# Predicting Bitcoin Mid-Price Trends Using Deep Learning Models

Antoine TURKIE, Aymane HAOULANI, Redouane HADNI
*Ecole Polytechnique Féderale de Lausanne, Switzerland*

***Abstract*—This study explores mid-price trend prediction in Bitcoin markets using Limit Order Book data. We evaluate various models, including Logistic Regression, LSTM, CNN-LSTM, and transformers, and propose a novel architecture, BiTran. By incorporating Time Absolute Position Encoding (tAPE) and optimized attention mechanisms, BiTran achieves notable performance improvements, particularly over existing transformer-based methods. These findings highlight the potential of transformers for financial trend forecasting, especially for longer prediction horizons.**

## I. Introduction

One fundamental data structure used in the financial markets is the **Limited Order Book (LOB)** 1. A LOB is a collection of pending buy and sell orders arranged by price levels. It continuously updates as orders are placed, filled, or canceled, providing a granular view of supply and demand at varying price points. Each entry in the LOB includes bid prices, ask prices, and their associated volumes.

In this work, **we focus on predicting the trend of the *mid-price* (the average of the best bid and best ask prices) of Bitcoin using various architectures**. Specifically, we use a Bitcoin LOB with a depth of 10 price levels on each side (bids and asks).

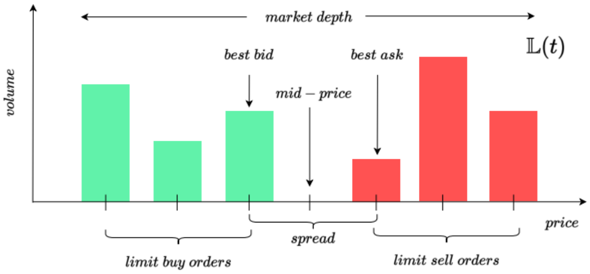

Fig. 1: Graphical Representation of a LOB.

## II. Preprocessing

### A. Data collection

We collect Limit Order Book data using Binance's WebSocket API, streaming bid and ask prices along with volumes for specified depth levels. Our script processes updates in real-time, appending timestamps and saving the data in structured, timestamped CSV files organized by date.

### B. Encoding the Mid-Price Evolution

We define the mid-price at time $t$ as:

$$\text{Midprice}_t = \frac{\text{Best Bid Price}_t + \text{Best Ask Price}_t}{2}.$$

To predict the future trend, we consider a horizon $k$ (e.g., 10, 20, 50, 100 timesteps into the future). We compute the percentage variation of the mid-price over the next $k$ steps using a smoothed approach:

$$\text{Variation}(t,k) = \frac{\overline{\text{Midprice}}_{t+1:t+k} - \text{Midprice}_t}{\text{Midprice}_t} \times 100,$$

where $\overline{\text{Midprice}}_{t+1:t+k}$ is the average mid-price over the next $k$ steps:

$$\overline{\text{Midprice}}_{t+1:t+k} = \frac{1}{k}\sum_{i=1}^{k}\text{Midprice}_{t+i}.$$

Using the average mid-price over the next $k$ steps provides a more robust measure of future price movement by mitigating the impact of short-term fluctuations and noise inherent in high-frequency trading data.

We define a threshold parameter $\theta$ to categorize this variation into three classes:

$$\text{label}(t,k) = \begin{cases} D & \text{if Variation}(t,k) < -\theta, \\ S & \text{if } -\theta \leq \text{Variation}(t,k) \leq \theta, \\ U & \text{if Variation}(t,k) > \theta. \end{cases}$$

**These labels "D" (Down), "S" (Stable), and "U" (Up) serve as the target classes for our classification models**.

### C. Feature Engineering

Ntakarisa et al. [1] showed that handcrafted econometric features outperform automated feature extraction methods like LSTM autoencoders.

We have therefore created the following features :

- **Volume-based features:** Cumulative bid and ask volumes for the top $k$ levels, and volume imbalance measures.
- **Price-based features:** Weighted average prices over top levels, relative spread between best ask and best bid, and rolling statistical measures (mean, standard deviation) of the mid-price over various windows.

- **Temporal features:** Time of day (in fractional hours), helping capture intraday seasonalities.

