Project Report
Deep Learning
Bitcoin LOB

Submitted by:
Aayush Gupta, Prisha Gupta, Punnag Choudhury, Sehajpreet Kaur

Deep Learning for Bitcoin Price Movement Prediction Using Limit Order Book Data

This project studies the use of deep learning structures to forecast rapid fluctuations in Bitcoin prices in the cryptocurrency market using high-speed Limit (Order Book) LOB digital (LOB) data. It also investigates a key issue faced by many algorithmic traders regarding the extraction of significant predictive signals out of the extreme noise and complexity of very fast-paced financial data. We show how with correctly applied regularization and proper confidence based filtering techniques, it is possible to achieve impressive predictive accuracy with deep learning models and subsequently develop profitable trading strategies.

Introduction and Motivation

The Financial Context

Every single millisecond, the financial markets, especially cryptocurrency exchanges, create a tremendous amount of microstructure data. One of these data sources is the Limit Order Book, which contains all of the current pending buy/sell orders within the market at different prices. While traditional price-volume data show price and volume only, the LOB snapshots show everything; therefore they contain all of the pending orders in the market, what the market participants want to do, and how they are positioned, in real-time.

Bitcoin is the largest and most liquid cryptocurrency; because of this, it makes an excellent way to test LOB-based prediction models. Bitcoin has several unique characteristics, such as being traded 24 hours per day, having a high degree of volatility and transparency in its order flow, which creates an excellent opportunity for conducting high-frequency prediction tasks. However, due to the level of liquidity and trading volume on the Bitcoin market, the amount of noise and complexity in the market can be overwhelming.

The Core Challenge

- Predicting price movement over short time periods based on Limit on Buy (LOB) data involves several challenges that are interrelated.
- Data Characteristics - The order book is updated multiple times every second with thousands of small micro-changes, most of which reflect random fluctuations rather than useful directional price movement signals. Therefore, the signal-to-noise ratio is extremely low, making it very difficult to extract true patterns.
- Temporal Complexity - Very few financial patterns are isolated events. For example, in a given period, a small change in order imbalance at T could be influenced by events occurring at T-10 or T-40. Traditional predictive models are not able to effectively model long distance dependencies and the inter.
- Class Imbalance - The dominant class is the "flat" or "stationary" class. Most time steps will have no price movement, causing an extreme class imbalance. Therefore, a model will typically achieve high levels of accuracy by simply predicting the majority class (flat or stationary). This pattern is of no use when creating a model for actual trading.

- Non-Stationarity - Market regimes change quickly. Therefore, patterns that work when markets are calm will usually not work when they are volatile. Thus, a deep learning model must be able to generalise across different regimes and conditions.
- Overfitting Risk - Another risk associated with deep learning models is the potential for them to simply memorize the noise rather than learning the actual underlying true patterns. As a result, a model that has superior training performance may exhibit poor validation performance.

Research Objectives

The goals of this study include:

- To create and deploy a deep learning development system that can process high-frequency limit order book data and produce an accurate forecast of price changes in the financial sector.
- To build solid labeling engineering strategies that lower noise levels and opening balance discrepancies.
- To evaluate and compare model structures and variations, including classic machine learning benchmarks and up-to-date deep learning methodologies, through quantitative analysis.
- To determine if predictive capacity would result in a profitable trading strategy when simulated under real-world conditions.
- To discover the significant elements of a model's success. Elements, such as horizon option, threshold constancy, and confidence information.

Literature Review and Existing Solutions

DeepLOB (2018)

DeepLOB is the first paper to introduce a hybrid architecture that combines Convolutional Neural Networks (CNNs) with Long Short Term Memory (LSTM) for predicting limit order books (LOBs). The authors designed this model to be able to capture both spatial patterns in different price ranges and sides of an order book using CNNs, and temporal evolution of the LOB by employing LSTMs. DeepLOB was tested on data from the London Stock Exchange, and found to provide evidence that deep learning techniques can identify common characteristics of limits orders across many different financial instruments.

Some limitations of the DeepLOB approach include: potential overfitting of the model when used with different types of financial assets; being highly sensitive to the selection of hyperparameters when building the model; and difficulties in maintaining consistent performance during periods of volatile market conditions.

Better Inputs Matter More (2025)

Recent studies conclude that advanced preprocessing and engineering of your features is more crucial to the final outcome of a prediction than just adding more layers in your deep learning architecture.

