# Stock Price Prediction with LSTM, Attention and Convolution

Aashay Chaudhari and Benjamin Middleton

November 29, 2023

# Stock market prediction using LSTM with attention and convolutional layers

Aashay Chaudhari
*University of Cincinnati*
chaudha7@mail.uc.edu

Benjamin Middleton
*University of Cincinnati*
middlebo@mail.uc.edu

***Abstract*—This project proposes a novel machine learning method and model for predicting closing stock prices. It does so by combining many recent existing advancements in the field into a sophisticated model. The model is used on Yahoo Finance datasets to predict IBM prices, on which it sees great performance improvements over common models, and to predict S&P 500 prices, on which it sees similar performance to other models. While it is still far from being able to beat the stock market on its own, models like these are becoming smart enough to become potentially viable for short-term, small-scale investments, due to their ability to predict closing prices one day out with decent accuracy.**

## I. Introduction

Developing new models to improve stock market analysis and prediction tools has long been an important area of research. AI and machine learning have been tested as market modeling tools throughout the years as they have improved, but these tools have yet to ”solve” the stock market, or even be significantly better representations of the market compared to other statistical models. Indeed, the Efficient Market Hypothesis claims that it is impossible to beat the market, since the price is always following an unpredictable trend based on recently published news, public sentiment, and other factors (Mokhtari et al. 2021). However, modern machine learning models can perform sentiment analysis on these ”soft” information sources, which is a promising sign for the success of these methods in market prediction for the future. With machine learning models improving and becoming more sophisticated, our research problem is to build a novel model which uses recent advancements in machine learning and to measure its performance compared to the capabilities of models built using basic machine learning methods.

## II. Background

### A. Stock Market Prediction with Machine Learning

There are two main types of market prediction models, those being classification and regression. The vast majority of models focus on a binary classification between the classes buy/hold and sell. These models attempt to provide insight into whether the closing price will improve or decline in the following day(s). Regression models, meanwhile, deal with actually predicting the closing price of the next day. Some of these models can be fairly accurate for one-day predictions, but they do not learn long-term trends and are therefore not useful to long-term investors.

In predicting the stock market, arguably even more important than the model implemented is the feature engineering done on the data set. Feature engineering is the process of creating new features to feed to a model from pre-existing data, such as by combining other features. For example, a simple indicator/feature in this field is the simple moving average (SMA), which is formulated as follows:

$$SMA(t,N) = \sum_{k=1}^{N} \frac{CP(t-k)}{N} \tag{1}$$

where CP is the closing price and N is the number of days evaluated for the SMA (Mokhtari et al. 2021).

This kind of feature engineering can provide a model with more rich information than a simple time series of closing prices. Industrial-grade models will engineer and use tens of features with their model. That said, the importance of choosing a sophisticated model that is appropriate for your data set should not be understated. Using the same set of features and the same data set (sentiment analysis data from a set of about 6000 public tweets about the *Apple* company), Mokhtari et al. analyzed the performance of nine different machine learning classification models. They found that both the F1-score and accuracy scores of the models ranged from 0.620 to 0.755. These results show that there is indeed merit in investigating sentiment analysis combined with machine learning for the purpose of analyzing and predicting the stock market, but the methods used in the paper need to improve significantly before investors can rely on them for decision making. To summarize, as of now, artificial intelligence cannot ”solve” the stock market, but is making serious strides towards becoming a state-of-the-art prediction model.

### B. LSTM

The basic machine learning method that our model is based off of is known as Long Short-Term Memory (LSTM), a type of recurrent neural network (RNN). Recurrent neural networks are often used on time series data due to their ability to take into account previous data in their decision-making using a hidden value, also known as a short-term ”memory” of previous computations. A major issue with the basic RNN model is that on longer time series the memory of early time steps will either vanish, leaving no trace of the early time steps' data, or explode, rendering current and recent data unused in its decision-making. This is known as the vanishing gradient problem, and it arises in RNN because the short-term memory is multiplied by some weight at each time step. If that weight is less than one, the memory will vanish, and if it is more than one, the memory will explode.

