# CryptexLLM: How LLM Generalizability Forecasts High Volatility

Mary Tsaryk[1], Logan Rao[2], Anwar Benhnini[2] (advisor), Ioan Raicu[2] (advisor)

[1] Department of Computer and Information Science, Fordham University, [2] Department of Computer Science, Illinois Institute of Technology

## Introduction

- Many time series datasets show predictable seasonality, but **high-volatility data** remains **difficult to forecast** due to regime shifts and noise that erode accuracy.
- **Traditional methods** struggle to generalize across different datasets or changing market conditions, limiting their usefulness in real-world deployment.
- **Our approach:** Adapt large language models (LLMs) for time-series forecasting to **leverage pre-training knowledge** and improve robustness on volatile data.
- Our case study is **Bitcoin** (BTC) – selected because:
  - **(1)** It exhibits extreme volatility over a decade.
  - **(2)** Direction matters more than exact price for trading.
  - **(3)** It trades 24/7/365, providing abundant data.
  - **(4)** Plentiful sentiment signals enable multimodal analysis.

## Adaptive Loss

**Mean Squared Error:**
$$\mathrm{MSE} = \frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - y_t)^2$$

**Mean Absolute Error:**
$$\mathrm{MAE} = \frac{1}{T}\sum_{t=1}^{T}|\hat{y}_t - y_t|$$

**Mean Absolute Percentage Error:**
$$\mathrm{MAPE} = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{\hat{y}_t - y_t}{\max(|y_t|, \varepsilon)}\right|$$

**Mean Absolute Directional Loss:**
$$\mathrm{MADL} = \mathbb{E}\big[-\operatorname{sign}(y_t\hat{y}_t)\cdot|y_t|\big]$$

$y_t$ – True (actual) value at time step t
$\hat{y}_t$ – Predicted value at time step t
$N$ – Number of samples or series
$T$ – Number of time steps

**Sharpe ratio** (of predicted returns)
$$r_t = \hat{y}_t - \hat{y}_{t-1}, \quad \mathrm{Sharpe} = \frac{\mathbb{E}[r_t]}{\sqrt{\mathrm{Var}(r_t)} + \varepsilon}$$

**Generalized Mean Absolute Directional Loss**
$$\mathrm{GMADL} = \mathbb{E}\left[-\left(\sigma(a\, y_t\hat{y}_t) - \tfrac{1}{2}\right)\cdot|y_t|^b\right], \quad \sigma(z) = \frac{1}{1+e^{-z}}$$
with parameters a > 0, b > 0;
controls steepness/weighting of sign errors

**Directional Loss Function**
$$d_t = \operatorname{sign}(\hat{y}_t - \hat{y}_{t-1}) \neq \operatorname{sign}(y_t - y_{t-1})$$
$$\mathrm{DirLoss} = \frac{1}{T-1}\sum_{t=2}^{T}\mathbf{1}[d_t]$$
$$\mathrm{DLF} = \lambda\cdot\mathrm{DirLoss} + (1-\lambda)\cdot\mathrm{MAE}$$

**Sentiment Preprocessing**
$$\bar{s}_t = \frac{1}{k}\sum_{i=0}^{k-1} s_{t-i}, \quad k \in [3, 7]$$
$$x_t = \bar{s}_{t-\ell}, \quad \ell \in \{0, 1, 2\}$$
**Smooth** daily sentiment over 3–7 days, then use a **lagged value** so each day's signal only uses information available before the decision time (avoid leakage). Choose $\ell = 0$ with a strict daily cutoff; otherwise $\ell = 1 - 2$. Tune $k$, $\ell$ by directional accuracy and backtest return.

**Combined Loss**
$$\mathrm{Combined} = \left(\sum_i w_i \cdot \mathcal{L}_i\right) \times \texttt{scale\_factor}$$
Combined loss is a **weighted sum** of whichever losses you choose, then multiplied by a constant scale_factor (default **0.1**) for numerical stability.

## CryptexLLM Pipeline

### Architecture
- **TimeLLM** enables efficient training by keeping the LLM frozen and training only small adapters.

### Data & Features
- Multi-granular **OHLCV** candlesticks + **financial indicators** (momentum, volatility, behavior).
- **Correlation-based** indicator selection to reduce redundancy; add market sentiment via APIs.

### Training Setup
- **Models:** Llama 7B · Llama 3.1 8B · DeepSeek R1 7B · Mistral 7B · Gemma 3 1B · Qwen 3 7B.
- **Infrastructure:** 16 NVIDIA Tesla V100s (32 GB) across 4 nodes.
- **Tech stack:** PyTorch · MLflow · Optuna · MinIO · PostgreSQL.

### Evaluation
- In **backtesting**, we buy when the model predicts ↑, sell when ↓.

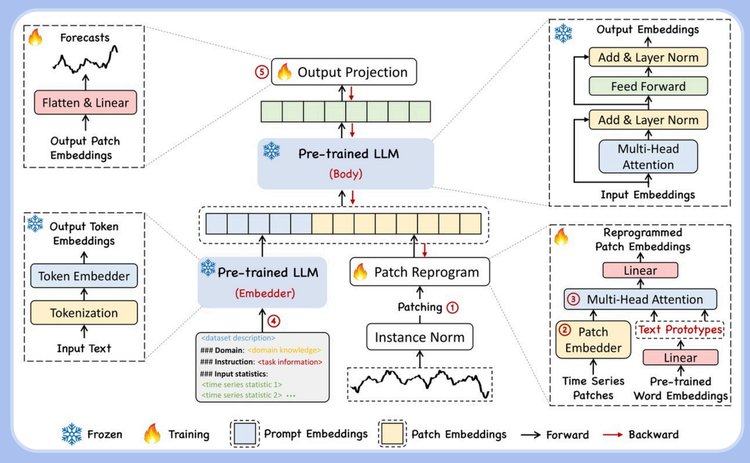


**Figure 1** - TimeLLM Model Architecture

### Optimization
We use **Walk Forward Optimization:** repeatedly training a model on a rolling "in-sample" period of historical data, then testing it on the immediately following "out-of-sample" period.