In a comparative study on the Bybit BTC/USDT market data, they used many methods including Logistic Regression, XGBoost, and various types of Deep Learning architectures. The conclusion

drawn was: "when you use properly-engineered inputs and a smoothing approach, it has been shown to greatly improve model performance no matter how complex the models are."

The main drawback of this research was the dependence on quality of the preprocessing step. You will consistently get poor results if you design the inputs poorly, even if you are using an incredibly sophisticated model.

Digital Asset Limit Order Books (2020)

This work specifically addressed cryptocurrency LOB prediction using Coinbase data at 100ms precision. Using temporal CNNs, they achieved approximately 47% walk-forward accuracy on test sets. While this might seem modest, it represents a meaningful edge in high-frequency trading contexts where even slight advantages compound rapidly.

Identified Research Gaps

Our search of published research found four significant gaps in the literature.

1. Most previous research defined label based on small price movement which means that they were really measuring noise rather than true price movement.
2. In addition, while some researchers have explored using binary classification (Up/Down) most researchers have not explored allowing models to abstain (predict as "flat") from making a prediction.
3. Furthermore the majority of researchers reported high accuracy rates without testing those accuracy rates with any actual trading simulations or profitability analyses.
4. In addition, we found that researchers typically underutilized some of the important features of LOB microstructure, such as Imbalance, Spreads, and Depth Ratios.

Dataset and Data Engineering

Data Source and Structure

Binance, which is one of the biggest crypto exchanges in the world, provides us with high-frequency limit order book (LOB) data for BTC/USDT trading pairs in the form of updated data every 250 milliseconds. This amount of time between LOB updates reflects the fast-paced nature of the cryptocurrency market.

- The content of each raw LOB update from Binance consists of:
- Bid prices and volumes : representing buy-side interest across a range of prices
- Ask prices and volumes : representing sell-side interest across a range of prices
- Timestamp : allowing for accurate ordering of raw order book updates
  Thus, there are 40 raw features for each LOB time slice (10 total price points on 2 sides, each side having 2 features).

Temporal Sequence Construction

Rather than treating each snapshot independently, we construct temporal sequences to provide context. Each training example consists of a window of 20 consecutive LOB snapshots, representing approximately 5 seconds of market history. This windowing approach allows models to learn temporal patterns in order flow dynamics.

Formally, each input sequence is:

X_t = [LOB_{t-19}, LOB_{t-18}, ..., LOB_{t-1}, LOB_t]

Where each LOB_i represents the full 40-feature snapshot at timestep i.

Feature Engineering

We created other features to measure key market behaviour with respect to buyers and sellers. Some of these include:

- Mid-Price: (best_bid + best_ask) / 2, which estimates what is fair for the market at each time point;
- Bid-Ask Spread: The variance between the best ask and best bid, providing an indication of how tight or liquid a market is;
- Order Imbalance: (bid_volume - ask_volume) / (bid_volume + ask_volume) gives an idea of order pressure;
- Depth Ratios: Ratios of volume based upon price point(s) which can be useful for showing market depth in comparison to other price points;
- Weighted Mid-Price: The volume weighted average of both the bid and ask levels, which can give a more precise estimate of what the market is clearing at.

Noise Reduction Through Savitzky-Golay Filtering

In high-frequency financial datasets, there is a significant amount of microstructure noise, which consists of random fluctuations in prices but does not represent a true price trend. To help reduce this noise, we used Savitzky-Golay (SG) filter techniques to smooth the mid-price movements but preserve the underlying price trend.

Savitzky-Golay filtering uses the least squares method to fit a low-order polynomial through successive sets of adjacent data points to eliminate high-frequency noise while preserving the integrity of the financial data signals. This is especially important when using stable and meaningful training labels to train a machine learning model.

Label Engineering: The Critical Innovation

The definition of the label is perhaps one of the most important design decisions in predicting lob. Having the wrong labels generally results in more noise than signal being captured, thus creating a model which memorizes random patterns.

The following two schemes were tested:

<u>Binary Class:</u>
Label = 1 (up) if return at time (t+H) is > return threshold
Label = 0 (down) if return at time (t+H) is < return threshold

<u>Ternary Class:</u>
Label = 2 (up) if return at time (t+H) is > return threshold
Label = 1 (flat) if |return at time (t+H)|<return threshold
Label = 0 (down) if return at time (t+H) is < return threshold

The following return was calculated at horizon (H):
Return = (MidsPrice_at_t=H)-MidsPrice_at_t) / MidsPrice_at_t

After testing various combinations, we were able to achieve the best results from the following:
H = 10 (approx. 2.5 seconds)
Return threshold = 0.0005 (0.05%)

We were able to eliminate all noise and still retain sufficient balance of classes for effective training.

