Stock Price Direction Prediction Based on Deep Learning

DATS 6303 Deep Learning Final Individual Project Report

Luhuan Wang

Introduction

This report documents my contribution to the stock price direction prediction project in the group. The goal of the project was to develop a deep learning model to predict whether the stock price will rise or fall at the end of the next year based on financial reports and adjusted closing prices. The team implemented multiple models, including LSTM, RNN, CNN-LSTM, and LSTM with attention, to compare their effectiveness.

In this report, I will focus on my personal contributions, including data preprocessing, feature engineering, training LSTM models, and implementing advanced model architectures such as LSTM with attention.

Personal work description

Algorithm background:

mainly work on LSTM, which is a recurrent neural network designed to solve the gradient vanishing problem in training sequences. It can retain information for longer periods of time and is ideal for time series data.

I also contributed the LSTM model with attention mechanism, which uses the attention mechanism to allow the model to focus on specific time steps in the sequence, improving performance by weighting more critical inputs.

Algorithm Details:

LSTM Equations

$$1.\ ForgetGate:\ \left[f_t = \sigma\left(W_f \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_f\right)\right]$$

$$2.\ InputGate[i_t = \sigma(W_i \cdot [\begin{matrix} h_{t-1} \\ x_t \end{matrix}] + b_i)]$$

$$3.\ CandidateCellState):\ [\tilde{C}_t = tanh(\ + b_C)]$$

$$4.\ (OutputGate):\ [o_t = \sigma\left(W_o \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_o\right)]$$

$$5.\ (CellStateUpdate):\ [C_t = f_t * C_{t-1} + i_t * \tilde{C}_t]$$

$$6.\ (HiddenState):\ [h_t = o_t * tanh(\tilde{C}_t)$$

Attention Mechanism

$$AttentionWeights:\ [a_t = softmax(e_t),$$

$$e_t = W_a \cdot h_t]$$

$$ContextVector:\ [c = \sum_t a_t * h_t]$$

These equations drive the core computations in the LSTM and attention mechanisms.

Personal Contributions

Implemented a sequence generation function to create training samples for LSTM and other models:

```
def create_sequences(data, seq_length):
    xs, ys = [], []
    for i in range(len(data) - seq_length):
        x = data.iloc[i:(i + seq_length)][features].values
        y = data.iloc[i + seq_length]['price_direction']
        xs.append(x)
        ys.append(y)
    return np.array(xs), np.array(ys)
```

Model Implementation

I trained a baseline LSTM model with 128 units and achieved 64% accuracy on the test set. I implemented LSTM with Attention, which enhanced the model's ability to focus on key time steps and achieved 65% accuracy.

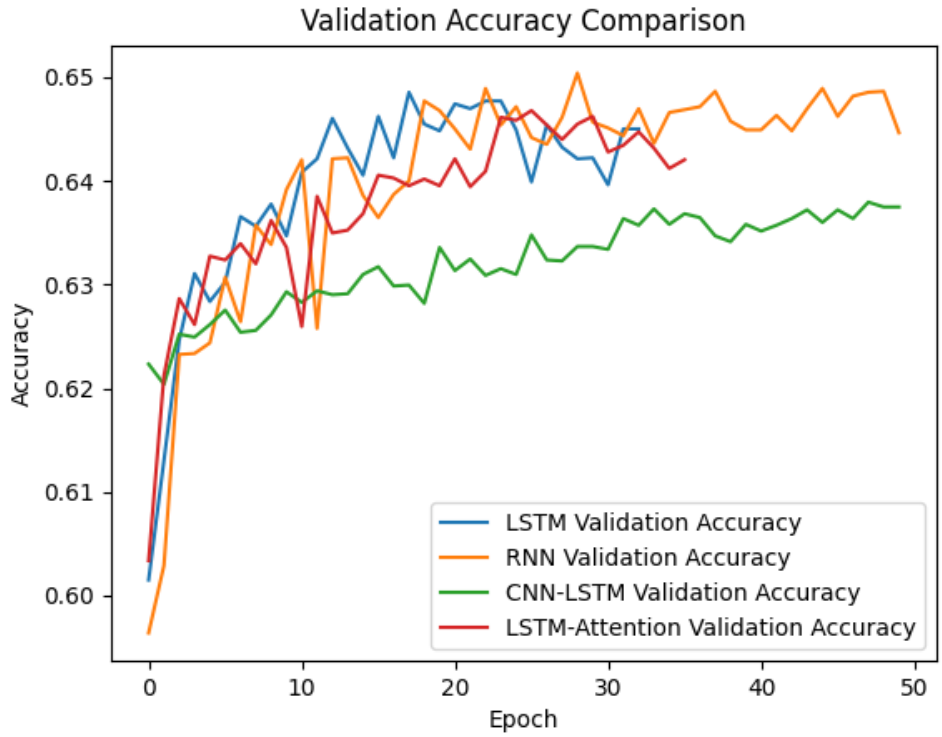
Attention Layer implementation

```
class Attention(Layer):
    def build(self, input_shape):
        self.W = self.add_weight(name='attention_weight', shape=(input_shape[-1], 1),
                                 initializer='random_normal', trainable=True)
        self.b = self.add_weight(name='attention_bias', shape=(1,),
                                 initializer='zeros', trainable=True)
    def call(self, x):
        e = K.tanh(K.dot(x, self.W) + self.b)
        a = K.softmax(e, axis=1)
        output = x * a
        return K.sum(output, axis=1)
```

Results

I found that LSTM with Attention performed slightly better because it focused on the key time steps. CNN-LSTM performed poorly, probably because of the limited local feature patterns in

the dataset.




```
Final Model Accuracies:
LSTM: 0.6477
RNN: 0.6485
CNN-LSTM: 0.6375
LSTM-Attention: 0.6468
```

Main Findings

LSTM and LSTM with attention mechanism are the most effective models for predicting stock price trends.

Attention mechanism increases interpretability and slightly improves accuracy.

In this group assignment, I learned that proper preprocessing and feature selection are crucial for deep learning performance. Advanced architectures like Attention can provide marginal

gains but require more computational resources.

Future Work

I may try increasing the dataset size and including additional features. I may also try experimenting with the Transformer model to better capture long-term dependencies.

Calculate the percentage of the code that you found or copied:

I mainly participated in model design, training and code optimization.

Copied Percentage= (40/90) ×100≈44.44%

References

https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714

https://www.researchgate.net/figure/Architecture-of-the-Hybrid-1D-CNN-LSTM-model-for-human-activity-recognition_fig4_343341551

https://medium.com/@poudelsushmita878/recurrent-neural-network-rnn-architecture-explained-1d69560541ef

https://www.researchgate.net/figure/Flow-chart-of-CNN-LSTM-Attention-model_fig3_363533496