PRODUCT DEMAND FORECASTING USING DEEP LEARNING

A Thesis

Presented to

The Department of Computer Science

Lamar University

In Partial Fulfillment

of the Requirements for the degree

of Master of Science in Computer Science

by

Kartheek Golla

December 2019

PRODUCT DEMAND FORECASTING USING DEEP LEARNING

Kartheek Golla

Approved:

Dr. Kami Makki
Supervising Professor

Dr. Stefan Andrei
Committee Member

Dr. Xingya Liu
Committee Member

Dr. Stefan Andrei
Chair, Department of Computer Science

Dr. Lynn Maurer
Dean,College of Arts and Science

Dr.William Harn
Dean, College of Graduate Studies

ABSTRACT

PRODUCT DEMAND USING DEEP LEARNING

by

Kartheek Golla

This research is intended to present the problem of developing a fast, reliable, accurate model to predict the products' demand for a certain period of time, using the normal state-of-the-art techniques such as Convolutional Neural Network, Recurrent Neural Networks and comparing them with each other to acquire the best model that can be feasible for today's businesses.

Nowadays, the hardest problem for the retail stores is to know the amount of a specific product that needs to be purchased at a time. Do they need to purchase a product less than the previous demand or more than the previous demand? These types of questions mainly depend on the demand of the customers' needs. If these questions are answered, then business goals such as profit maximization, revenue optimization can be achieved, and overstocking or understocking of products will no longer be an issue.

Demand forecasting is a very challenging problem, as time-series data has no fixed style or consistent pattern. This can increase the loss, cause out of stock days, and many other difficulties for businesses.

In order to achieve this goal, the total data of the stores needs to be gathered. So, that one can use the store's data for different products and places. We develop a deep learning model for accurate prediction of the demand for a certain product. Our performance assessment will be based on Symmetric Mean Absolute Percentage Error (sMAPE).

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my advisor, Dr. Kami Makki, for his supervision and immense guidance. I also thank him for his valuable insights and the enormous patience which he exhibited while I was pursuing my research, which helped me to stay motivated and achieve my goal.

I would like to express my gratitude to my committee members Dr. Stefan Andrei and Dr. Xingya Liu, for their valuable guidance during my research work.

I also want to thank all my friends and seniors, (Chandini, Ramya Krishna, Nanda Vardan, Kiran Reddy, Reddi Babu, and Tarun Kumar) who and gave their constant feedback while pursuing research. Their discussions and opinions helped to address some of the issues in my models. Besides, whenever I was stressed, they were always by my side to encourage me.

Finally, I thank my whole family for their blessings and moral support throughout my life. I am indebted to them for supporting me through my entire journey of life so far and future.

Table of Contents

List of Tables

List of Figures

Abbreviations

CNN:	Convolutional Neural Network
RNN:	Recurrent Neural Networks
LSTM:	Long Short-Term Memory
GRU:	Gated Recurrent Units
sMAPE:	Symmetric mean absolute percentage error
CPU:	Central Processing Unit
GPU:	Graphical Processing Unit
DL:	Deep Learning
ML:	Machine Learning
AI:	Artificial Intelligence
DBN:	Deep Belief Networks
DNN:	Deep Neural Networks

Chapter 1

Introduction

Demand forecasting is predicting future demand based on historical time-series data related to demand. Demand forecasting is considered a very challenging and complex problem because it requires considering many factors such as patterns of sales demand, and the order of temporal dependence between the data. This temporal structure can also be an advantage for forecasting because of seasonality and trends that can help to improve the model's skill. Traditionally, demand forecasting can also be solved using various methods like exponential smoothing methods, decision trees, Random forest, Autoregressive Integrated Moving Average models (ARIMA). However, such methods also have limitations, such as corrupted or missing data is not supported, only they work with univariate inputs (Alsharif, Younes, and Kim 2019; Brownlee 2019).