### D. *Data Imbalance and Scale*

#### 1) *Data Imbalance*

In October, Bitcoin exhibited a bullish trend, leading to naturally imbalanced class distributions in the dataset. Given the substantial size of the dataset, we introduce class weights based on label distribution. This approach assigns higher weights to underrepresented classes, penalizing errors on these classes more heavily during training:

$$\text{weight}_i = \frac{n}{k \cdot n_i}$$

where:

- $n$ is the total number of samples,
- $k$ is the number of unique classes,
- $n_i$ is the number of samples in class $i$.

#### 2) *Standardization*

We use standardization over min-max scaling to bring values to a comparable scale, leveraging the Central Limit Theorem's assumption of Gaussian distribution thanks to the large dataset size.

## III. Training

### A. *Common choices between the chosen architectures*

#### 1) *Large data handling*

Limited order book (LOB) data is substantial, with approximately 800,000 data points generated in a single trading day. Since we plan to use data spanning 10 days, processing this volume of data directly in our models is infeasible due to computational constraints. To address this, we adopted two key strategies: **upgrading our computational resources** and **implementing chunk-based and batch-based data processing for training and evaluation**.

##### a) *Computational Resources*

To meet the high computational demands, we upgraded to Google Colab Pro, which provides access to powerful L4 machines equipped with **24 GB of VRAM and 52 GB of RAM**.

##### b) *Chunks and Batches*

To handle the large dataset, we divided it into manageable chunks and further split these chunks into mini-batches for incremental processing 2.

##### c) *Mini-batch Size*

The selection of mini-batch size is a critical factor that influences training performance and accuracy. A smaller batch size for instance introduces stochasticity into the optimization path, which can slow down convergence but often leads to improved generalization.

In our experiments to select the optimal value, we utilized a transformer architecture and varied the mini-batch size while keeping the number of epochs fixed. The outcomes of these experiments are the following:

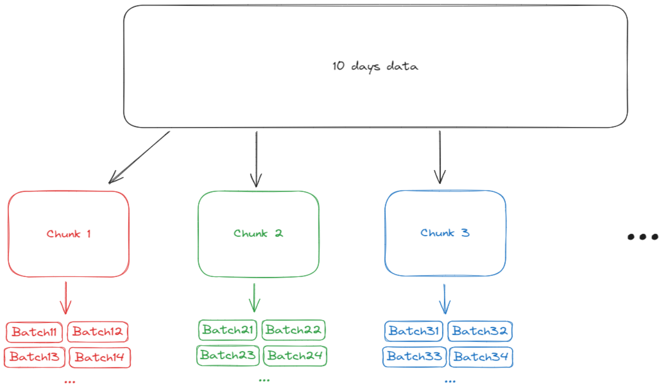

Fig. 2: Large data handling using chunks and batches.

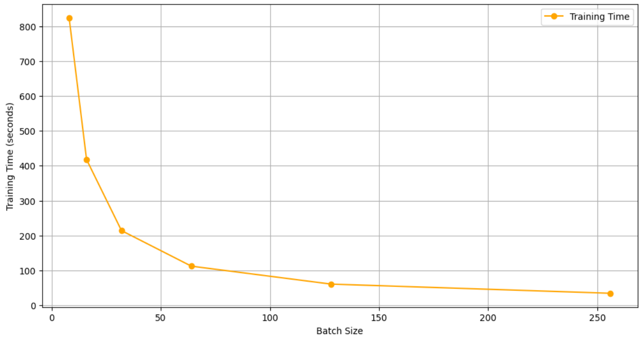

Fig. 3: Evolution of the training time with the batch size.

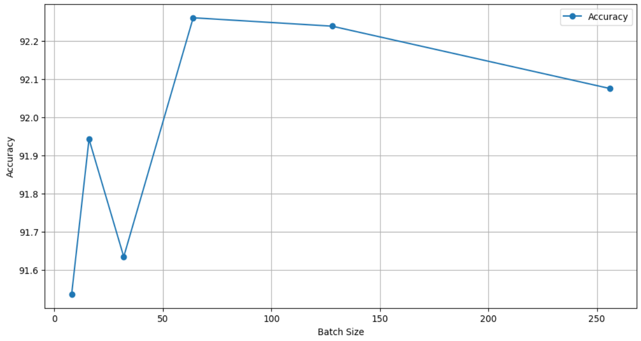

Fig. 4: Evolution of the testing accuracy with the batch size.

The plots validate the expected trade-off between training time 3 and testing accuracy 4. Based on our results, **a mini-batch size of 64 emerged as the optimal choice, balancing both efficiency and performance effectively**.

