STOCK PRICE PREDICTION PROJECT

PROJECT REPORT

MD JAVED

Year/Sem

Mrjaved.1993@outlook.com

INTRODUCTION

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly.

But, using the classic algorithms of machine learning, text is considered as a sequence of keywords; instead, an approach based on semantic analysis mimics the human ability to understand the meaning of a text.

Some Machine Learning Methods

Machine learning algorithms are often categorised as supervised or unsupervised.

- Supervised machine learning algorithms can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
- In contrast, unsupervised machine learning algorithms are used when the information used to train is neither classified nor labelled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
- Semi-supervised machine learning algorithms fall somewhere in between supervised and unsupervised learning, since they use both

labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.

- Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

How to choose the right machine learning model

The process of choosing the right machine learning model to solve a problem can be time consuming if not approached strategically.

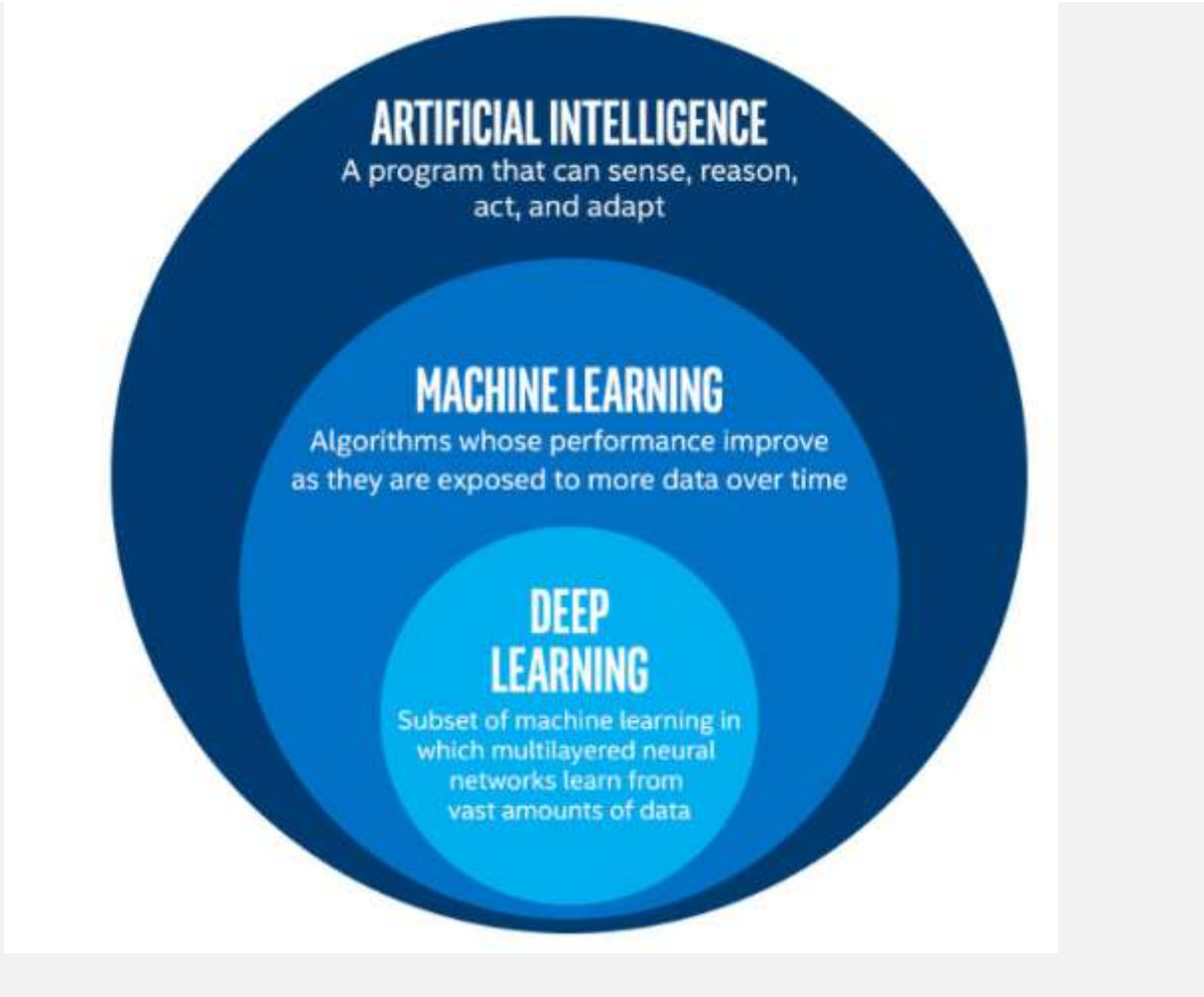
Step 1: Align the problem with potential data inputs that should be considered for the solution. This step requires help from data scientists and experts who have a deep understanding of the problem.