Deep learning models can deal with more complex time-series forecasting problems such as missing data, complex nonlinear relationships, and multiple input variables. So, in our study, we use the deep learning models to solve the time-series forecasting. We develop and compare different models like convolutional neural networks (CNN), recurrent neural networks (RNN) to find the best model with the accurate prediction and best performance for a demand forecasting system.

Convolutional neural network provides better results on sequence data for developing the relationships and extracting different features of data. Also, other variations of RNN such as Long Short-Term Memory (LSTM) models have the property of detecting long-term and short-term relationships between data features, which is highly needed for demand forecasting problems (Brownlee 2019).

LSTM is one type of Recurrent neural networks. The CNN and RNN “LSTM” are classified as a single type model, and CNN- LSTM is classified as a hybrid model.

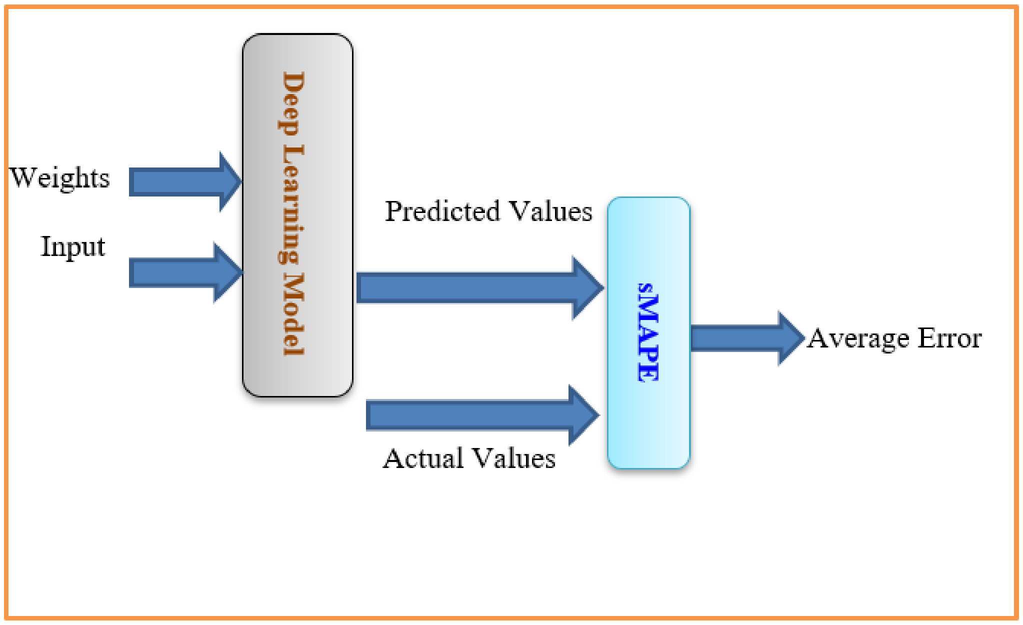


Figure 1: Deep Learning Process.

Each model of these will be trained on stores’ sales dataset, and then the results will be compared for answering questions such as “What is the best single type model?” or “What is the better model for this problem, Hybrid or Single model?” , or ”Which is the best suitable deep learning model for demand forecasting problem?”

1.1 Background

Demand forecasting is the technique for estimating the probable demand for a product. It is based on the past demand for the product and various facts about the product in the market. It plays an important role in any type of businesses, as demand forecasting reduces the risks related to financial aspects and helps to make efficient decisions. Also, the market becomes more competitive, industries are mainly focusing on predictive analytics techniques to reduce the costs and increase their productivity and profit margins (Box et al. 2015; Kilimci et al. 2019).

Over stocking causes increased storage, labor, insurance costs, a decrease in quality, and degradation of the product. On the other hand, Stockouts might affect in loss of sales and reducing customer satisfaction and business trustworthiness. Customers might choose to competitors or buy substitute items if they cannot find the goods that they are looking for. Customers and sales losses are a serious problem for any business. To optimize the stocking and carry fewer financial risks in the retail industry, it is very important to have a precise inventory control system and demand forecasting (McGoldrick and Andre 1997; Kilimci et al. 2019).