Class Imbalance Handling

Even with optimal horizon and threshold selection, the dataset exhibited significant class imbalance, with the "flat" class dominating. We addressed this through:

1. **Threshold tuning**: Adjusting τ to balance class distribution
2. **Class weighting**: Assigning higher loss weights to minority classes (though this surprisingly degraded performance in our experiments)
3. **Evaluation metrics**: Using ROC-AUC, precision-recall curves, and confusion matrices rather than accuracy alone

Model Architectures and Implementation

Baseline Models

Logistic Regression serves as a simple linear classifier that can create a reasonable interpretative starting point. However, despite the straightforwardness of this technique, it still demonstrated a reasonable level of success and showed that there is a linear predictive relationship present in data sets.

XGBoost represents a robust gradient boosting framework with powerful performance capabilities on structured datasets. It also provided a strong, complex/non-linear baseline model and generated results comparable to those of the complex models and exhibited that features that have been adequately designed and engineered can carry substantial predictive potential, even in relation to models created through the use of advanced techniques such as deep learning.

DeepLOB Architecture

The DeepLOB architecture is our main architecture for LOB forecasts. The convolutional and recurrent parts of the architecture are unique LOB forecast architectures.

Convolutional Layers:

- Purpose: Extract spatial patterns across the 40-feature LOB snapshot
- Configuration: 1D causal convolutions with increasing filter counts ($64 \rightarrow 128$)
- Activation: ReLU for non-linearity
- Batch Normalization: Applied after each convolution for training stability

The convolutional layers capture relationships between different price levels and order book sides, learning patterns like "when bid volume at level 5 increases while ask volume at level 2 decreases."

LSTM Layer:

- Purpose: Model temporal dependencies across the 20-timestep sequence
- Configuration: 128 units with dropout for regularization
- Benefit: Captures how order book states evolve over time, learning patterns like "increasing bid pressure over the past 10 timesteps predicts upward movement"

Dense Prediction Head:

- Dropout Layer: 0.3 rate for regularization against overfitting
- Dense Layer: Maps LSTM output to class probabilities
- Activation: Softmax for multi-class classification

Training Configuration

Optimizer: Adam with initial learning rate of 0.001, chosen for its adaptive learning rate properties and robust performance on noisy gradients.
Loss Function: Categorical cross-entropy with label smoothing (0.1) to prevent overconfidence and improve generalization.
Regularization Techniques:

- Dropout (0.3) in the prediction head
- Batch normalization after convolutional layers
- Early stopping monitoring validation loss with patience of 10 epochs
- Learning rate reduction on plateau

Class Weighting: Initially implemented but removed after experiments showed it degraded validation performance, likely by causing the model to overfit minority class noise.

Alternative Architecture: Transformer

We also experimented with Transformer-based architectures, motivated by their success in capturing long-range dependencies in NLP and other sequential tasks. The Transformer architecture included:

- Multi-head self-attention mechanisms to capture relationships across timesteps
- Positional encodings to maintain temporal ordering information
- Feed-forward networks with residual connections

However, Transformers proved highly sensitive to noisy labels and consistently underperformed DeepLOB in our experiments. The attention mechanisms appeared to overfit to spurious correlations in the noisy financial data, highlighting that model sophistication must match data characteristics.

Experimental Results and Analysis

Binary Classification Performance

The binary DeepLOB model (predicting only Up or Down) achieved strong performance:

- **Accuracy**: 75.8%
- **ROC-AUC**: 0.84
- **Precision-Recall**: Balanced across both classes

Compared to baseline models, metrics have greatly outperformed baseline models. The ROC curve demonstrated an excellent ability to discriminate, as the model maintained a high true positive rate and a low false positive rate across a wide range of threshold levels.
In the case of the P-R curve, the model demonstrated an RR of reasonable quality for high recall levels, indicating that the model is likely able to predict something, rather than producing random guesses.

Ternary Classification Performance

The ternary DeepLOB model (Up, Flat, Down) showed highly variable performance depending on label construction:

- **With optimal parameters (H=10, τ=0.0005)**: 97.4% validation accuracy
- **With suboptimal parameters**: Often below 60% validation accuracy

This dramatic performance variation underscores the critical importance of label engineering. The optimal configuration effectively separated signal from noise, creating learnable patterns.