Step 2: Collect data, format it and label the data if necessary. This step is typically led by data scientists

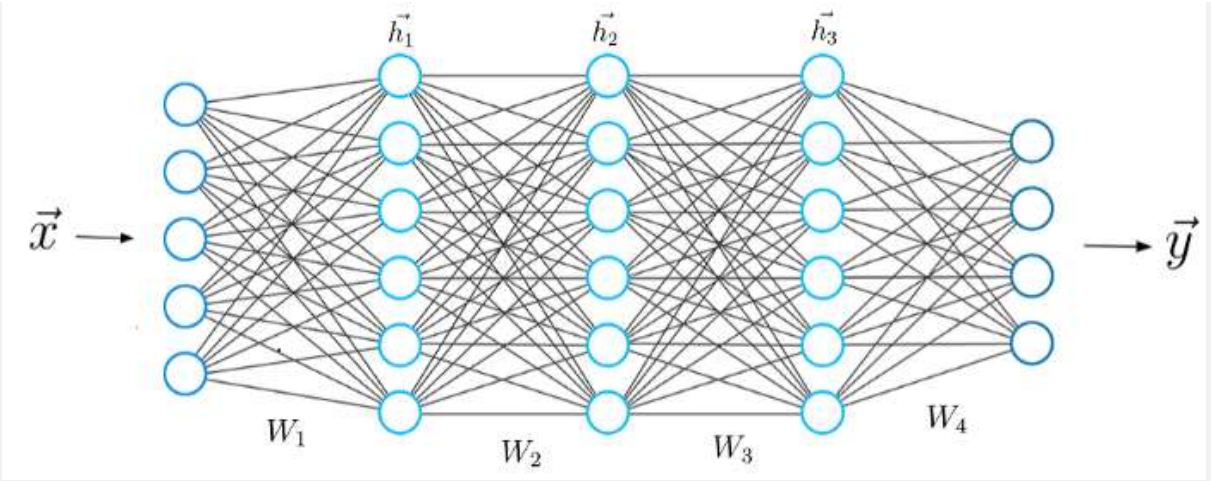
Step 3: Choose which algorithm(s) to use and test to see how well they perform. This step is usually carried out by data scientists.

Step 4: Continue to fine tune outputs until they reach an acceptable level of accuracy. This step is usually carried out by data scientists with feedback from experts who have a deep understanding of the problem.

Deep Learning is a subset of Machine Learning, which on the other hand is a subset of Artificial Intelligence. Artificial Intelligence is a general term that refers to techniques that enable computers to mimic human behavior. Machine Learning represents a set of algorithms trained on data that make all of this possible.



Deep Learning, on the other hand, is just a type of Machine Learning, inspired by the structure of a human brain. Deep learning algorithms attempt to draw similar conclusions as humans would by continually analyzing data with a given logical structure. To achieve this, deep learning uses a multi-layered structure of algorithms called neural networks.



A typical Neural Network.

The design of the neural network is based on the structure of the human brain. Just as we use our brains to identify patterns and classify different types of information, neural networks can be taught to perform the same tasks on data.

The individual layers of neural networks can also be thought of as a sort of filter that works from gross to subtle, increasing the likelihood of detecting and outputting a correct result.

The human brain works similarly. Whenever we receive new information, the brain tries to compare it with known objects. The same concept is also used by deep neural networks.

Neural networks enable us to perform many tasks, such as clustering, classification or regression. With neural networks, we can group or sort unlabeled data according to similarities among the samples in this data. Or in the case of classification, we can train the network on a labeled dataset in order to classify the samples in this dataset into different categories.

Artificial neural networks have unique capabilities that enable deep learning models to solve tasks that machine learning models can never solve.

All recent advances in artificial intelligence in recent years are due to deep learning. Without deep learning, we would not have self-driving cars, chatbots or personal assistants like Alexa and Siri. The Google Translate app would continue to be as primitive as 10 years ago (before Google switched to neural networks for this App), and Netflix or Youtube would have no idea which movies or TV series we like or dislike. Behind all these technologies are neural networks.

We can even go so far as to say that today a new industrial revolution is taking place, driven by artificial neural networks and deep learning.

Long before deep learning was used, traditional machine learning methods were mainly used. Such as Decision Trees, SVM, Naïve Bayes Classifier and Logistic Regression.