#### 2) *Optimization Problem*

##### a) *Choice of the Loss Function*

Since the task involves predicting categorical variables, the appropriate loss function is the **cross-entropy loss**. This loss function measures the dissimilarity between the predicted class probabilities and the true labels.

*b) Optimization Method*

Y. Pan and Y. Li from Carnegie Mellon University proposed in their paper [4] an attempt to formally justify why accelerated gradient descent methods like **Adam** outperform Stochastic Gradient Descent in deep learning models. Their justification hinges on the concept of *directional sharpness*, which measures the sensitivity of the loss landscape to parameter updates.

Convinced by their findings, we chose **Adam** as the optimizer for our different deep learning architectures. Its adaptive learning rate mechanism is well-suited for handling the high-dimensional parameter spaces and variability inherent to our dataset.

### *B. Logistic Regression*

We implement a logistic regression model as a baseline for comparison. We selected a learning rate of 0.001 as it provided a good balance between convergence speed and stability. Regularization was not applied, as validation set results indicated that the model was not overfitting.

### *C. LSTM Model : Architecture and Hyperparameters*

The Long Short-Term Memory (LSTM) architecture is a simple and effective choice for modeling time series data. It improves upon the basic Recurrent Neural Network (RNN) by solving the vanishing gradient problem, enabling the model to learn long-term dependencies 5.

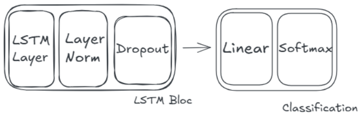

Fig. 5: LSTM Architecture.

- **Number of Layers:** 2 LSTM layers. This setup provides enough complexity to capture patterns in the data while keeping training time manageable.
- **Dropout:** A dropout rate of 0.2 was chosen to prevent overfitting. This value is kept reasonably low since the model cannot be trained with a very high number of epochs.
- **Hidden Size:** The hidden size is set to 50 units, determined through grid search.

### *D. CNN-LSTM Model : Architecture and Hyperparameters*

Omole and Enke [2] introduced a CNN-LSTM architecture for predicting Bitcoin price trends. Their model combines a 1D Convolutional Neural Network (CNN) to capture local temporal patterns with Long Short-Term Memory (LSTM) layers for modeling long-term dependencies 6. This hybrid approach takes advantage of the complementary strengths of both architectures. For consistency in evaluation, we opted to adopt the same hyperparameters and architecture for our model.

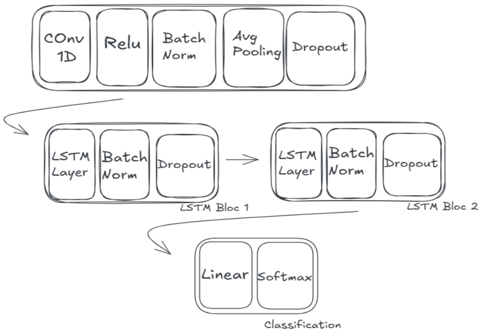

Fig. 6: CNN-LSTM architecture.

- **Convolution filters**: 64 filters with a kernel size of 3.
- **First LSTM layer hidden size**: 512 units.
- **Second LSTM layer hidden size**: 320 units.
- **Dropout**: 0.5 after LSTM layers to prevent overfitting.

### *E. Transformers*

*1) Baseline Architecture 7*

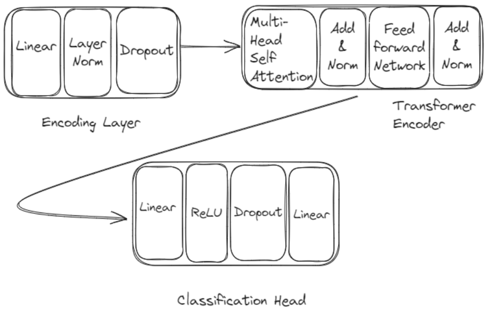

Fig. 7: Baseline Transformer architecture overview.

- **Embedding Dimension:** 64.
- **Number of Heads:** 4.
- **Number of Layers:** 2
- **Dropout:** 0.1. Prevents overfitting.
- **Weight Decay:** $1 \times 10^{-5}$, regularizes large weights.

