# Time series Forecasting for Stock Prices

Achyut Saroch
Group ID ; 1
RA1811003030302

# Abstract

This is project is based on the utility of CNN network over Time series data.

In this project I will able to predict Stock Prices in near future using Time series forecasting while using a CNN model. I am building this project in Java Script not in python as Tensorflow is available for JavaScript. JavaScript is better than python as it will help to implement deep learning on Web Based FrameWork and JavaScript is somewhat faster than python and model will light for machine. Via JavaScript Api calling will be easy rather than making a Flask script for python GUI in Web.

# Technologies used

Tensorflow.js

JavaScript

Html and CSS



# Introduction

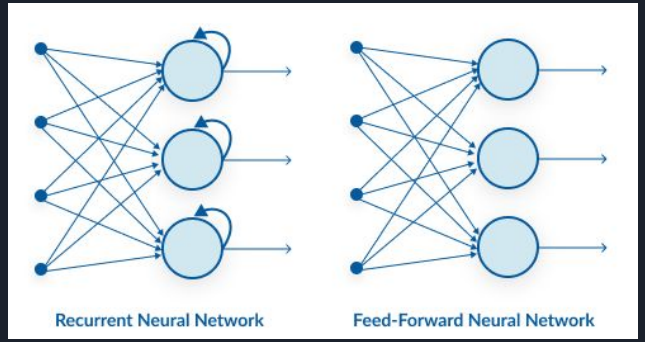
**Machine learning** (**ML**) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so.

A **neural network** is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, **neural networks** refer to systems of neurons, either organic or artificial in nature.

# Type of Neural Network

## RNN

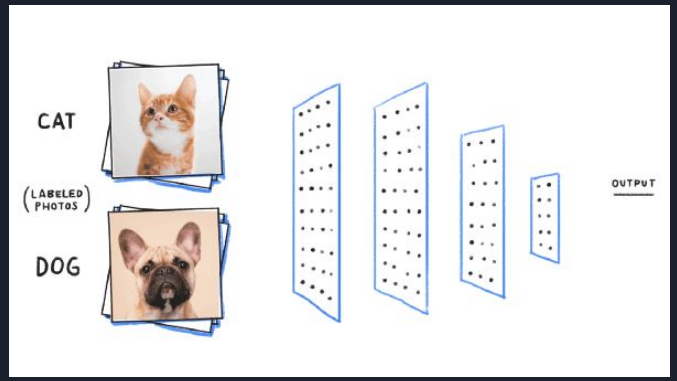
A **recurrent neural network** (**RNN**) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs.This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.



# Type of Neural Network

## CNN

In deep learning, a **convolutional neural network** (**CNN**, or **ConvNet**) is a class of deep neural networks, most commonly applied to analyzing visual imagery.They are also known as **shift invariant** or **space invariant artificial neural networks** (**SIANN**), based on their shared-weights architecture and translation invariance characteristics.They have applications in image and video recognition, recommender systems,image classification, medical image analysis, natural language processing,and financial time series.



# Vanishing Of Gradient

In machine learning, the **vanishing gradient problem** is encountered when training artificial neural networks with gradient-based learning methods and backpropagation. In such methods, each of the neural network weights receive an update proportional to the partial derivative of the error function with respect to the current weight in each iteration of training. The problem is that in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value. In the worst case, this may completely stop the neural network from further training. As one example of the problem cause, traditional activation functions such as the hyperbolic tangent function have gradients in the range $(0, 1)$, and backpropagation computes gradients by the chain rule. This has the effect of multiplying $n$ of these small numbers to compute gradients of the "front" layers in an $n$-layer network, meaning that the gradient (error signal) decreases exponentially with $n$ while the front layers train very slowly.

# Stop Vanishing gradient Problem in Different Neural Network

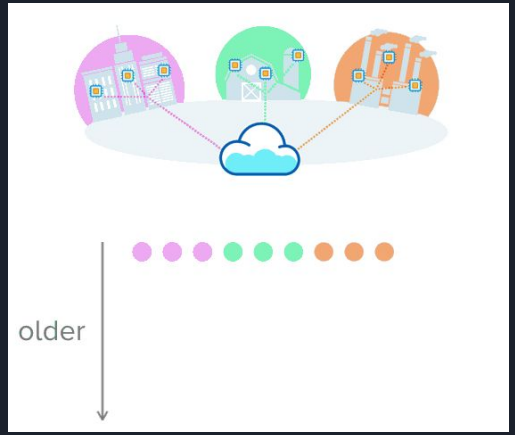
**RNN:**

In case of RNN we can use LSTM (Long Short-term Memory). Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).