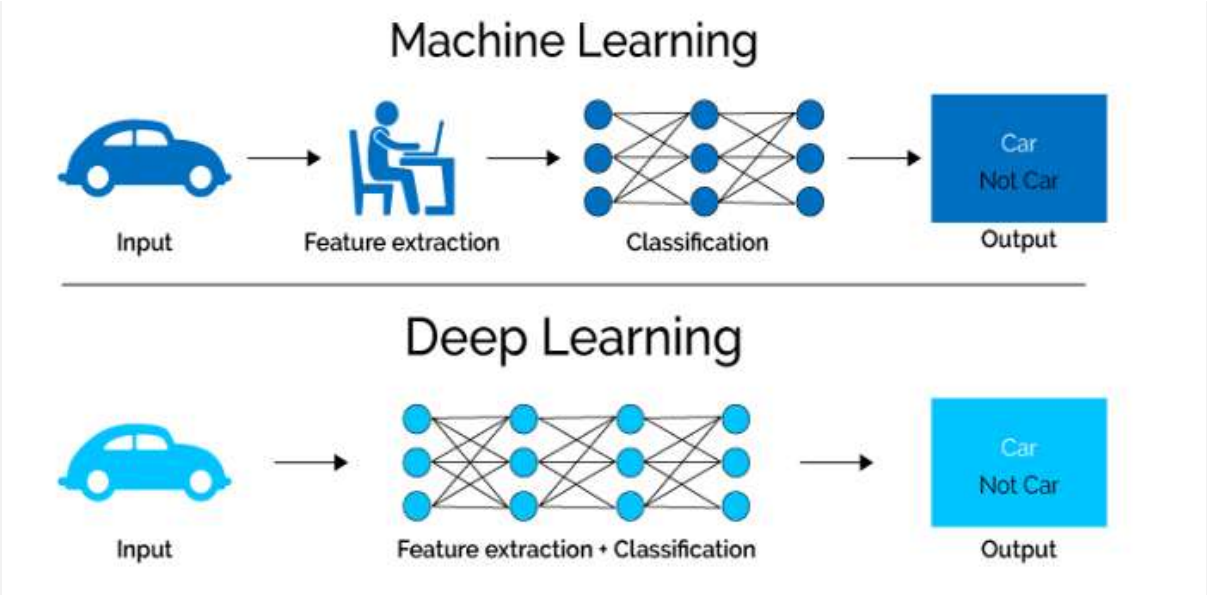
These algorithms are also called flat algorithms. Flat here means that these algorithms can not normally be applied directly to the raw data (such as .csv, images, text, etc.). We need a preprocessing step called Feature Extraction.

The result of Feature Extraction is a representation of the given raw data that can now be used by these classic machine learning algorithms to perform a task. For example, the classification of the data into several categories or classes.

Feature Extraction is usually quite complex and requires detailed knowledge of the problem domain. This preprocessing layer must be adapted, tested and refined over several iterations for optimal results.

On the other side are the artificial neural networks of Deep Learning. These do not need the Feature Extraction step.

The layers are able to learn an implicit representation of the raw data directly and on their own. Here, a more and more abstract and compressed representation of the raw data is produced over several layers of an artificial neural-net. This compressed representation of the input data is then used to produce the result. The result can be, for example, the classification of the input data into different classes.



Feature Extraction is only required for ML Algorithms.

During the training process, this step is also optimized by the neural network to obtain the best possible abstract representation of the input data. This means that the models of deep learning thus require little to no manual effort to perform and optimize the feature extraction process.

Let us look at a concrete example. For example, if you want to use a machine learning model to determine if a particular image is showing a car or not, we humans first need to identify the unique features or features of a car (shape, size, windows, wheels, etc.) extract the feature and give them to the algorithm as input data.

In this way, the algorithm would perform a classification of the images. That is, in machine learning, a programmer must intervene directly in the action for the model to come to a conclusion.

In the case of a deep learning model, the feature extraction step is completely unnecessary. The model would recognize these unique characteristics of a car and make correct predictions.

