September 2, 2025

TSL LSTM White Paper

Cover Page

LSTM-Based Prediction and Trading System: Architecture, Equations, and Results

Prepared by **Trading System Lab®**
with assistance from ChatGPT

Date: August 31, 2025

Keywords: LSTM, Trading, Machine Learning, Adam Optimizer, Equity Curve, Non-Stationary Time Series

---

About Trading System Lab®

Trading System Lab® (TSL) is an advanced research and development platform for designing, testing, and deploying algorithmic trading strategies. Built with proprietary genetic programming, machine learning, and AI-driven optimization, TSL delivers cutting-edge tools for traders, quants, and institutions seeking automated, adaptive, and high-performance trading solutions.

Abstract

This report presents the design and implementation of a custom Long Short-Term Memory (LSTM) neural network for financial market prediction, developed entirely in open-source C code without reliance on closed libraries or DLLs. The system integrates directly with a trading simulator, producing both predictive outputs and equity curve performance metrics across training, validation, and out-of-sample datasets. By combining transparency in design with modern optimization techniques, the project demonstrates the potential of LSTM models to capture sequential market dependencies while also highlighting their limitations in handling non-stationary time series.

Executive Summary

This project delivers a custom Long Short-Term Memory (LSTM) neural network built entirely in open C code, without reliance on proprietary DLLs or closed-source machine learning libraries. It is designed for financial forecasting and integrates directly with a trading simulator. The system produces reliable trading signals and equity curves across training, validation, and out-of-sample data.

Key Features:

- Open implementation in 100% C code.
- Predictive outputs with one output for trading signal, another constrained non-negative.
- Integration with trading simulation producing equity curves.
- Optimization of output layer weights using Adam.

Limitations:

- Markets are non-stationary, requiring retraining.
- Regime shifts and volatility spikes may reduce predictive power.
- Current build trains only the output layer; deeper optimization possible.

Table of Contents

1. Introduction
2. System Architecture
3. Canonical LSTM Equations
4. Adam Optimizer
5. Data Pipeline
6. Trading Simulation
7. Results
8. Advantages of LSTM
9. Limitations in Non-Stationary Series
10. Conclusion & Future Work

---

1. Introduction

Financial markets are highly dynamic and exhibit non-stationary behavior, making prediction challenging. This project explores the use of Long Short-Term Memory (LSTM) networks for financial forecasting and trading. By leveraging sequence modeling, LSTMs aim to capture temporal dependencies and generate actionable trading signals.

---

2. System Architecture

The system follows a structured pipeline: input data is read from CSV, transformed into features, passed through an LSTM network, and mapped to two outputs. These outputs are then evaluated by a trading simulator, which produces an equity curve as a measure of strategy performance.

Figure 1: System Pipeline

---

3. Canonical LSTM Equations

At each time step *t*, with input **x_t**, hidden state **h_{t-1}**, and cell state **c_{t-1}**, the LSTM computes:

$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$ (Forget gate)

$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$ (Input gate)

$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$ (Output gate)

$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$ (Candidate cell state)

$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$ (Cell state update)

$h_t = o_t \circ \tanh(c_t)$ (Hidden state update)

$y_t = W_y h_t + b_y$ (Output layer)

Glossary of Parameters

- **x_t**: Input vector at time step *t*
- **h_{t-1}**: Previous hidden state
- **c_{t-1}**: Previous cell state

- **f_t**: Forget gate activation
- **i_t**: Input gate activation
- **o_t**: Output gate activation
- **c̃_t**: Candidate cell state
- **c_t**: Updated cell state
- **h_t**: Updated hidden state
- **W_f, W_i, W_o, W_c**: Weight matrices for input connections
- **U_f, U_i, U_o, U_c**: Weight matrices for recurrent connections
- **b_f, b_i, b_o, b_c**: Bias vectors
- **W_y**: Output weight matrix
- **b_y**: Output bias vector

---

4. Adam Optimizer

The Adam optimizer is used to update model parameters based on the gradient of the loss function. It maintains running estimates of first and second moments of gradients.

$$m_t = \beta1 * m_{t-1} + (1 - \beta1) * g_t$$

$$v_t = \beta2 * v_{t-1} + (1 - \beta2) * g_t^2$$

$$\hat{m}_t = m_t / (1 - \beta1^t)$$

$$\hat{v}_t = v_t / (1 - \beta2^t)$$

$$\theta_t = \theta_{t-1} - \alpha * \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$$

---

5. Data Pipeline

Market data is read from CSV files into **PriceData** structures. Each row provides date, time, OHLC values, derived features, and two target outputs. These are used for training, validation, and out-of-sample testing.

