Deep Learning-Informed Cryptocurrency Trading

Ilan Brauns (ibrauns), Thomas Stanger (tstanger), Chris Jeong (cjeong6), Rosen Iliev (riliev)
CSCI 1470 (Deep Learning), Brown University

Introduction

The cryptocurrency market is famously volatile, with price swings that often defy traditional analysis. Rule-based bots and classic models struggle to adapt to sudden, nonlinear shifts. Motivated by the challenge of making sense of this chaos, we set out to predict Bitcoin's future price within a ±5% margin of error — ambitious, yet practical.

Unlike much prior work focusing on reinforcement learning agents, we pursued a deep learning-based regression approach. Our system exclusively uses historical price data, testing whether powerful architectures like LSTMs and Transformers can uncover hidden patterns in noisy time series.

Methodology

We built a modular framework that trains and compares five different deep learning models:

- **Multilayer Perceptron (MLP)**
- **Convolutional Neural Network (CNN)**
- **Recurrent Neural Network (RNN)**
- **Long Short-Term Memory (LSTM)**
- **Transformer**

Each model receives a sliding window of Bitcoin price history as input and predicts a future price. We normalized price data, filled missing dates, and used consistent time windows to ensure models learned from clean, chronological sequences.

Training relied on the Mean Squared Error (MSE) loss and tracked Mean Absolute Error (MAE) for performance evaluation. Hyperparameters like learning rate, window size, and layer dimensions were carefully tuned through systematic experimentation. There is also a hyperparameter for “prediction distance”, or how far in advance we want to predict the price - this is set to 1440 rows, or 1 day, by default.

We also focused on setting up a clean experiment management system. Each training run automatically logged key metrics, model configurations, and loss curves, enabling structured comparisons. A single main.py file served as the driver script, dynamically handling different models based on command-line arguments.

Data

We sourced historical cryptocurrency data from:

- CoinMetrics Community Dataset
- Kaggle Bitcoin, Ethereum, and Solana Datasets

Data cleaning involved:

- Filling missing timestamps with the previous available value
- Normalizing values
- Aligning features such as Open, High, Low, Close, Volume BTC, and Volume USD
- Using batch sizes of 360 (6 hours in the dataset) for training

Challenges

Building a unified codebase across diverse architectures was a major hurdle. CNNs, RNNs, and Transformers each have unique requirements for input shaping, leading to careful engineering efforts.

Handling missing data in 24/7 markets without natural holidays presented another difficulty. Minor gaps required interpolation strategies to maintain training stability without introducing bias.

Finally, hyperparameter tuning proved tricky — even slight changes in learning rate or sequence length could dramatically affect model performance, requiring extensive experimentation.

Results

Ranking	Model	Final Loss	Mean Absolute Error (USD)	Model Direction Accuracy
1	RNN	0.00042714	1621.413	54.0%
2	CNN	0.00044865	1805.642	53.8%
3	LSTM	0.00045700	1600.595	53.4%
4	MLP	0.00064806	1587.748	53.1%
5	TRANSFORMER	0.00087198	1732.9	53.1%

Figure 1. Model result metrics.

The RNN achieved the best overall performance in terms of loss and directional accuracy, while the MLP achieved the lowest absolute error — a surprising result considering its simpler architecture.

Discussion

Overall, deep learning shows real promise for cryptocurrency price prediction, especially when the focus is on probabilistic forecasting rather than exact numbers. Building a unified codebase early paid off, as it allowed us to easily test different models.

It surprised us how competitive the simple MLP was compared to more complex models like LSTMs and Transformers. This highlights that, for relatively structured time series like Bitcoin prices, simpler models can sometimes be more effective and efficient.

This project reaffirmed that no single architecture "wins" universally. While deep learning excels at modeling complex, nonlinear patterns, even simpler models like MLPs can compete when data is structured carefully.

Future work could include:

- Incorporating sentiment analysis (news, tweets) alongside historical prices
- Expanding to multi-asset trading strategies
- Fine-tuning Transformer-based models for even longer time dependencies
- Testing different feature sets (e.g., volume, market cap) beyond price

Github Repo