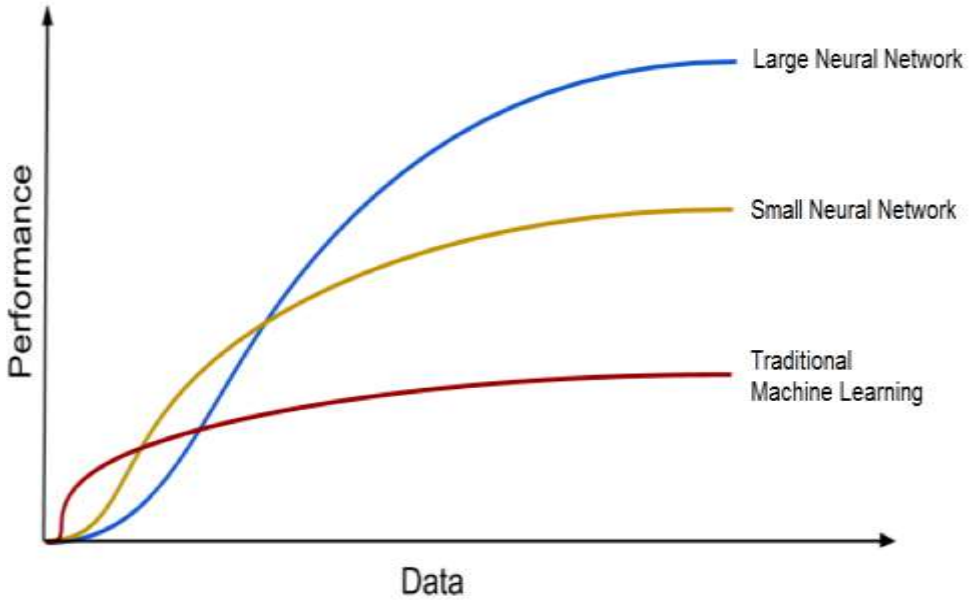
That completely without the help of a human.

In fact, refraining from extracting the characteristics of data applies to every other task you'll ever do with neural networks. Just give the raw data to the neural network, the rest is done by the model.

The Era of Big Data…

The second huge advantage of Deep Learning and a key part in understanding why it's becoming so popular is that it's powered by massive amounts of data. The "Big Data Era" of technology will provide huge amounts of opportunities for new innovations in deep learning. As per Andrew Ng, the chief scientist of China's major search engine Baidu and one of the leaders of the Google Brain Project,

"*The analogy to deep learning is that the rocket engine is the deep learning models and the fuel is the huge amounts of data we can feed to these algorithms.*"



Deep Learning Algorithms get better with the increasing amount of data.

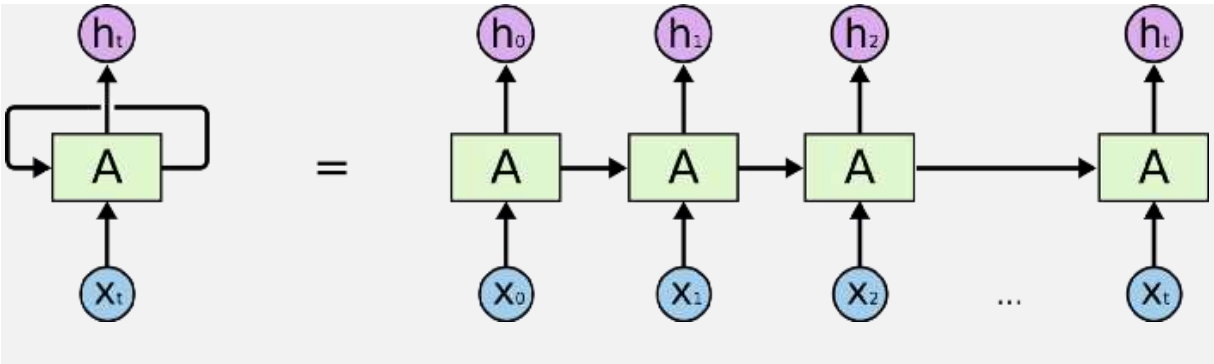
Deep Learning models tend to increase their accuracy with the increasing amount of training data, where's traditional machine learning models such as SVM and Naive Bayes classifier stop improving after a saturation point.

DEEP LEARNING ALGORITHMS:

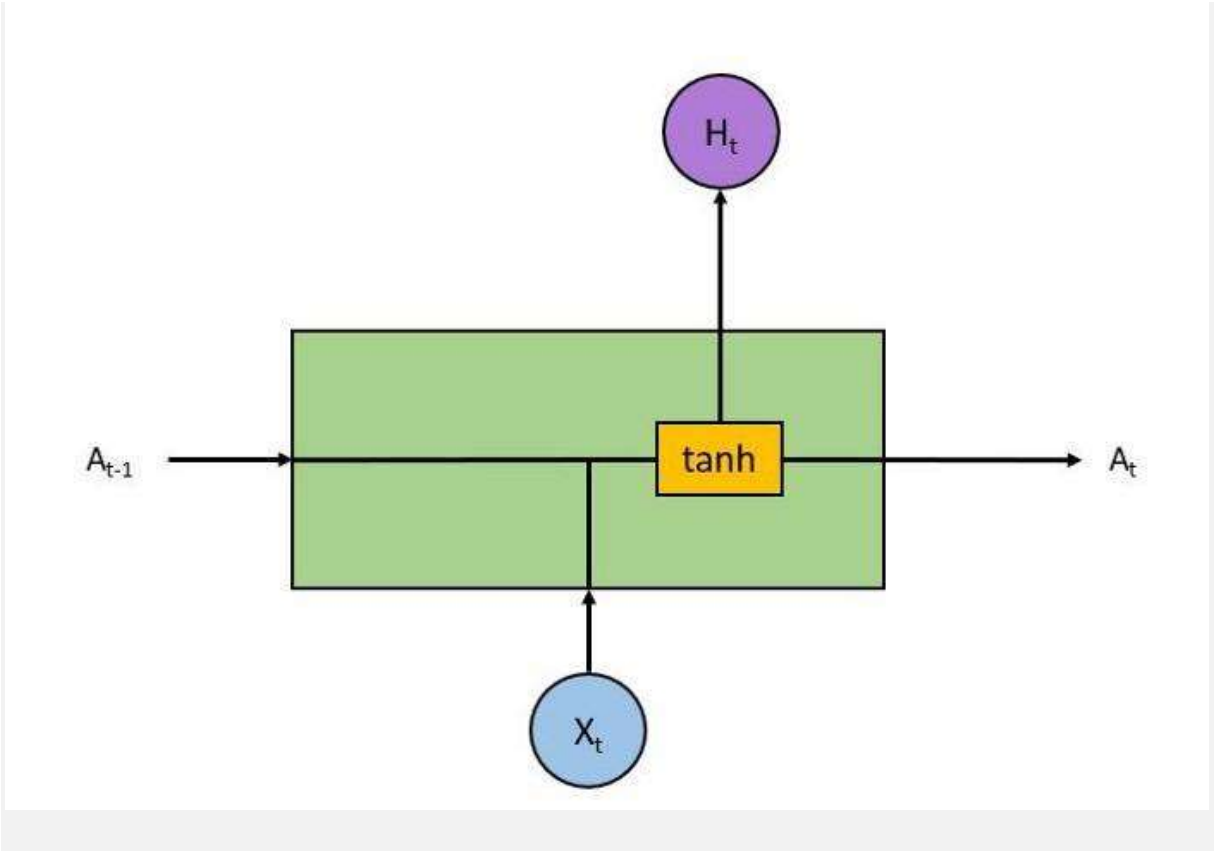
LSTM:

Long Short Term Memory

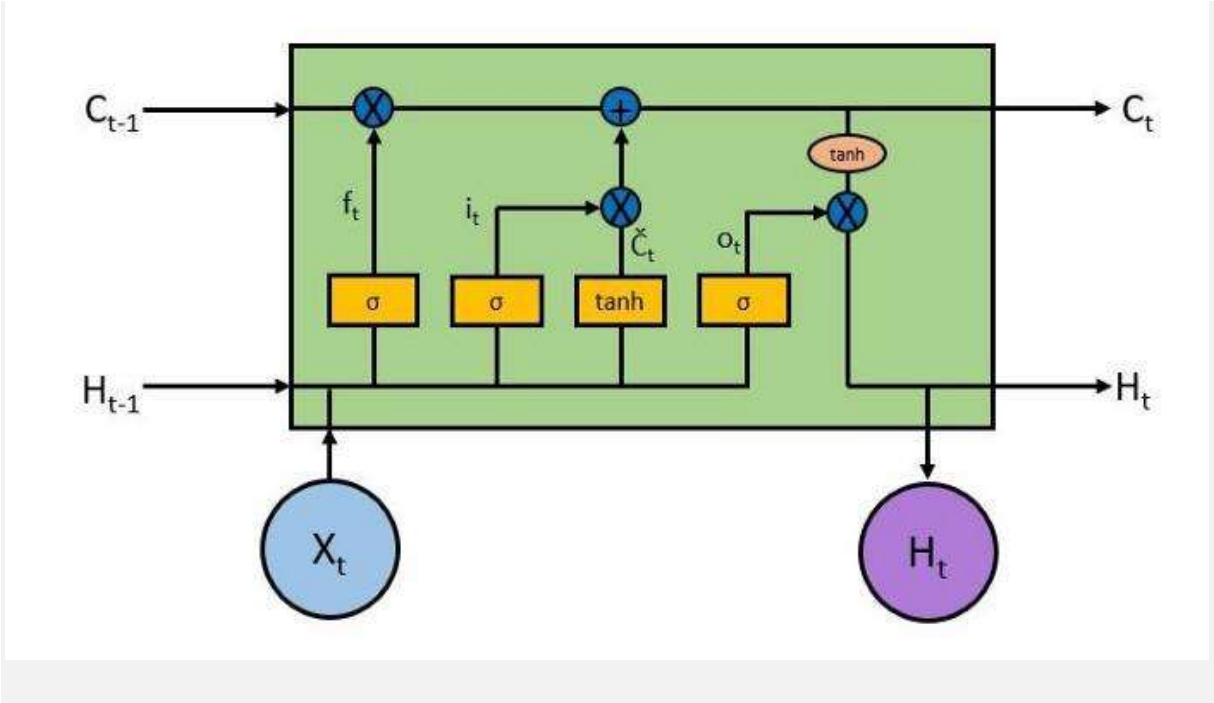
Long Short Term Memory networks — usually just called LSTMs — are a special kind of RNN, capable of learning long-term dependencies. Refined and popularized by many people in following work. They work tremendously well on a large variety of sequence modelling problems, and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is their default behavior. Let's recall how an RNN looks