CNN:

In case of CNN we can use ReLU. The **rectified linear activation function** or **ReLU** for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

# What is time Series Data and Forecasting?




A **time series** is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. Examples of time series are heights of ocean tides, counts of sunspots, and the daily closing value of the Dow Jones Industrial Average.

**Time series *analysis*** comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. **Time series *forecasting*** is the use of a model to predict future values based on previously observed values. While regression analysis is often employed in such a way as to test theories that the current values of one or more independent time series affect the current value of another time series, this type of analysis of time series is not called "time series analysis", which focuses on comparing values of a single time series or multiple dependent time series at different points in time. Interrupted time series analysis is the analysis of interventions on a single time series.

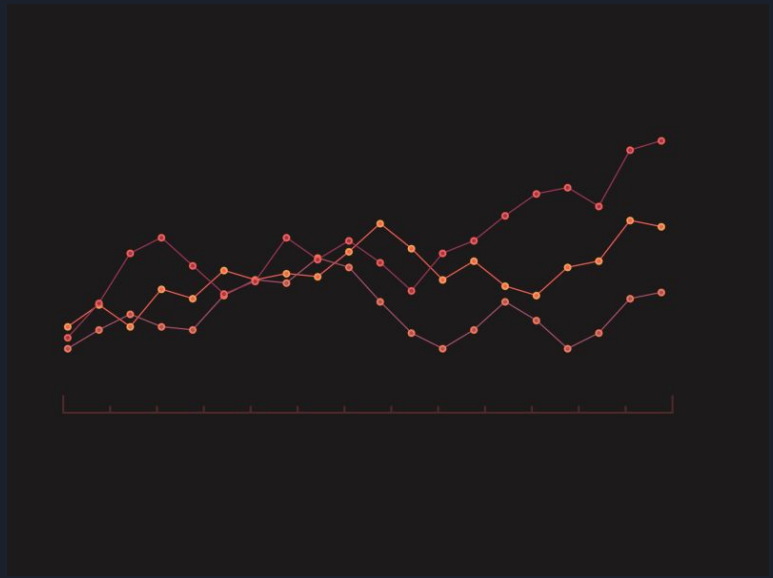
# 1D Convolutional Neural Network

Convolutional Neural Network (CNN)  models were developed for image classification, in which the model accepts a two-dimensional input representing an image's pixels and color channels, in a process called feature learning.

This same process can be applied to one-dimensional sequences of data. The model extracts features from sequences data and maps the internal features of the sequence. A 1D CNN is very effective for deriving features from a fixed-length segment of the overall dataset, where it is not so important where the feature is located in the segment.

# Problem Statement

Forecast the Stock price of a given Stock using Convolutional Neural Network by method of Time series Forecasting where data are time dependent.



# Proposed Solution

This solution use a 1D Convolutional Neural Network as the problem is single variant problem. Here I use slide window technique where the last window define the feature for the next prediction.

Than build  a 1D CNN using tensorflow.js . This CNN consist of  7 layer (inputLayer,2 covn1d layer,2 average pooling layer,flatten,dense.

# Result

The project was success as the  1D Convolutional Neural Network was able to predict the stock prices for the next stock day with average of loss (prediction error)  = 0.003101  on 1 year data.