However, analysis of confusion matrices revealed the model still exhibited some bias toward the majority (flat) class, particularly in ambiguous cases. This bias is inherent to imbalanced data and highlights the importance of confidence filtering for practical deployment.

Horizon and Threshold Sensitivity Analysis

We conducted systematic experiments varying both prediction horizon and return threshold:

Horizon Effects:

- H = 30: Too short, captured mostly noise
- H = 50-60: Optimal range, balanced signal extraction
- H > 100: Too long, patterns became diffuse

Threshold Effects:

- $\tau = 1\times10^{-5}$: Too small, labeled noise as signal
- $\tau = 2\times10^{-4}$: Optimal balance of class distribution and signal quality
- $\tau > 5\times10^{-4}$: Too large, removed too much data

The optimal configuration (H=10 for our 250ms snapshots, equivalent to 2.5 seconds, and τ=0.0005) emerged from this systematic search.

Model Comparison

Comparing across all architectures:

Model	Accuracy	ROC-AUC	Notes
Logistic Regression	~68%	0.72	Strong baseline
XGBoost	~72%	0.78	Best traditional ML
Binary DeepLOB	~76%	0.84	Best overall
Ternary DeepLOB	~58-97%*	N/A	Highly parameter-dependent
Transformer	~65%	0.75	Overfit prone

*Ternary performance varied dramatically with label construction parameters.

Trading Simulation and Practical Validation

Simulation Framework

Achieving high classification accuracy doesn't guarantee trading profitability. To assess real-world viability, we implemented a realistic trading simulator with the following components:

Position Logic:

- **Long position** on "Up" predictions: Buy the asset, profit from price increases
- **Short position** on "Down" predictions: Sell the asset, profit from price decreases
- **Flat position** on "Stationary" predictions: Hold cash, avoid market exposure

Transaction Costs: Incorporated realistic trading fees (0.1% per trade) to account for exchange commissions.
Slippage Modeling: Assumed 0.05% slippage on execution to model market impact and adverse selection.
Initial Capital: Started with $10,000 virtual capital.

Raw Model Performance

Initial trading simulation using raw model predictions (executing every signal without filtering) produced disappointing results:

- Frequent position changes leading to excessive transaction costs
- Many false signals in noisy market conditions
- Equity curve showed high volatility with minimal net gain
- Transaction costs consumed most potential profits

This demonstrated that classification accuracy alone is insufficient—the model generates too many low-quality signals.

Confidence-Based Filtering: The Key Innovation

The breakthrough came from implementing confidence-based trade filtering. Rather than executing every prediction, we only took positions when the model's softmax probability exceeded a confidence threshold.

Implementation:

```
        if max(prediction_probabilities) > 0.6:
            execute_trade(predicted_class)
        else:
    hold_current_position()
```

This simple filter transformed performance completely:

- Reduced trade frequency by approximately 70%, dramatically lowering transaction costs
- Improved signal quality by eliminating low-conviction predictions
- Smoother equity curve with consistent upward trajectory
- Final return: ~12× initial capital over the backtest period

The confidence filter effectively separates high-quality signals (where the model is certain) from ambiguous cases (where the model should abstain).

Equity Curve Analysis

The final equity curve displayed several encouraging characteristics:
Steady Growth: The curve shows consistent upward progression rather than isolated lucky trades.
Acceleration Pattern: Returns accelerate in later periods, suggesting the model performs better in certain market regimes.
Drawdown Control: Maximum drawdown remained reasonable (~15%), indicating acceptable risk management.
Regime Adaptation: The model continued generating profitable signals across varying market conditions.
These characteristics suggest the model has learned genuine predictive patterns rather than overfitting to specific market conditions.

Key Learnings and Insights

Label Engineering is Paramount

The most crucial component of a model's success is the construction of labels. In this example, the difference of 97% accuracy versus randomly selecting labels was entirely due to the determination of the horizon and threshold. A larger lesson to take from this is that defining your prediction target correctly in a noisy domain is more critical than choosing the correct architecture.

In many cases, simpler architectures will outperform more complex ones. Although there has been great success with Transformers in Natural Language Processing applications, the simpler CNN-LSTM architecture consistently provides superior performance to attention-based architectures when trained on a dataset such as DeepLOB. When working with financial data, the extreme level of noise necessitates strong inductive bias (structured sequentially and spatially) and therefore the CNN-LSTM architecture suits this domain more than does the more flexible attention mechanism, which can lead to overfitting to noise.