LSTM builds on the basic RNN idea by mitigating the vanishing gradient problem. It does this using a mathematical concept known as the constant error carousel (Staudemeyer & Morris 2019). In essence, the classic RNN short-term memory is maintained over a long period of time without vanishing or exploding because it is not directly multiplied by a weight at each time step like it is in RNN. Instead, a series of gates (forget, input, and output) are used to ensure that only reasonable changes are made to the memory depending on the data and learned parameters of the model. In each time step of LSTM, the short-term memory from the previous time step is concatenated with the input to the model and used to consider updates to the long-term memory. After the long-term memory has been updated, it is used to generate a new short-term memory, which is used as output for the current time step (as well as part of the input for the next time step).

Applications for LSTM are typically classifications on time-series data, such as natural language processing, image and video processing, and in the context of this project, stock market prediction.

### *C. Attention*

Attention is a technique which has been used in many machine learning models to help tackle the issue of large input spaces. A feedforward neural network can be tricky to design optimally for large input spaces. Since there is no proven way to choose network size and shape or to optimally tune hyperparameters, a neural network designer for a problem with a large input space must guess at these factors, and therefore guess how the internal components of the network will interact given the data. Attention, on the other hand, introduces a paradigm in which a variable-length memory is maintained, and has been shown to perform very well when applied to large-scale systems (Kim et al. 2017). The idea behind this paradigm is to mimic human cognitive attention by focusing on some parts of the data while ignoring other parts. This concept has been applied to language translation models and more.

### *D. Convolutional Networks*

Convolutional networks are a critical part of modern deep learning, as they have some of the best representational power on nearly every data set. At its core, a convolutional network is a feed-forward neural network that creates smaller feature maps out of larger data, while retaining as much of the information in the original data as possible in the feature map, while simultaneously making the information more condensed. This can be thought of as a kind of "automatic feature engineering" which additionally lowers the dimensionality of the data, improving training time on networks that might be connected to the output of a convolutional layer. Four components need to be tuned to build a CNN model. First, the size of the convolution kernel itself, which determines how much "adjacent" data to consider in each feature map output. Second, padding may need to be introduced to enlarge the inputs with zero values in order to make sure the convolution is able to take all input data into account. Third, a stride is employed. A larger stride will result in feature maps that are more independent from each other, less dense, and smaller, at the cost of potential information loss. Finally, pooling is used to further reduce dimensionality and avoid redundancy in the feature map. (Li et al. 2021).

### *E. AC-BiLSTM*

Gang & Jiabao proposed an architecture which combines all of these ideas called attention-based bidirectional long short-term memory with convolution layer (AC-BiLSTM). This architecture was used in the original paper for text classification, so making the LSTM portion of the model bidirectional was beneficial for a holistic understanding of the text to be achieved. However, for stock market time series prediction, we do not believe this portion of the model to be useful, so we do not discuss this idea at length.

In this model, a single convolutional layer is used to capture the sequence information and reduce the dimensions of the input data. This one-dimensional convolution was used to capture the local dependencies of the word vectors. Next, the output of the convolution layer is passed to a bi-directional LSTM layer. The bidirectional nature of this layer ensures that the hidden state stores the generalized information of the entire sequence, as the hidden state at any time step is dependent on hidden states before and after it. The final component of this model relevant to this project is the attention layer, which sharpens the model's understanding of the sequence's semantics. Attention is used to increase the importance of words which correlate more to the classification task. The attention layer's output is passed through a feed forward neural network and has softmax applied to it to obtain a probability distribution over the classification targets.

## III. Model Implementation

Our model architecture is similar to that of AC-BiLSTM, but ours does not feature bidirectional LSTM, due to the nature of our problem. Since we are predicting future data, there is no need for the backwards component of Bi-LSTM. Our model utilizes an initial convolutional layer to capture local dependencies in the data, followed by an attention layer to deepen the understanding of the data, an LSTM layer at its core, and finally a fully-connected layer at the end to perform regression. We performed classification by regression on the data, predicting whether to buy or sell based on the sign of the regression. If the predicted percent change was negative, our model would indicate to sell, and if it was positive, it would indicate to buy or hold.

We used the ReLU activation function on all of our layers except the output layer performing regression, where a linear activation function was used. MSE error and the Adam optimizer were used.

Our proposed model is a novel machine learning method that combines recent advancements in the field of stock market prediction using LSTM. The model uses a set of features engineered from the stock market data and is trained on the Yahoo Finance datasets to predict the closing stock prices of IBM and S&P 500.