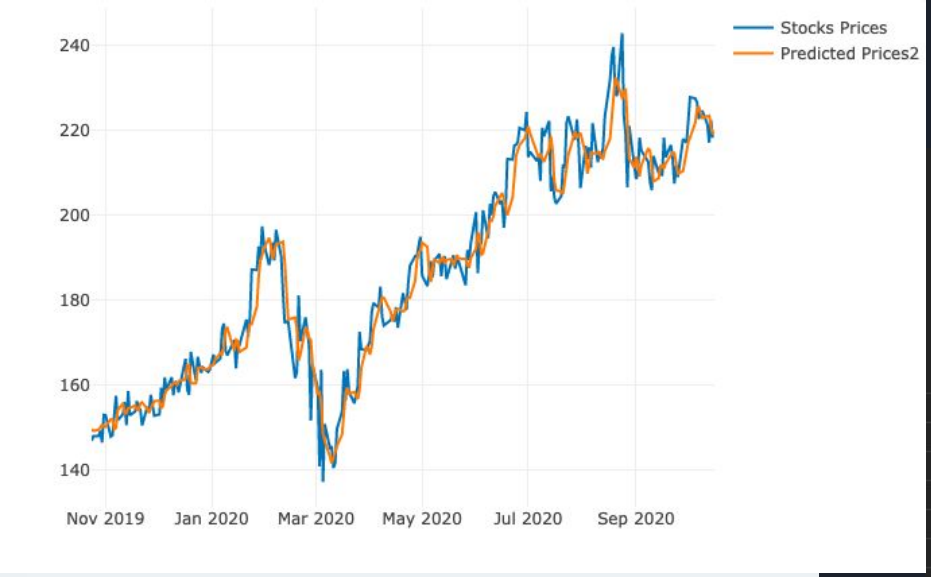
Source code: https://github.com/kosac-achyut/StockPredictor_js

Application Link:  https://kosac-achyut.github.io/StockPredictor_js/

| Epoch | Date Size(total Object in Data) | MeanSquaredError |
|---|---|---|
| 100 | 247 | 0.0028 |
| 100 | 247 | 0.0034 |
| 100 | 247 | 0.00307 |
| 100 | 247 | 0.00307 |
| 100 | 247 | 0.0032 |
| 100 | 247 | 0.00307 |
| Average | | 0.003101666667 |

# Result
# CNN model having 7 layers

```
const CNN_build = function (data) {
    return new Promise(function (resolve, reject) {

        console.log(data);
        const model = tf.sequential();

        model.add(tf.layers.inputLayer({
            inputShape: [7, 1],
        }));

        model.add(tf.layers.conv1d({
            kernelSize: 2,
            filters: 128,
            strides: 1,
            use_bias: true,
            activation: 'relu',
            kernelInitializer: 'VarianceScaling'
        }));

        model.add(tf.layers.averagePooling1d({
            poolSize: [2],
            strides: [1]
        }));

        model.add(tf.layers.conv1d({
            kernelSize: 2,
            filters: 64,
            strides: 1,
            use_bias: true,
            activation: 'relu',
            kernelInitializer: 'VarianceScaling'
        }));

        model.add(tf.layers.averagePooling1d({
            poolSize: [2],
            strides: [1]
        }));


        model.add(tf.layers.flatten({

        }));

        model.add(tf.layers.dense({ // to add linearity property
            units: 1,
            kernelInitializer: 'VarianceScaling',
            activation: 'linear'
        }));

        return resolve({
            'model': model,
            'data': data
        });
    });
}
```




Getting Data for Stock : msft

Optimization Algorithm : adam
Generating Features.....
Building CNN Model.....
Model Training and Testing.....
Finished ΔΔΔΔ

Loss after last Epoch (100) is: 0.0030708992853760 72
Predicted Stock Price of msft for date 27-10-2020 is: 219.695$
Model Excution time: 217.4 seconds

script.js:11

```
▼Object
   max: 242.91
   min: 137.18
  ▶originalData: (254) [140.7, 143.06, 148.07, 147.6, 148.01, 148.03, 146.9, 146.85, 147.97, 148.01, 149.23,…
   size: 247
   timePortion: 7
  ▶trainX: (1729) [0.03329234843469197, 0.05561335477158797, 0.10299820296982869, 0.0985529178095147, 0.1024…
  ▶trainY: (247) [0.09145937766007745, 0.10205239761657045, 0.10243071975787368, 0.11396954506762494, 0.0877…
  ▶__proto__: Object
```