Thus, the practice of predictive methods and technological tools are fetching more popularity among the retailer industries, also in various sectors (Grewal, Roggeveen and Nordfalt 2017). Extensive research has been performed in the area of demand forecasting using big data and predictive analytics technologies (Bradlow et al. 2017; Kilimci et al. 2019).

1.2 Concepts

1.2.1 Demand Forecasting

Demand forecasting is a branch of the predictive analytics domain. It is basically a sequence of data points measured at successive intervals in time. Demand forecasting focuses on understanding and predicting customer demand to optimize decisions of business management that are related to supply. Demand forecasting is used in inventory management, production planning, and also sometimes in assessing future capacity requirements, or in making decisions on whether to enter a new market (Demand Forecasting 2019; Kilimci et al. 2019).

1.2.2 Deep Learning

Traditional forecasting methods predict future demand based on historical time-series data related to demand. Deep learning models have been successfully applied to time-series forecasting as these methods can use a huge amount of data, extract features related to the demand, and find patterns using different learning algorithms to predict future demand. Deep learning is a part of machine learning techniques that applies deep neural network architectures to resolve numerous complex problems. Deep Learning models have become highly popular and were recently applied to many fields such as image processing, speech recognition, natural language processing, and machine translation (Bagheri, Gao, and Escalera 2013; Kilimci et al. 2019). Deep learning models have presented better predictions and results when compared with classical time-series algorithms such as K- Nearest Neighbors (kNN), decision trees, and random forest (Kilimci et al. 2019).

1.2.3 Neural Networks

Neural networks(NN) are a class of deep learning algorithms used to understand the complex patterns in datasets by using non-linear activation functions and multiple hidden layers. Neural Networks takes an input, sends it through multiple hidden neurons (with functions and coefficients), and sends the output as prediction representing the combined input of all the neurons (Neural Networks Concepts, Machine Learning Glossary n.d.).

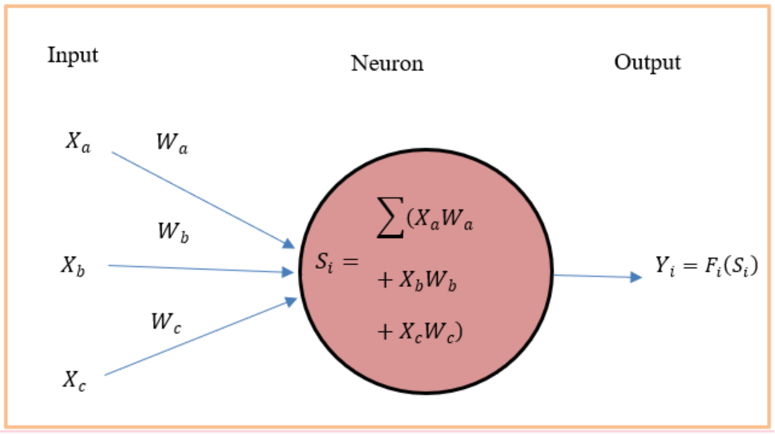


Figure 2: Simple diagram of Artificial Neuron.

In Figure 2, a neuron computes its state by adding all the inputs (X_a, X_b, X_c) multiplying by respective weights (W_a, W_b, W_c), then it applies an activation function ($F_i(S_i)$) which normalizes the result and returns an output(Y_i).

