced to sell at 0.02 just before expiration, or churn excessively if the market oscillates around the signal, with spread costs eroding returns. While the results so far are positive, they make clear that robust risk management must be built into the trading logic from the ground up. The machine learning signal is already sufficiently dynamic to follow market movements; what remains is to design trading rules that prevent overtrading and preserve the edge provided by the model.

5. Conclusions

This is only one of the many models that I trained before obtaining results. Some of them yielded useless predictions, while for most of them I could not see how to build a consistent trading logic around the outputs. The greatest conclusion from this process is that machine learning by itself does not provide a sufficient framework to generate reliable predictions, and that a proper risk management logic must be built on top of it, with separate directives and priorities. While live-testing this logic as it stands, one possible improvement would be to employ machine learning to parametrize a stochastic differential equation (SDE) model.

The difficulty in this case lies in formulating a suitable model in the first place; nonetheless, machine learning could play the role of a calibration engine, estimating the drift, diffusion, and jump components dynamically from observed data. In particular, ML can be used to learn the joint dynamics of strikes, maturities, and time, while incorporating external features such as realized volatility, order flow, or liquidity indicators. Embedding these learned relationships into the parametrization of an SDE would enforce structural consistency, producing predictions that live inside a financially meaningful model rather than as isolated numerical outputs.

Appendix A: Temporal Convolutional Network for Multi-Horizon Classification

The Temporal Convolutional Network (TCN) used in this work takes as input the last $L = 60$ minutes of ETHUSDT market data, where each time step is represented by $d = 11$ features. The input sequence can be written as

$$X = \{x_{t-L+1}, x_{t-L+2}, \ldots, x_t\}, \quad x_i \in \mathbb{R}^d.$$

Each convolutional layer with kernel size $k = 3$ and dilation factor $d_l = 2^l$ computes

$$h_t^{(l)} = \sigma\left(\sum_{j=0}^{k-1} W_j^{(l)} \cdot h_{t-j\cdot d_l}^{(l-1)} + b^{(l)}\right),$$

where $W_j^{(l)}$ are kernel weights, $b^{(l)}$ are biases, and σ is the ReLU activation function. Dilated convolutions expand the receptive field exponentially, allowing the model to capture both short-term and long-term dependencies efficiently. After the final convolutional layer the hidden representation $h_t^{(L)} \in \mathbb{R}^{16}$ is reduced to a fixed-size vector z_t, which is passed through a fully connected layer. The network output is given by

$$\hat{y}_t = \sigma(W_{\text{fc}} z_t + b_{\text{fc}}), \quad \hat{y}_t \in (0,1)^5,$$

corresponding to the probability that the close price exceeds the strike at horizons of 5, 10, 20, and 30 minutes, as well as at the hourly expiration. Training is performed with the binary cross-entropy loss

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{5}\Big[y_{ij}\log(\hat{y}_{ij}) + (1-y_{ij})\log(1-\hat{y}_{ij})\Big],$$

where y_{ij} are the observed binary labels. The network is lightweight, with a total of 5,829 trainable parameters, which makes it suitable for real-time applications.

Layer (type)	Output Shape	Param #
Conv1d-1	[−1, 32, 62]	1,088
ReLU-2	[−1, 32, 62]	0
Dropout-3	[−1, 32, 62]	0
Conv1d-4	[−1, 32, 66]	3,104
ReLU-5	[−1, 32, 66]	0
Dropout-6	[−1, 32, 66]	0
Conv1d-7	[−1, 16, 74]	1,552
ReLU-8	[−1, 16, 74]	0
Dropout-9	[−1, 16, 74]	0
Linear-10	[−1, 5]	85
Total		**5,829**

Table A.1: Layer-by-layer architecture summary of the TCN-Multi model.

The architecture summary in Table A.1 highlights that the network remains compact, with fewer than six thousand parameters in total, yet retains the ability to capture temporal dependencies at multiple scales through dilated convolutions. This lightweight design is particularly suitable for real-time trading systems, where both inference speed and memory efficiency are critical. Figure A.1 illustrates the overall computation graph of the model, emphasizing the sequential convolutional blocks followed by the fully connected output layer that jointly predicts the multi-horizon probabilities.