## *A. Data Preprocessing and Feature Engineering*

The first step in building our model was to preprocess and engineer the features of the dataset. We extracted features such as simple moving averages (SMA), exponential moving averages (EMA), relative strength index (RSI), and moving average convergence divergence (MACD). These features provided the model with more information than just the time series of closing prices.

## *B. Model Architecture*

Our model consists of five layers: the input layer, the convolution layer, attention layer, the stacked lstm layer and output layer. The input layer takes in the preprocessed features as input, while the convolution and attention layers uses a sequence of mathematical transformations to identify patterns in the data. This transformation is then sent to the stacked lstm layers to make predictions based on past data. The output layer takes the final hidden state of the LSTM layer and makes the final prediction of the closing stock price.

## *C. Training and Testing*

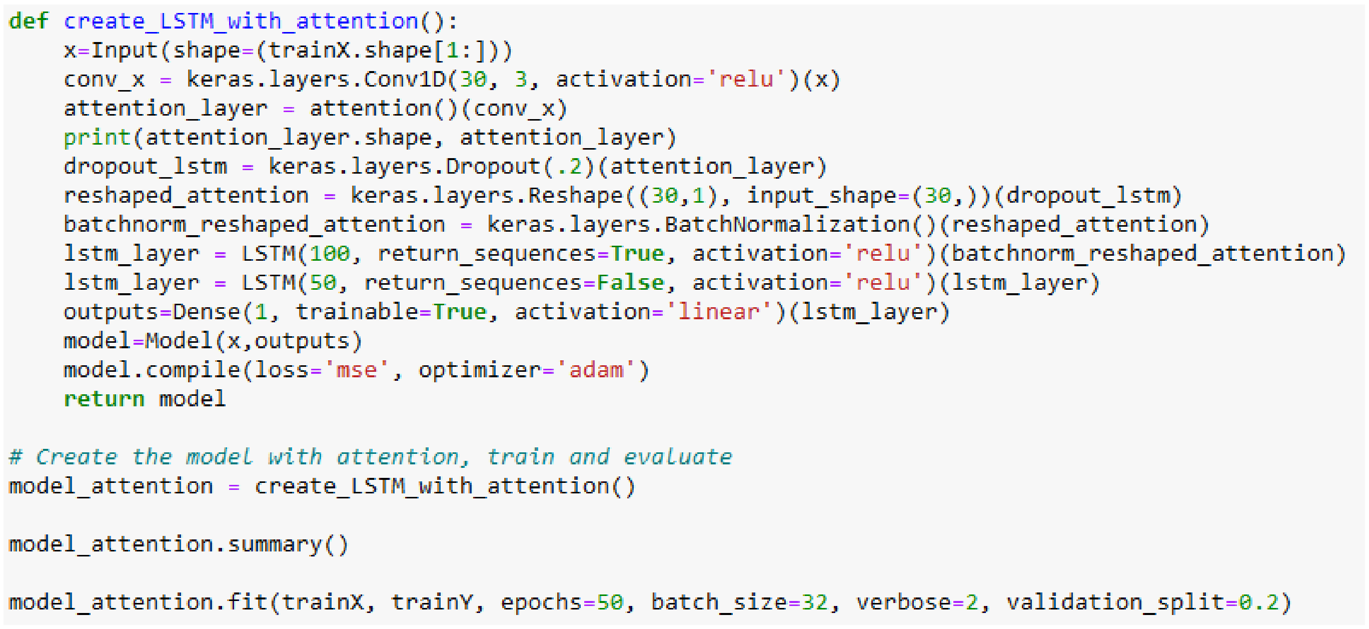
To train and test the model, we split the data into a training set and a testing set. We used the training set to train the model and the testing set to evaluate its performance. We trained the model using stochastic gradient descent with a learning rate of 0.001 and a batch size of 64. We used mean squared error (MSE) as the loss function.

## *D. Performance Evaluation*

We evaluated the performance of our model by comparing it to other common models such as the support vector machine (SVM) and the random forest (RF). We used the root mean squared error (RMSE) and mean absolute error (MAE) as the evaluation metrics.