MODEL SUMMARY: script.js:65

layer utils.ts:62

| Layer (type) | Output shape | Param # | |
|---|---|---|---|
| | | | layer utils.ts:152 |
| | | | layer utils.ts:64 |
| input1 (InputLayer) | [null,7,1] | 0 | layer utils.ts:152 |
| | | | layer utils.ts:74 |
| conv1d_Conv1D1 (Conv1D) | [null,6,128] | 384 | layer utils.ts:152 |
| | | | layer utils.ts:74 |
| average_pooling1d_AveragePoo | [null,5,128] | 0 | layer utils.ts:152 |
| | | | layer utils.ts:74 |
| conv1d_Conv1D2 (Conv1D) | [null,4,64] | 16448 | layer utils.ts:152 |
| | | | layer utils.ts:74 |
| average_pooling1d_AveragePoo | [null,3,64] | 0 | layer utils.ts:152 |
| | | | layer utils.ts:74 |
| flatten_Flatten1 (Flatten) | [null,192] | 0 | layer utils.ts:152 |
| | | | layer utils.ts:74 |
| dense_Dense1 (Dense) | [null,1] | 193 | layer utils.ts:152 |
| | | | layer utils.ts:74 |

Total params: 17025 layer utils.ts:83
Trainable params: 17025 layer utils.ts:84
Non-trainable params: 0 layer utils.ts:85
layer utils.ts:86

StockPrice Predictor

Home DOCs GitHub

# StockPrice Predictor

# v1.2

Using the power of Convolutional Neural Network now we can predict the Stock Prices form the given Time Series Data. By using 1d Convolutional Neural Network we are able to feed sequentail data to Convolutional Network and are able to use the power and speed CNN model to predict Value to given Stock on that Day.

Chose stock to predict the value of:

Select

Chose Chart time:

Select

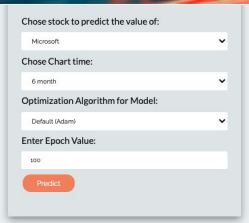
Optimization Algorithm for Model:

Default (Adam)

Enter Epoch Value:

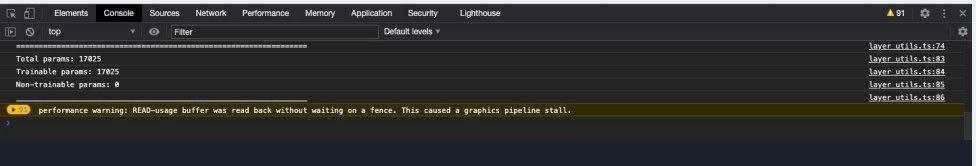
100

Predict





```
Getting Data for Stock : msft

Optimization Algorithm : adam
Generating Features....
Building CNN Model....
Model Training and Testing....
Finished ΔΔΔΔ

Loss after last Epoch (100) is: 0.008049661293626832
Predicted Stock Price of msft for date 27-10-2020 is 219.427$
Model Excution time: 20.35 seconds
```




Stock Prediction

Home

## Stock Predictor using 1D Convolutional Nueral Network.

Convolutional neural networks, also called ConvNets, were first introduced in the 1980s by Yann LeCun, a postdoctoral computer science researcher. Till then Machine Learing has come a long way. In 2019 while reading a article on 2D CNN I came across 1D CNN and it's useage in Time Series Data Manuplation. This caught my attention since CNN is specifically designed to process pixel data and used in image recognition and processing and it looked like a interesting challenge.

This application is frontend only so there is no need for backserver. And by using Tensorflow.js it is easy to implement the 1D CNN. For code click on the GitHub link in nav.

All the work that need to be done can be set up in 6 steps:

1. Get the Data
2. Generate Features
3. Generate Ml Model
4. Train the ML Model
5. Test the ML Model
6. Predict with Ml Modle

### Get the Data

Here I have used API form IEX. In this application I'm are using the chart endpoint which has predifined historical period. You can chose chart form 6 month to 5 year.

```
url = "https://cloud.iexapis.com/stable/stock/"+symbol+"/chart/"+chart_time+"?token="+token;
```

The result from this API is json array with historical data for the requested company. Below is some example from the response data.


```
[
  {
    "date":"2018-02-20",
    "open":169.4694,
    "high":171.6453,
    "low":168.8489,
    "close":169.2724,
    "volume":33930540,
    "unadjustedVolume":33930540,
    "change":-0.5713,
    "changePercent":-0.336,
    "vwap":170.3546,
    "label":"Feb 20, 18",
    "changeOverTime":0
  },
  ...
]
```


### Generate Features

After the data is retrieved we need to process it and prepare the feature set and label set. While I was reading reserach paper, mostly I found paper where they were using date field as feature. So I ended up using 7 feature for my test set and each one of them is leabeled with the stock price for the next day. The reason was Firstly, the date is all ways incressing. Secondly, date are independent of stock price.

```
// timePortion = 7
```

# Conclusion

In this project have introduced a CNN model to make the stock prediction and used a conv1d function to process the 1D data in the convolutional layer. The result has been evaluated by different stock data, and finally indicates that our CNN model is robust and can also be used to make the predictions even if the source data is 1D sequential.