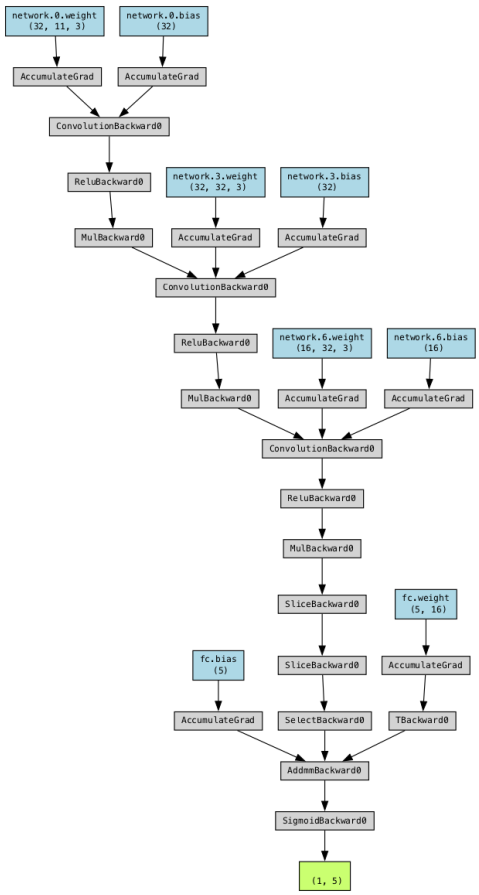

Fig. A.1: Computation graph of the TCNMulti model.

As shown in Table A.1, the overall parameter count remains below six thousand, which makes the network highly efficient compared to recurrent alternatives such as LSTMs or GRUs. Despite its compactness, the use of dilated convolutions expands the receptive field exponentially, enabling the model to integrate both short-term fluctuations and longer-term hourly trends. This balance between parsimony and expressiveness is well suited to systematic trading applications, where low latency and robustness to overfitting are essential. Figure A.1 further clarifies the computation flow: successive convolutional blocks extract hierarchical temporal features, which are then condensed into a fixed-size representation for joint prediction across multiple horizons. In this way, the architecture leverages temporal structure while keeping inference fast and interpretable.

Appendix B: Results in more depth

Despite having trained many models with initially promising results, most failed to produce a meaningful trading signal once tested in practice. The approach presented here, however, shows more consistent potential. In this setup, the neural network generates predictive margins: when the market price crosses below the margin it indicates a bearish movement, whereas convergence of the prediction toward the market price suggests a bullish trend. At first glance this might appear trivial, akin to saying "when the price goes up, it goes up." Yet this interpretation does not hold once the framework is tested across a sufficiently large sample of contracts, which is the direction of my ongoing work. Even in preliminary random selections, a consistent pattern emerges, pointing to the possibility of a robust and repeatable trading signal.

Building on this idea, I designed a signal logic that operates as follows. The signal is generated whenever the model–predicted probability p_t crosses the market ask–up a_t. A long signal is issued if

$$\text{signal}_t = +1 \quad \text{when } p_{t-1} \le a_{t-1} \text{ and } p_t > a_t,$$

while a short signal is issued if

$$\text{signal}_t = -1 \quad \text{when } p_{t-1} \ge a_{t-1} \text{ and } p_t < a_t,$$

and otherwise $\text{signal}_t = 0$. Once a position is opened at entry price x_e and closed at price x_t, the realized profit and loss is

$$\text{PnL}_t = \begin{cases} x_t - x_e, & \text{if long,} \\ x_e - x_t, & \text{if short.} \end{cases}$$

Any position left open at the final time T is force–closed at x_T, and cumulative performance is computed as

$$\text{CumPnL}_t = \sum_{i=1}^{t} \text{PnL}_i.$$

Table B.1 reports the distributional properties of cumulative PnL across horizons. With only 22 observations per horizon, the statistics should be interpreted cautiously, yet some patterns stand out. The mean is positive for all horizons, with the highest values at t+10 (0.144) and t+5 (0.114). The medians, however, are closer to zero or negative, indicating that profitability is concentrated in a subset of trades rather than evenly distributed. This is consistent with the relatively large standard deviations (around 0.36–0.39) and wide ranges, with minimum losses approaching −0.6 and maximum gains near 0.9. The upper quartiles (Q_{75}) are substantially positive across horizons, which suggests that while typical outcomes may be modest or even negative, the right tail contains strong winners that drive the positive means. In other words, the strategy benefits from payoff asymmetry: losses are frequent but bounded, while occasional large gains account for most of the profitability.

This pattern is acceptable as long as the model is able to predict the correct direction, since by construction the contract settles to either 0 or 1 at maturity, allowing gains to outweigh losses. With a larger dataset, a machine learning model could be trained to address this issue more effectively, yet the more pressing limitation lies in the absence of a proper trading logic beyond the raw signal. The strategy could already be improved by introducing simple constraints from the trader, such as avoiding entries after a certain time t or ignoring signals below a predefined threshold.