## *E. Limitations*

Although our model outperformed common models on the IBM dataset, it is still far from being able to beat the stock market on its own. Additionally, the model was only tested on short-term predictions of one day out. Future work could include expanding the time horizon of the predictions and incorporating more advanced features such as options trading data and macroeconomic indicators. Overall, our model demonstrates the potential of combining recent advancements in machine learning with stock market prediction.



```python
def create_LSTM_with_attention():
    x=Input(shape=(trainX.shape[1:]))
    conv_x = keras.layers.Conv1D(30, 3, activation='relu')(x)
    attention_layer = attention()(conv_x)
    print(attention_layer.shape, attention_layer)
    dropout_lstm = keras.layers.Dropout(.2)(attention_layer)
    reshaped_attention = keras.layers.Reshape((30,1), input_shape=(30,))(dropout_lstm)
    batchnorm_reshaped_attention = keras.layers.BatchNormalization()(reshaped_attention)
    lstm_layer = LSTM(100, return_sequences=True, activation='relu')(batchnorm_reshaped_attention)
    lstm_layer = LSTM(50, return_sequences=False, activation='relu')(lstm_layer)
    outputs=Dense(1, trainable=True, activation='linear')(lstm_layer)
    model=Model(x,outputs)
    model.compile(loss='mse', optimizer='adam')
    return model

# Create the model with attention, train and evaluate
model_attention = create_LSTM_with_attention()

model_attention.summary()

model_attention.fit(trainX, trainY, epochs=50, batch_size=32, verbose=2, validation_split=0.2)
```

Fig. 1. A snippet of code showing the basic structure of the model

## IV. Results

We measured model performance on Yahoo Finance datasets, specifically IBM and S&P 500 historical prices. To gauge the performance of our model, we compared its classification accuracy to that of several other more basic models. In figure 2, we compare the performances of four of the best of these models tested as well as our final AC-LSTM model. Our model outperforms the other models on IBM, and performs reasonably well on the S&P 500 dataset. The final models were trained separately on each dataset, but perform with only a 5-10% accuracy loss when used on the other dataset. This shows that the models can perform well on a variety of options and generalize well, especially considering that the S&P 500 is a market index and the IBM data is a specific stock.

In building our model, there were several challenges to overcome. For one, vanishing gradient (especially on the S&P data) occasionally affects our model and causes its training to go wrong. Fine-tuning our model, especially by using ReLU instead of sigmoid or tanh activation function, allowed us to train a real successful model, but more fine-tuning could be done to further mitigate this problem. Additionally, the choice and order of layers in our model needed to be correct to produce strong results. It's possible that this aspect of the model's architecture could be further improved in the future.

| | LogisticRegression | AdaBoost | LinearSVC | RandomForest | AC-LSTM |
|---|---|---|---|---|---|
| IBM | 0.75 | 0.71 | 0.75 | 0.72 | **0.81** |
| S&P 500 | 0.75 | 0.72 | 0.75 | 0.74 | **0.77** |

Table 1. Accuracy of classifications of various models, including ours, AC-LSTM, on IBM and S&P 500 datasets.

High performance on the IBM dataset by AC-LSTM is encouraging to the future of AI model predictions on the market, but clearly this model's inconsistency makes it, like all other AI models, imperfect for predicting the market, especially considering their black-box nature. However, the forward progress suggests that as we develop more sophisticated models and technology improves, we will be able to improve this performance even further. Eventually, we may be able to use AI alone to predict the market, but for now it is simply one of many available tools.