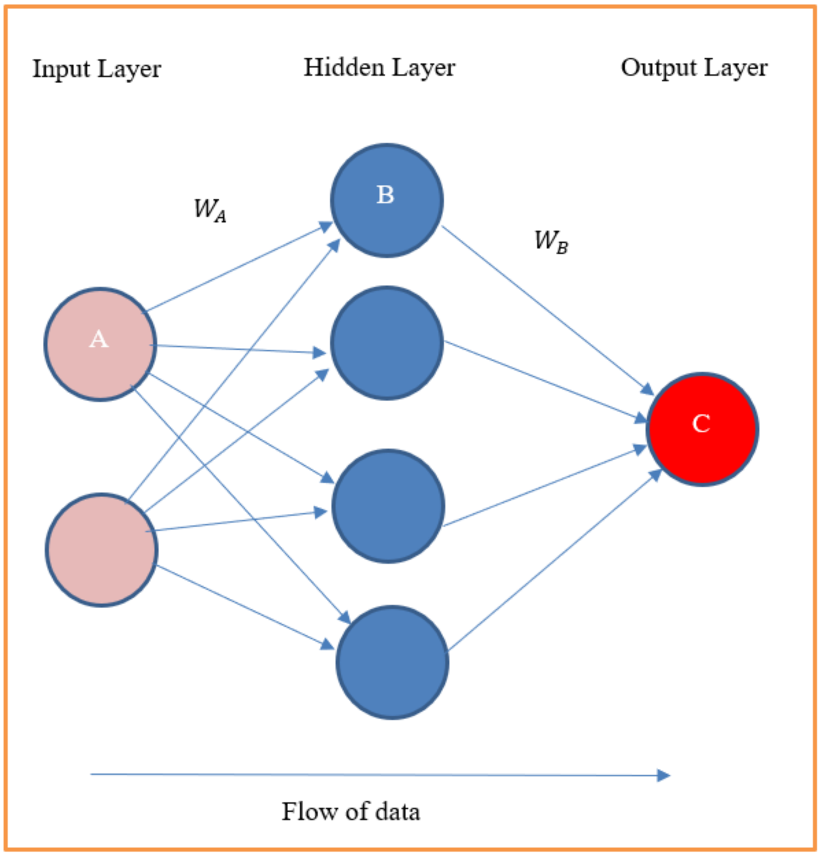


Figure 3: Simple Neural Network.

In Figure 3, the simple architecture of a neural network is shown. Synapses connect inputs to neurons, neurons to neurons, and also neurons to outputs. Every connection between two neurons has a unique synapse with a unique weight attached to it. Every neuron in each layer is connected to every neuron in the next layer to pass the output of the current layer as input for the next layer. The Input layer will receive the input from the loaded dataset, and its output serves as input to the hidden layer. This process repeats until it reaches the final output layer (Mehtha 2019).

Figure 3 shows only one type of neural network. There are several types of neural networks, with each having its unique strengths. In this thesis, we use a neural network such as Convolutional neural networks (CNN), Recurrent neural networks (RNN), Hybrid model (CNN-LSTM).

The rest of this thesis is organized as follow. We survey the related works in Chapter 2. In Chapter 3, we provide an overview of different software that was used in this project. We explain the implementation procedure in Chapter 4. In Chapters 5, 6, 7, we develop CNN, LSTM, Hybrid model (CNN-LSTM), respectively. In Chapter 8, we discuss and analyze the results and the performance of our models. Finally, conclusions are presented in Chapter 9.

Chapter 2

Literature Review

This section summarizes some of the researches about demand forecasting with deep learning. Demand forecasting methods has many areas of application like retail forecasting, stock market, energy load demand, tourism, and transportation. Time-series methods are used by traditional approaches to demand forecasting. Time-series methods include exponential smoothing, damped trend methods, Holt’s linear trend method, exponential trend method, Naïve method, moving averages, Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average methods (ARIMA) (Hyndman, 2018). Among these ARMA and ARIMA are the most popular to give the best predication for the time-series problems (Gujarati 2003; Kilimci et al. 2019).

In the classification problems, the total performance of learning algorithms highly depends on the data representation’s nature (Kilimci et al. 2019; Bengio, Courville and Vincent 2013). Deep learning is an application of artificial neural networks, which mimic the normal human brain (Haykin 2008). Deep Learning uses sequential layers of neurons, where each layer extracts more complex and abstracts features from the output of previous layers. Thus, deep learning can automatically perform feature extraction without any preprocessing step. Time-series analysis, object recognition, and genomics is a few of the fields where deep learning is applied successfully (Bengio 2012; Kilimci et al 2019).