Table B.1: Cumulative PnL results and summary statistics across NN prediction horizons.

contract	t+5	t+10	t+20	t+30	T
Sep 24, 3AM	-0.110	-0.120	-0.190	-0.290	-0.270
Sep 24, 4AM	0.810	0.720	0.710	0.670	0.670
Sep 24, 5AM	0.000	0.660	0.610	0.600	0.620
Sep 24, 6AM	-0.080	-0.120	-0.250	-0.330	-0.330
Sep 24, 9AM	-0.070	-0.090	-0.160	-0.250	-0.220
Sep 24, 10AM	0.340	0.340	0.310	0.280	0.290
Sep 24, 11AM	-0.570	-0.080	-0.440	-0.440	-0.440
Sep 24, 12PM	0.390	0.390	0.390	0.390	0.390
Sep 24, 1PM	0.368	0.368	0.268	0.358	0.358
Sep 24, 2PM	0.000	-0.050	-0.150	-0.240	-0.240
Sep 24, 3PM	-0.300	-0.300	-0.300	-0.300	-0.300
Sep 24, 4PM	0.920	0.850	0.800	0.800	0.800
Sep 24, 5PM	-0.020	-0.130	-0.130	-0.130	-0.130
Sep 24, 8PM	-0.010	-0.030	-0.170	-0.170	-0.170
Sep 24, 9PM	-0.104	-0.104	-0.154	-0.154	-0.154
Sep 24, 10PM	0.330	0.330	0.270	0.280	0.280
Sep 24, 11PM	0.000	-0.078	-0.168	-0.208	-0.168
Sep 25, 12AM	-0.570	-0.570	-0.630	-0.270	-0.270
Sep 25, 1AM	0.450	0.450	0.360	0.540	0.540
Sep 25, 2AM	-0.020	-0.020	-0.260	-0.260	-0.260
Sep 25, 3AM	0.420	0.420	0.380	0.380	0.380
Sep 25, 4AM	0.330	0.330	0.330	0.250	0.250
Count	22	22	22	22	22
Mean	0.114	0.144	0.065	0.068	0.074
Min	-0.570	-0.570	-0.630	-0.440	-0.440
25%	-0.078	-0.101	-0.185	-0.258	-0.255
50%	0.000	-0.025	-0.140	-0.142	-0.142
75%	0.361	0.385	0.353	0.375	0.375
Max	0.920	0.850	0.800	0.800	0.800
Std	0.377	0.360	0.390	0.386	0.384

Moreover, the current signal design is admittedly primitive, but the fact that it achieves a positive and satisfactory growth rate suggests that performance can only improve once more sophisticated trading rules are integrated.

Appendix C: Project Directory Structure

In this appendix, the focus is placed on the supporting infrastructure that underpins the main trading framework. The `utils` directory implements the essential services that enable the system to manage internal state, coordinate asynchronous processes, and deliver reliable inputs to the higher-level trading logic.

```
├── data
│   ├── bets.csv
│   ├── conds_passed.txt
│   ├── figures
│   ├── ledger.csv
│   ├── pnl_summary.txt
│   └── tests
├── main.py
├── models
│   ├── bnn_1min.pt
│   ├── nn_model_multi.pt
│   ├── tcn_multihorizon.pt
│   ├── tcn_trendrev.pt
│   ├── tcn_trendrev_houropen.pt
│   ├── tcn_vol_best.pt
│   └── vol_model.pth
├── requirements.txt
├── standalone
│   ├── __init__.py
│   ├── manual-trading.ipynb
│   ├── pnl_analysis.py
│   └── visual.py
├── storage
│   ├── ETHUSDT.txt
│   └── train_multihorizon.py
├── trading
│   ├── __init__.py
│   ├── __pycache__
│   ├── back_tester.py
│   ├── risk_manager.py
│   ├── trade.py
│   └── trade_helpers.py
└── utils
    ├── __init__.py
    ├── __pycache__
    ├── bet_manager.py
    ├── bot_state.py
    ├── config.py
    ├── stream_odds.py
    └── stream_underlying.py
```

The file `bet_manager.py` is responsible for discovering and maintaining betting opportunities. It retrieves candidate markets, filters them by activity and expiration, and attaches the relevant start and end times. By persisting this information in `bets.csv`, and by updating auxiliary records such as `conds_passed.txt` and `ledger.csv`, it ensures that only valid opportunities are considered at any given time. In addition, `bet_manager.py` supplements these records with corresponding reference prices, thereby providing the baseline for subsequent decisions.

The state of the entire trading system is centralized in the dataclass defined in `bot_state.py`. This structure, called `BotState`, stores identifiers for current and upcoming conditions, order book odds, market times, and reference price information. It also manages asynchronous event flags, which coordinate the independent streaming tasks. As such, `BotState` functions as the shared memory of the system, ensuring consistency across concurrent processes.

The file `stream_odds.py` implements the streaming interface for odds data. Through continuous subscription and monitoring, it reconstructs the order books of the relevant instruments and extracts the best bid and ask levels. These values are injected into the shared state in real time, allowing the trading logic to evaluate odds as they evolve. To safeguard reliability, the streamer handles stale connections, reconnects automatically, and maintains responsiveness through asynchronous event signaling.

Finally, `stream_underlying.py` establishes a persistent streaming connection to capture underlying price dynamics. It records both candlestick data and order book depth, updating the shared state with each new message. This integration provides the essential link between the odds data and the underlying asset, allowing the framework to compare relative information consistently over time.