*2) Improved Architecture BiTran*

*a) Embedding*

A key area for improvement is the embedding layer. The baseline architecture we presented does not incorporate advanced positional encoding, which is particularly useful for time series data. To address this, we explored three main methods: random/no particular positional encoding, Positional Index Embedding (PIE), and a novel method introduced in 2023 for time series data called Time Absolute Position Encoding (tAPE) [3].

Rather than simply training separate models to determine the best approach, we employed unsupervised learning techniques like t-SNE to visualize the embeddings in a low-dimensional space 8 9. This allowed us to assess

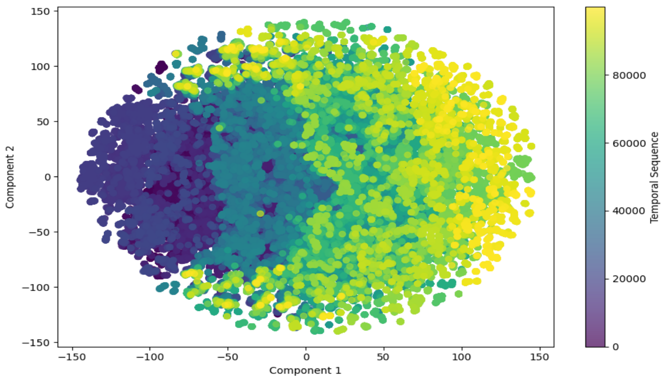

Fig. 8: t-SNE visualization of random positional encoding.

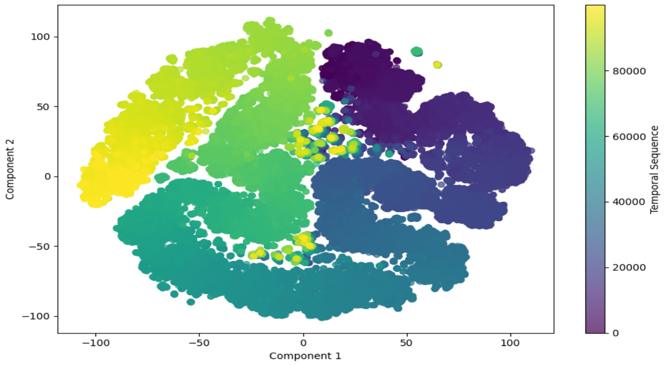

Fig. 9: t-SNE visualization of the tAPE method.

whether the embeddings effectively capture temporal relationships in the data.

As shown in the figures, the tAPE method successfully captures temporal relationships, clustering close time frames together. This enables the attention mechanism to identify relevant patterns more effectively compared to a simple projection. Based on these results, we selected tAPE as the embedding method for the improved transformer model.

*b) Attention heads*

After a grid search over the possible number of attention heads (those that divide the input dimension), 8 revealed itself as the optimal value.

## IV. Experimental Setup

To be consistent with earlier works using the same type of data, we train and test our model under the following conditions:

- **Sequence Length** ($T = 10$ timesteps): The sequence length $T$ refers to the number of past timesteps considered as input when making a prediction.
- **Training and Testing Periods:** We train the models on 7 days of data (October 15 to October 21) and test on 3 days (October 22 to October 24).
- **Choosing $\theta$:** We select $\theta$ values that produce a more balanced distribution of the three classes. For each horizon $k$, we experimented and chose the best $\theta$.
- **Early Stopping:** We employ early stopping based on validation performance to find the optimal number of training epochs. In this case, training converged within 2 epochs.

## V. Results

| Horizon | Model | Accuracy | F1-Score |
|---|---|---|---|
| k=10 | Logistic Regression | 86.2 | 87.65 |
| | LSTM | 97.21 | 97.19 |
| | CNN-LSTM | 94.58 | 94.57 |
| | Baseline Transformer | 88.23 | 88.9 |
| | Baseline + tAPE | **97.54** | **97.48** |
| | BiTran | 94.85 | 94.89 |
| k=20 | Logistic Regression | 94.38 | 94.22 |
| | LSTM | **97.58** | **97.56** |
| | CNN-LSTM | 94.83 | 94.75 |
| | Baseline Transformer | 57.52 | 56.83 |
| | Baseline + tAPE | **97.58** | **97.56** |
| | BiTran | 93.52 | 93.43 |
| k=50 | Logistic Regression | 84.15 | 83.39 |
| | LSTM | **90.87** | **90.72** |
| | CNN-LSTM | 77.93 | 78.03 |
| | Baseline Transformer | 52.38 | 40.99 |
| | Baseline + tAPE | 65.65 | 62.61 |
| | BiTran | 88.89 | 88.88 |
| k=100 | Logistic Regression | 74.01 | 72.71 |
| | LSTM | 82.22 | 81.67 |
| | CNN-LSTM | 78.70 | 78.15 |
| | Baseline Transformer | 70.75 | 66.05 |
| | Baseline + tAPE | 91.20 | 91.19 |
| | BiTran | **91.86** | **91.87** |
| Average Performance | Logistic Regression | 84.67 | 84.49 |
| | LSTM | 91.97 | 91.79 |
| | CNN-LSTM | 86.51 | 86.37 |
| | Baseline Transformer | 67.22 | 63.19 |
| | Baseline + tAPE | 87.99 | 87.21 |
| | **BiTran** | **92.28** | **92.26** |