Convolutional neural network is an instance of deep learning and multiple layers of neural networks. In CNN, every layer consists of several two-dimensional planes, and they are composed of several neurons. All the neurons use(share) the same filter in each two-dimensional plane. It results in features tunable parameters, which eventually reduce computation complexity (Qi et al. 2016; Bengio, Courville and Vincent 2013; Kilimci et al. 2019). CNNs are mainly developed for image classification. 1D CNN model is variation of CNN model, this can be applied for one dimensional sequence of data like text data , time series data. Recurrent neural networks, such as Long Short-Term Memory model (LSTM), learns the mapping function while mapping the inputs to outputs and builds the relations between them. Because of this ability to learn long term correlations in a sequence, LSTM is also used in solving time-series problems. Recent literature points out that CNNs can predict sequences in a much faster, more computationally efficient manner than LSTMs (Bai, Kolter and Koltun 2018; So 2019). Also, Hybrid models of the CNNs, RNNs are evolving which are excel at predicting temporal and spatial data. In our study we will use CNN-LSTM as our hybrid model to predict the time series data.

In our study, we develop and compare different models like CNN, LSTM, Hybrid model (CNN-LSTM) to find the model with the accurate prediction and best performance for the demand forecasting system. The details of the comparative study can be found in Section 8.

Chapter 3

Software Overview

3.1 Software Requirement:

Operating System: Windows 10.

Platform: Python 3.7.

IDE: Jupyter Notebook and Google Collaborator.

Libraries: TensorFlow, Keras, Pandas, Scikit-learn, Matplotlib, Numpy, SciPy.

Table 1: Hardware Requirement.

OPERATING SYSTEM	SOFTWARE	PROCESSOR	RAM	Hard disk
Windows	Google Collaboratory, Jupyter Notebooks.	Intel	8GB	100 GB

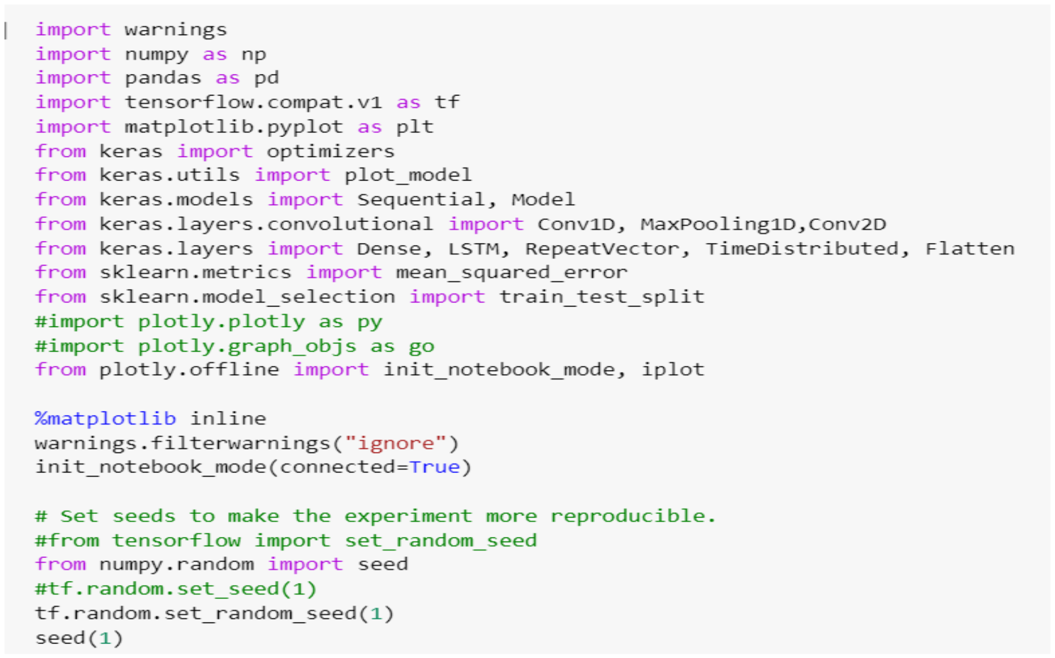
3.2 Python

Python is a high-level, general-purpose interpreted programming language. Guido van Rossum created Python in 1991 at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from several other languages, including Modula-3, ABC, C, C++, SmallTalk, Algol-68, and Unix shell and other scripting languages. Python's design philosophy stresses code readability with its notable use of significant whitespace. Programmers write clear and logical code for small and large-scale projects by Its(python's) language constructs and object-oriented approach.