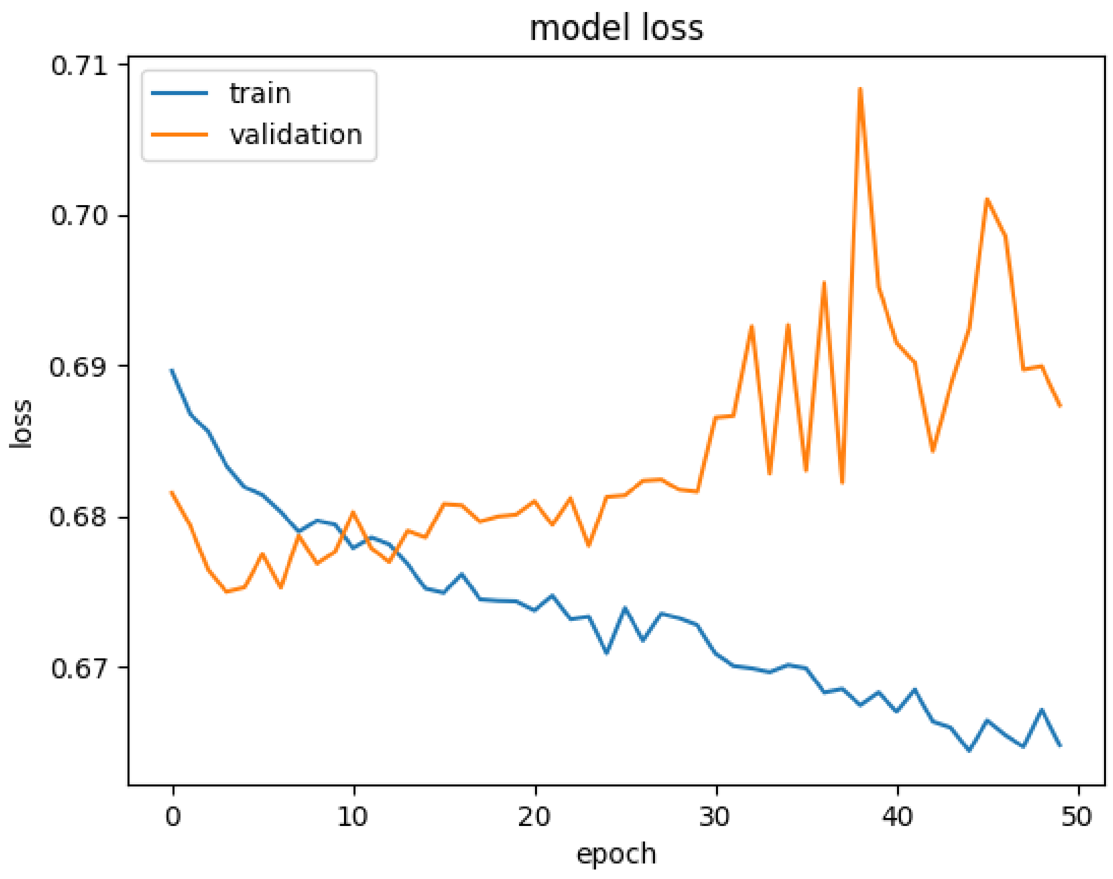


Fig. 2. A plot of training and test loss against epochs. Using this information we decide to stop training around 10-15 epochs.