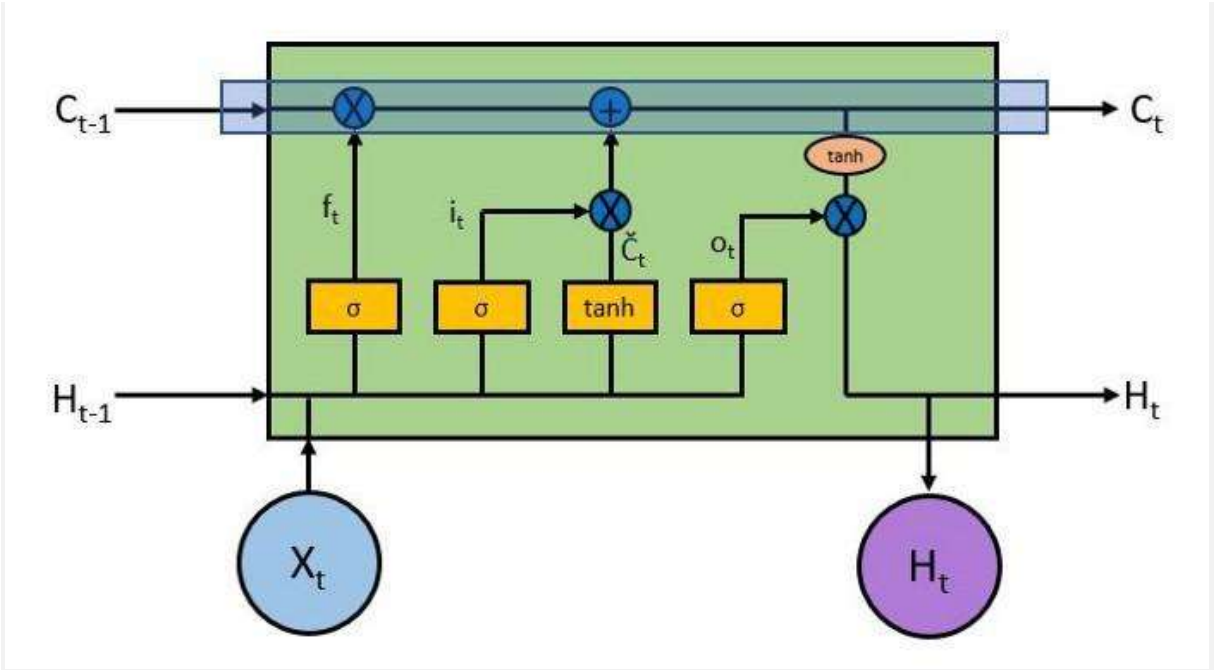
As we saw in the RNN article the RNN unit takes the current input (X) as well as the previous input (A) to produce output (H) and current state (A)



LSTMs also have a similar structure though the internals have different components as compared to a single tanh (activation) layer in the RNN. There are 4 layers inside an LSTM block which interact together.



At first it looks pretty complicated and intimidating but lets try to break it down and understand what is the purpose of each layer and block. The key to the operation of LSTM is the top horizontal line running from left to right enclosed in the highlight below.