There are some interesting directions for further study:

(1) We can use the stock data with more features to make predictions, for the movement of stock may not only influenced by the features of open, close, high and low prices. Moreover, we can make a comparison about which features are key to the result.

(2) We can use more data sets to judge whether the result will be better, for the number of the data-sets is of great importance in deep learning.

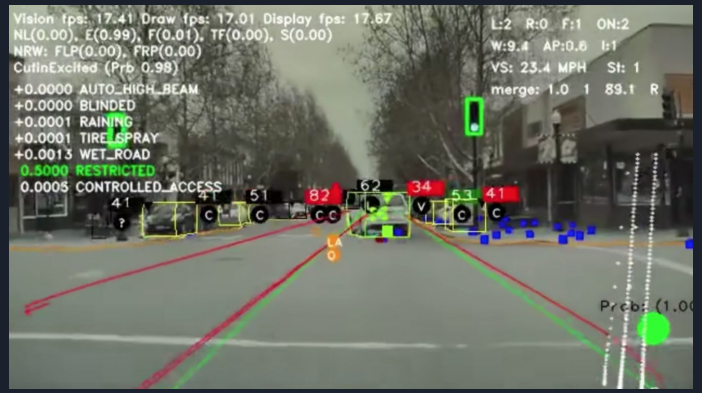
(3) If feature are implemented efficiently by increasing epoch we can decrease the loss. But will increase the complexity of the Model.

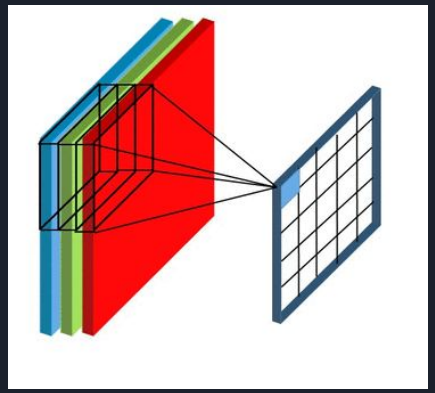
# Motivation For Project

The main Motivation for the project is usage of CNN . Most of the forecasting Method for stock are  RNN with LSTM based and most of the industrial model like Arima (AutoRegressive Integrated Moving Average) uses linear Regression and RNN. As more and more paper are coming out on usage of CNN in forecasting it's proved that we can use CNN in forecasting and avail better result than RNN. As all model of neural network are prone vanishing gradient problem CNN is most least to Prone and if there is we can somewhat eliminate it by ReLu ( Rectified linear activation function). The other motivation is  JavaScript. Most of the people think than JavaScript is used only in Web Development but as language compiler has advance it can be also used in Machine learning which can be directly applied to Web App with or without any FrameWork. I believe in future javascript will surpass python in Machine learning application use.

# Contribution and Future Scope

This project is implementation on the ongoing research on Convolutional Neural Network and how to exploit their power in different field. One of the field is about Time Series Data where data change with respect of time. Some company like Tesla use Convolutional Neural Network to Auto drive there car and data come inform of Data Images or Data Points. Most of the Stock Predictor and Bot uses RNN network as RNN is less Complicated than CNN but Library are become stronger is easier to implement to use CNN so why not Exploit the power of CNN in that structure also.



# Reference

Article:


1. https://en.wikipedia.org/wiki/Time_series
2. https://en.wikipedia.org/wiki/Long_short-term_memory
3. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
4. https://www.tensorflow.org/js
5. https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/
6. https://towardsdatascience.com/stock-market-action-prediction-with-convnet-8689238feae3/


Research Papers:


Stock Price Prediction Using Convolutional Neural Networks on a Multivariate Time Series by Sidra Mehtab and Jaydip Sen (https://arxiv.org/abs/2001.09769)

Deep learning for time series classification: a review (https://arxiv.org/pdf/1809.04356.pdf)


Books


Hands on Machine Learning with Scikit-learn and TensorFlow by *Aurélien Géron*