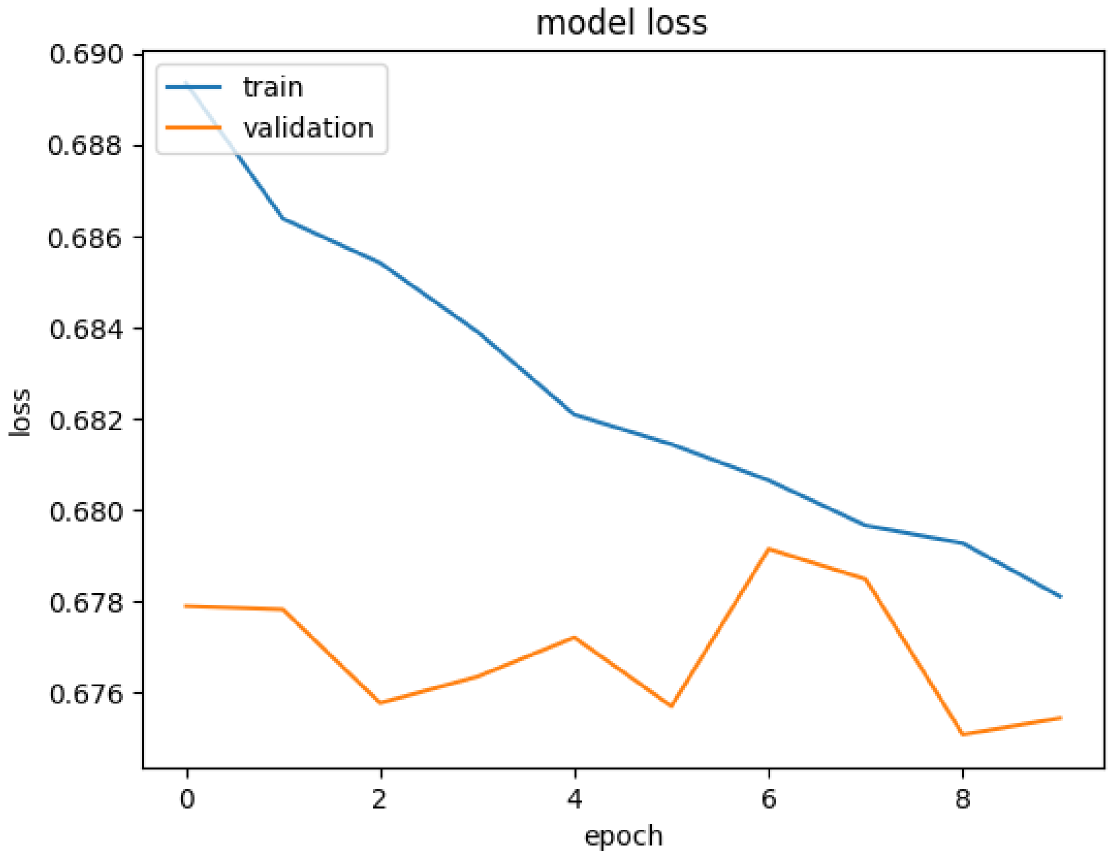


Fig. 3. A plot of training and test loss against epochs for 10 epochs.

## V. Conclusions

This project attempts to improve upon common machine learning models specifically for predicting daily stock option activity, and advising whether to buy/hold or sell on a daily basis. It does so by combining many recent advancements in machine learning into a novel model for time-series regression and classification. Namely, convolutional neural network, attention mechanisms, long short-term memory, and basic fully connected networks are combined in this model. The model sees some improvement over other common machine learning models depending on the data.

Overall, our model predicts the market as successfully as any other AI-powered method out there, improving in certain areas but behaving inconsistently in others. AI-powered market prediction in general needs to continue to improve if it is to be used professionally and on a large scale, especially considering its black-box nature. That said, models are rapidly approaching viability, at least on a small scale. Nearly 80% correct daily market prediction means that these kinds of tools are starting to make predictions that traders should seriously calculate, weigh and consider. As a small-scale project, there is room to improve on feature engineering and more, but the performance of the model is more than satisfactory.

## VI. Future Work

In the future, research can be done by developing new models and by improving existing ones. As technology improves, more complex models will be able to be trained in a reasonable amount of time, and more representational power will be built into these models. To improve our model, we would continue fine-tuning certain aspects of its design, including improving the model architecture by continuing to experiment with layer order, size, and type. It would also be interesting to see the performance of an ensemble classifier built out of the most successful AI-powered market prediction methods, to see if they can cover for each others' weaknesses or if they instead share common weaknesses. Training this model on more datasets would also potentially give insight into the difference we see in its performance between the IBM and S&P datasets.

Another major improvement that could be made in this field is finding more optimal input features (a process known as feature engineering). Ideally, the optimal features would be independent of each other and encompass all information about a stock, including public sentiment and other fairly intangible elements. This is a continual challenge for market prediction, AI-powered or not. That said, feature engineering is critical to researching AI-powered models, and our model could stand to improve given more optimal input features.

A major shortcoming of these tools is their inability to predict long-term trends. Our tool, along with many others, fails to predict prices well at all even just a few days out. This is another direction that can be investigated for research, since most investments are over a longer term and would greatly benefit from long-term price prediction.

Another important area of work for AI-powered market prediction is the area of explainable AI. Even if these tools performed well enough to be implemented professionally, people act with extreme caution when investing their money into a ”black box” tool whose decisions are difficult for anyone to explain perfectly. For this reason, not only do the models actually need to improve, but they need to become explainable so that people are willing to trust them with their capital.

Clearly, there is a lot of work to be done in this area, and research can go in many different directions. However, once these models have improved and become explainable, the realm of market prediction will likely change permanently.

## References

[1] Gang Liu, Jiabao Guo, Bidirectional LSTM with attention mechanism and convolutional layer for text classification, Neurocomputing, Volume 337, 2019, Pages 325-338, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2019.01.078. (https://www.sciencedirect.com/science/article/pii/S0925231219301067)

[2] Kim, Y., Denton, C., Hoang, L., & Rush, A. M. (2017). Structured attention networks. arXiv preprint arXiv:1702.00887.

[3] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems.

[4] Mokhtari, S., Yen, K. K., & Liu, J. (2021). Effectiveness of artificial intelligence in stock market prediction based on machine learning. arXiv preprint arXiv:2107.01031.

[5] Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM–a tutorial into long short-term memory recurrent neural networks. arXiv preprint arXiv:1909.09586.