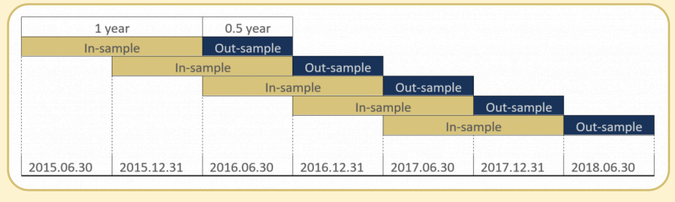


**Figure 4** - Walk Forward Optimization

## Backtesting

### Strategy
We simulate a simple trading strategy: buy when the model predicts ↑, sell when ↓.

### Metrics
Directional Accuracy (DA), Total Returns, Cumulative Equity Curve, Max Drawdown, Error Metrics (MAE/MSE), and Sharpe of predicted returns.

### Notes
The same rules, thresholds, and data splits are applied to every model for a fair comparison. Signals are generated at ***t*−1** and applied at ***t*** to avoid look-ahead/ leakage.

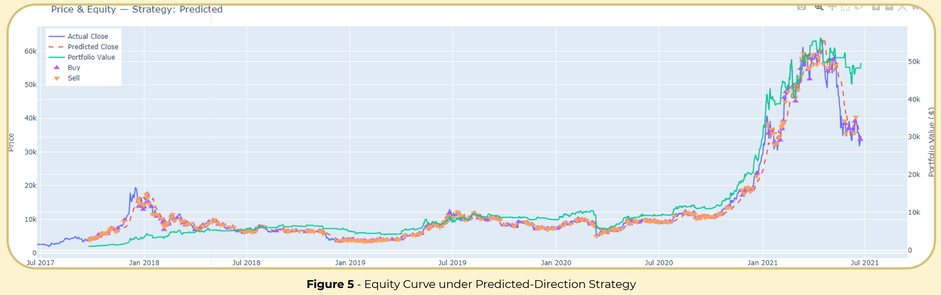


**Figure 5** - Equity Curve under Predicted-Direction Strategy

## Feature & Sentiment Study

**Goal**
Enrich raw OHLCV with informative signals.

**Features & sources**
Momentum (RSI, SMA), Volatility (Bollinger Bands, GARCH), Microstructure/behavior (VWAP, lags), and market sentiment (Reddit, Fear & Greed, news). Sentiment is smoothed and optionally lagged (0–2 days) to respect data cutoffs.

**Correlation-based feature selection**
1. **Compute the correlation** of each candidate feature with the target (return/direction).
2. **Rank features;** remove redundancies via inter-feature correlation pruning; keep the top set.
3. **Standardize** selected features; join with sentiment features for training.

**Effect:** preserves accuracy while cutting training time by ~70% versus using all indicators.

Website

GitHub

## Results

### Model performance
With correlation-selected indicators plus sentiment, **Llama-3.1-8B** yields the strongest backtest; Qwen-3, DeepSeek R1, and Mistral-7B are close. Gemma-3-1B is fastest but weaker, pointing to the importance of capacity (see Fig. 5).

### Interpretation
The weaker results from 1B models indicate that **capacity** matters. The 100–170 **input length** reflects a balance between capturing pattern context and avoiding stale information. **Direction-aware** training aligns optimization with trading outcomes.

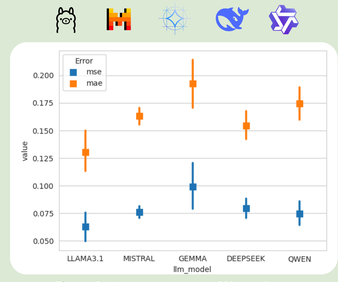


**Figure 2** - Average error and 68% confidence interval per LLM model

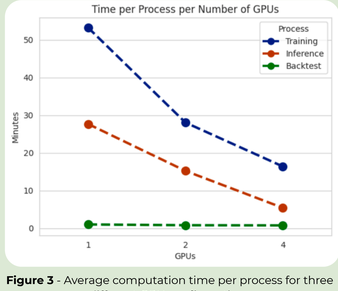


**Figure 3** - Average computation time per process for three different GPU configurations

### Computational efficiency
Training time scales roughly with parameter count and feature load; 1–4 GPUs deliver near-linear training speed-up, and inference scales near-perfectly (r ≈ −0.97). Backtesting gains little from more GPUs because it is already cheap.

### Impact of engineered features
Indicators improve accuracy but add cost; correlation-based feature selection retains the most predictive indicators and cuts training time by ~70% with minimal loss.

## References

**[1]** M. Jin et al., "Time-LLM: Time series forecasting by reprogramming large language models," arXiv:2310.01728, ICLR 2024.
**[2]** J. Michańków, P. Sakowski, R. Ślepaczuk, "Mean absolute directional loss as a new loss function for machine learning problems in algorithmic investment strategies," arXiv:2309.10546, 2023.

## Acknowledgement

Supported in part by NSF OAC-2150500. Many thanks to André Bauer, Assistant Professor of Computer Science at Illinois Institute of Technology (IIT), for his help during brainstorming and editing.