---

6. Trading Simulation

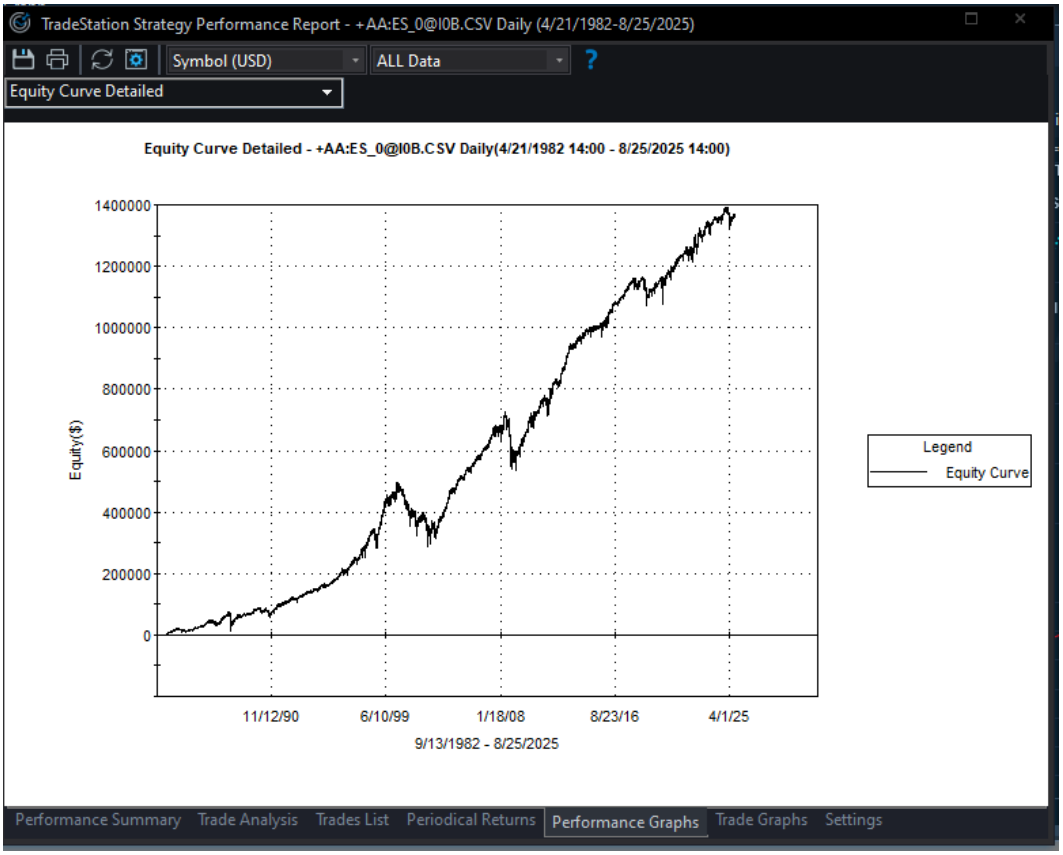
Trading logic is implemented to evaluate signals from the LSTM outputs. A long position is entered when the signal crosses above zero, and exited when it crosses below zero or at session close. Profit per trade is recorded and accumulated into an equity curve. A Custom Fitness Function (CFF) evaluates profitability and risk-adjusted returns.

---

7. Results

The following figure shows the equity curve generated by the LSTM trading system.

Figure 3: Equity Curve Generated by the LSTM System



---

8. Advantages of LSTM

- Ability to capture temporal dependencies in sequential data.
- Gating mechanisms prevent vanishing gradients.
- Flexibility to model complex market dynamics.

- Transparent open C implementation without closed libraries.

---

9. Limitations in Non-Stationary Series

- Performance may degrade during regime shifts.
- Requires retraining to adapt to non-stationary market conditions.
- Sensitive to hyperparameter selection.
- May underperform simpler models in highly noisy markets.

---

10. Conclusion & Future Work

The project demonstrates the feasibility of using LSTM networks in trading systems with a transparent C-based implementation. While results are promising, limitations in non-stationary environments require ongoing retraining. Future work includes exploring Temporal Convolutional Networks (TCNs) and Transformer-based models for enhanced robustness.

About Trading System Lab®

Trading System Lab® (TSL) is an advanced research and development platform for designing, testing, and deploying algorithmic trading strategies. Built with proprietary genetic programming, machine learning, and AI-driven optimization, TSL delivers cutting-edge tools for traders, quants, and institutions seeking automated, adaptive, and high-performance trading solutions.

This project and report were created with assistance from ChatGPT.