While it is true that just because a model can classify data accurately does not necessarily mean it will generate a profit when a trading strategy is employed, the inclusion of a confidence filtering mechanism can help connect the two (accuracy and profitability). Through the removal of low-quality signals and reduced transaction costs, along with improved risk-adjusted returns, this demonstrates the value of having a probabilistic output rather than a hard classification when implementing machine-learning models.

Adding class weights to address imbalance has been shown to have the opposite intended effect in this study due to noise within the minority class. In this case, upweighting noise in the minority class rather than learning to detect genuine patterns led to lower classification performance. Conversely, in many instances, using an imbalanced learning approach with clean data may provide a superior classification result than using a balanced approach with noisy data.

Limitations and Future Work

Current Limitations

As mentioned above, our experiments only focused on Bitcoin and we have not investigated whether the results of these experiments can be generalized to other cryptocurrencies or traditional assets.
It is anticipated that our results will perform differently in various market states (bull/bear/or through periods of extreme volatility).
Our simulation utilises a simplistic trading model as we have neglected to include various other factors associated with trading e.g. market impact, position size limitations & realistic order execution capabilities.
Even though we have implemented suitable time-series splitting in our feature engineering, some subtle data leakage is still present.
To build and operate deep learning models, they require a large amount of computational resources and therefore may not be able to be used in latency-critical environments.

Future Research Directions

The improvements made to transformer architectures is examining how, using specific regularization techniques, we can achieve better performance for financial data using careful temporal attention with structural priors. To extend this work to reinforcement learning, we need to move beyond classifying trades and instead learn optimal trading policy by interacting with simulated markets.

The multi-asset framework extends the multi-asset approach to building and analyzing portfolios of many different assets while capturing cross-asset dependencies facilitating the development of diversification strategies.

The development of better calibrated confidence estimates through the use of Bayesian Deep Learning or ensemble methods and handling the uncertainty that arises from the use of models in modelling (uncertainty quantification) allows us to provide better quality trade forecasts.

The integration of new sources of information such as social media sentiment, news events, or on-chain blockchain metrics into trading systems is an area ripe for innovation.

The final step in developing successful methods for successful execution is to create well-defined systems through sophisticated execution strategies, optimising for minimum impact on the market and trade timing.

The real-time production system will require thorough latency optimisation, excellent redundancy and fault tolerance, and monitoring capabilities to ensure that it remains in operation while utilizing advanced production technology.

Conclusion

This study proves that there is an opportunity for developing successful trading systems in very noisy capital markets through this research project using three innovative ideas: (1) Utilizing a Savitzky-Golay filter as a method of reducing noise; (2) Generating appropriate horizon-threshold label engineering techniques and creating appropriate labels for the prediction task; and (3) Developing model confidence to filter trades from the model and create profitable long/short positions from trade predictions.

Research results have shown that with respect to design choice and proper use of design choices, deep learning can yield profitable models that perform extremely well when executing trades based on the model's predictions. For example, with a perfectly tuned model, the model achieved a classification accuracy of 97.4 percent over multiple classifications of trades resulting in a 12-fold return on investment based on backtesting performance.

The findings of this research project have revealed a number of important considerations for the practical use of deep learning models in the field of finance. Some examples of these considerations include:

- A more sophisticated or advanced model is not necessarily going to perform better than a less sophisticated model.
- Transformers do not have to be considered "better" than other types of deep learning models or architectures.
- Profitability from trading cannot simply be determined through classification accuracy.

Therefore, to effectively use deep learning in the financial area, it is important to combine pertinent financial domain knowledge, sound engineering practices for modeling, and a realistic evaluation methodology. Although the model has not yet been validated or tested with respect to different types of

marketplaces or the robustness of the model, and has not yet been developed further with sophisticated execution techniques, this research shows that it is possible to establish a strong foundation for continued prediction model development and research on limit order book pricing. Because the model was systematically designed and evaluated using a well-defined evaluation framework and simulation approach, this research provides a clear set of guidelines for and establishes a model for those looking to conduct their research using deep learning in finance.

The financial markets remain one of the most challenging domains for machine learning due to their noise, non-stationarity, and adversarial nature. This project demonstrates that despite these challenges, carefully designed deep learning systems can uncover predictive patterns and create value—though success requires equal attention to data engineering, model design, and practical deployment considerations.