With some minor linear interactions along this line the cell state C allows information to flow through the entire LSTM unchanged which enables LSTM to remember context several time steps in the past. Into this line there are several inputs and outputs which allow us to add or remove information to the cell state. The addition or removal of information is controlled by gates. These are the sigmoid layers (Yellow boxes inside the RNN cell). They output numbers between zero and one, describing how much of each component should be let through. A value of zero means let nothing through, while a value of one means let everything through. An LSTM has three of these gates to control the cell state.

Forget Gate

Let's look at the first gate which is called the forget gate. This gate decides what information we're going to throw away from the cell state. This is decided by the first sigmoid layer which looks at the previous output and the current input —

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

Equation for the forget gate

Consider a sentence of which we are trying to predict the next word — *Bob called Carla to ask her out.* In this sentence the pronoun her is based on the subject Carla and not Bob so the machine while making prediction will have to *forget* the context *Bob* when it encounters a new subject *Carla*. This is what a forget gate accomplishes.

Now the next step involves what are we going to store in the cell state C.

Input Gate

The input gate is another sigmoid layer (Second yellow box from the left in the picture above) which outputs numbers between 0 and 1 and decides which values to update. The candidate values which will be used to update

the cell state are calculated by a tanh layer (Third yellow box from the left) and these two are combined to create an update to the state.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$

$$\widetilde{C}_t = \tanh(W_C * [h_{t-1}, x_t] + b_C)$$

In the sentence example above, we'd want to add the gender of the new subject to the cell state, to replace the old one we're forgetting.

It's now time to update the old cell state into the new cell state. The previous steps already decided what to do, we just need to actually do it.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$$

In the case of the language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.

Output Gate

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer (The rightmost yellow box inside the cell) which decides what parts of the cell state we're going to output. Then, we put the cell state through a tanh layer to push the values to be between -1 and 1 and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to. Mathematically it looks like —

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

For the language model example, since it just saw a subject, it might want to output information relevant to a verb. For example, in our case Carla is singular so it might output information about singular or plural, so that we know what form a verb should be conjugated into.

2. Dropout

Dropout is a regularization technique for neural networks that drops a unit (along with connections) at training time with a specified probability p

(a common value is p=0.5). At test time, all units are present, but with weights scaled by p (i.e. w becomes pw).

The idea is to prevent co-adaptation, where the neural network becomes too reliant on particular connections, as this could be symptomatic of overfitting. Intuitively, dropout can be thought of as creating an implicit ensemble of neural networks.

3. Dense

The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models.

In the background, the dense layer performs a matrix-vector multiplication. The values used in the matrix are actually parameters that can be trained and updated with the help of backpropagation.

The output generated by the dense layer is an ‘m’ dimensional vector. Thus, dense layer is basically used for changing the dimensions of the vector. Dense layers also applies operations like rotation, scaling, translation on the vector.

Keras Dense Layer Parameters

Let us see different parameters of dense layer function of Keras below –

A. Units

The most basic parameter of all the parameters, it uses positive integer as it value and represents the output size of the layer.

It is the unit parameter itself that plays a major role in the size of the weight matrixalong with the bias vector.

B. Activation

The activation parameter is helpful in applying the element-wise activation function in a dense layer. By default, Linear Activation is used but we can alter and switch to any one of many options that Keras provides for this.

C. Use_Bias

Another straightforward parameter, use_bias helps in deciding whether we should include a bias vector for calculation purposes or not. By default, use_bias is set to true.

D. Initializers

As its name suggests, the initializer parameter is used for providing input about how values in the layer will be initialized. In case of the Dense Layer, the weight matrix and bias vector has to be initialized.

E. Regularizers

Regularizers contain three parameters that carry out regularization or penalty on the model. Generally, these parameters are not used regularly but they can help in the generalization of the model.

F. Constraints

This last parameter determines the constraints on the values that the weight matrix or bias vector can take.

4. SEQUENTIAL

Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. Each layer has weights that correspond to the layer that follows it.

System Requirements

Hardware Requirements

CPU TYPE: Intel i3, i5, i7 or AMD
RAM Size: Min 512 MB
Hard Disk Capacity: Min 2 GB

System Requirements

Operating system: Windows, Linux, Android, iOS
Programming Language: Python
IDE: VSC, Jupyter Notebook, Pycharm, Anaconda Cloud IDE:
Google Collab

Datasets

GOOGLE DATASET

Google LLC is an American multinational technology company that

specialises in Internet-related services and products, which include online advertising technologies, a search engine, cloud computing, software, and hardware.