TABLE I: Results for the horizons k=10,20,50 and 100.

Our findings demonstrate that BiTran, our improved transformer architecture, achieves the best overall performance across all horizons, outperforming models like logistic regression, CNN-LSTM, and even LSTM, which excelled in short-term predictions. BiTran's success, particularly at longer horizons, is attributed to the integration of tAPE encoding and meticulous parameter fine-tuning, which transformed the baseline transformer from an acceptable model into an excellent one.

## VI. Conclusion

We introduced BiTran, a transformer-based architecture for mid-price trend prediction, demonstrating improved performance compared to prior work, including [5], on an even larger test set. The integration of tAPE and tailored attention mechanisms proved effective in handling noisy financial data. Future work could investigate the use of ensemble methods combining BiTran with other architectures or exploring alternative positional encodings to further improve model performance across diverse prediction horizons.

## VII. Ethical risks

One ethical risk identified in our project is the over-reliance on model predictions as ground truth, particularly in the highly volatile cryptocurrency markets where sudden trend shifts or compromised data can drastically harm the model's performance. Retail investors, institutional traders, or even everyday users relying solely on the model may suffer significant financial losses if predictions deviate from reality due to poor input data, market anomalies, or systemic issues such as network disruptions or paywall limitations by financial data providers. These factors introduce errors or biases into the data pipeline, which is the backbone of the model's functionality. This risk is severe because it could erode trust in AI tools, destabilize markets, and disproportionately harm less-resourced stakeholders who cannot independently validate predictions. Given the speculative and dynamic nature of cryptocurrency markets, as well as the challenges in ensuring perfect data reliability, the severity of this risk is high, and its likelihood is moderate, based on the unpredictability of external factors such as data integrity and market shocks. To evaluate this risk, we conducted discussions with finance specialists and analyzed existing data providers reliability, uncovering potential vulnerabilities in input integrity.

To mitigate this risk, we implemented a multi-pronged approach. First, anomalies caused by network disruptions and data transfer protocols were fixed prior to feeding data into the models. A data validation pipeline was established to ensure data quality, flagging inconsistencies and anomalies before predictions are made. We reported weighted F1-scores alongside traditional accuracy metrics to assess the model's reliability and reduce the chance of overfitting or overly optimistic performance evaluations. Second, data was sourced from Binance, a trusted financial institution, to improve input reliability. Additionally, we included stakeholder education measures by adding a clear disclaimer in the repository's README file, advising users not to rely solely on the models for financial decisions. These steps aim to ensure that predictions are treated as decision-support tools rather than infallible truths, reducing harm to stakeholders and improving the robustness of the system.

## References

[1] Ntakarisa, A., Mirone, G., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2019). *Feature Engineering for Mid-Price Prediction with Deep Learning*, https://arxiv.org/pdf/1904.05384.

[2] Omole, O., & Enke, D. (2024). *Deep learning for Bitcoin price direction prediction: models and trading strategies empirically compared*, https://doi.org/10.1186/s40854-024-00643-1.

[3] N. Foumani, C. Tan, G. Webb and M. Salehi, *Improving Position Encoding of Transformers for Multivariate Time Series Classification*, https://arxiv.org/pdf/2305.16642.

[4] Y.Pan and Y.Li, *Toward Understanding Why Adam Converges Faster Than SGD for Transformers*, https://arxiv.org/pdf/2306.00204

[5] James Wallbridge, *Transformers for Limit Order Books*, https://arxiv.org/pdf/2003.00130