It is considered one of the big four Internet stocks along with Amazon, Facebook, and Apple

The company is listed on the **NASDAQ** stock exchange under the ticker symbol **GOOG**.

We have Included a 5 year Stock Price of Google for this Project.

PROGRAMMING

Stock Price Predicition

"""

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd



"""**Data Preprocessing** """



#loading the Data

dataset_train = pd.read_csv('Google_Stock_Price_Train.csv')

print('shape is = {}'.format(dataset_train.shape))

print(dataset_train.head())



training_set = dataset_train.iloc[:,1:2].values

print('shape is ={}'.format(training_set.shape))

print(training_set[0:5])



#Visualizing the Data
```

```
plt.plot(training_set, color = 'red', label = 'Google Stock Price in Test set')

plt.xlabel('Time')

plt.ylabel('Google Stock Price')

plt.legend()

plt.show()



#feature Scaling

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler(feature_range=(0,1))

training_set_scaled = sc.fit_transform(training_set)

print(training_set_scaled[0:5])



#preaparing the dataset for Training

X_train = []

y_train = []

for i in range(60,1258):
```

```
    X_train.append(training_set_scaled[i-60:i,0])
    y_train.append(training_set_scaled[i,0])
X_train, y_train = np.array(X_train), np.array(y_train)


print('X_train shape = {}'.format(X_train.shape))
print('y_train shape = {}'.format(y_train.shape))


#reshaping the input data to fit in Keras RNN
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_train.shape


"""Model Development

"""

from keras.models import Sequential
```

```
from keras.layers import Dense

from keras.layers import LSTM

from keras.layers import Dropout



#LSTM Layers with Dropout regularization

regressor = Sequential()

regressor.add(LSTM(units= 50, return_sequences=True, input_shape =
(X_train.shape[1], 1 )))

regressor.add(Dropout(0.2))

regressor.add(LSTM(units=50, return_sequences= True))

regressor.add(Dropout(0.2))

regressor.add(LSTM(units=50, return_sequences= True))

regressor.add(Dropout(0.2))

regressor.add(LSTM(units=50))

regressor.add(Dropout(0.2))
```

```
#Output Layer

regressor.add(Dense(units=1))



#Compiling the model

regressor.compile(optimizer='adam', loss='mean_squared_error')



#fitting the model

regressor.fit(X_train, y_train, epochs=100, batch_size=32)



#loading the Data

dataset_test = pd.read_csv('Google_Stock_Price_Test.csv')

real_stock_price = dataset_test.iloc[:,1:2].values



#preprocessing the Data

dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']),
axis = 0)
```

```
inputs = dataset_total[len(dataset_total)-len(dataset_test)-60:].values

inputs = inputs.reshape(-1,1)

inputs = sc.transform(inputs)

X_test = []

for i in range(60,80):

    X_test.append(inputs[i-60 : i, 0])

X_test = np.array(X_test)

X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))



"""**Output Prediction**"""



#predicting the output

predicted_stock_price = regressor.predict(X_test)

predicted_stock_price = sc.inverse_transform(predicted_stock_price)



"""**Result Visualization**"""
```

```
plt.plot(real_stock_price, color = 'red', label = 'Real Google Stock Price')

plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted Google Stock Price')

plt.title('Google Stock Price Prediction')

plt.xlabel('Time')

plt.ylabel('Google Stock Price')

plt.legend()

plt.show()
```

OUTPUT CODE

```
"""**Output Prediction**"""

#predicting the output

predicted_stock_price = regressor.predict(X_test)
```

```
predicted_stock_price = sc.inverse_transform(predicted_stock_price)



"""**Result Visualization**"""


plt.plot(real_stock_price, color = 'red', label = 'Real Google Stock Price')

plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted Google Stock Price')

plt.title('Google Stock Price Prediction')

plt.xlabel('Time')

plt.ylabel('Google Stock Price')

plt.legend()

plt.show()
```

CONCLUSION

We can see the Prediction, analysis and Visualisation of Google stock Price through applying Deep learning algorithms such as LSTM, DENSE, DROP OUT and SEQUENTIAL.

Same way we can use any company's Stock Dataset directly and apply these algorithms it will give us the correct prediction.

This System Successfully runs on any system even on Cloud platforms.

REFERENCES

1. https://machinelearningmastery.com/start-here/#getstarted

2. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
3. https://www.sas.com/en_us/insights/analytics/machine-learning.html
4. https://groww.in/us-stocks/googl
5. https://www.investopedia.com/terms/d/deep-learning.asp
6. https://www.nasdaq.com/market-activity/stocks/goog