Python is dynamically typed and garbage-collected. It supports numerous programming paradigms, including procedural, functional, and object-oriented programming. The major strength of the Python is a large library which can be used for the deep Learning, GUI Applications (such as PyQt, Tkinter), Web frameworks like Django (used by Instagram, YouTube, Dropbox, etc.) and Image processing (like Pillow, OpenCV) and many more domains (Dave 2011; Kilimci et al. 2019).

3.3. Libraries

We have used the python libraries for building deep learning models such as TensorFlow, Keras, Pandas, Numpy, Scikit-learn, SciPy, Matplotlib, Plotly and a sample code of imported libraries is shown in Figure 4. In this section, we explain the importance of the libraries we used, as follow:



```
import warnings
import numpy as np
import pandas as pd
import tensorflow.compat.v1 as tf
import matplotlib.pyplot as plt
from keras import optimizers
from keras.utils import plot_model
from keras.models import Sequential, Model
from keras.layers.convolutional import Conv1D, MaxPooling1D,Conv2D
from keras.layers import Dense, LSTM, RepeatVector, TimeDistributed, Flatten
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
#import plotly.plotly as py
#import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot

%matplotlib inline
warnings.filterwarnings("ignore")
init_notebook_mode(connected=True)

# Set seeds to make the experiment more reproducible.
#from tensorflow import set_random_seed
from numpy.random import seed
#tf.random.set_seed(1)
tf.random.set_random_seed(1)
seed(1)
```

Figure 4: Sample code of imported libraries.

3.3.1 NumPy

NumPy is a Python package that stands for 'Numerical Python'. This library is consisting of multidimensional array objects and as well as a collection of routines for processing of array (Roy and Rai n.d.).

NumPy can perform the following operations:

- Logical Operations and Mathematical Operations on arrays.
- NumPy has in-built functions for linear algebra and random number generation.
- Fourier transforms and routines for shape manipulation.

3.3.2 Pandas

Pandas is a highly popular Python library used for data analysis. Pandas is not directly connected to Deep Learning (DL). Before training the model, the dataset must be prepared. In such cases, Pandas comes handy as it was developed, especially for data extraction and preparation. It delivers high-level data structures and a wide variety of tools for data analysis (Roy and Rai n.d.).

Pandas can perform the following operations:

- Reshaping, pivoting, label-based slicing, and sub-setting of large data sets.
- It provides high-performance merging, joining, and group by data for aggregation and transformations.
- It gives time-series functionality, data alignment, and integrated the handling of missing data.

3.3.3 SciPy

SciPy is a highly popular library among deep learning enthusiasts. SciPy contains different modules for optimization, integration, linear algebra, and statistics. There is a variance between the SciPy stack and the SciPy library. It is one of the core packages that makes up the SciPy stack. It is also very useful for image manipulation (Roy and Rai n.d.).

3.3.4 Scikit-learn

Scikit-learn is one of the popular libraries for classical deep learning algorithms. It is developed on top of two basic Python libraries, which are SciPy and NumPy. This library supports both unsupervised and supervised learning algorithms. Scikit-learn can also be used for data analysis and data-mining (Scikit-learn n.d.; Roy and Rai n.d.).

3.3.5 TensorFlow

TensorFlow is most of the popular open-source libraries for high-performance numerical computation. The Google Brain team developed it in Google. TensorFlow involves defining and running computations with the involvement of tensors. TensorFlow can train, test, and run deep neural networks which used to develop numerous AI applications. Deep learning research and applications widely use TensorFlow (Roy and Rai n.d.).

3.3.6 Keras

Keras is a highly popular deep learning library for Python. Keras is a high-level neural networks API. Keras is also capable of running on top of Theano, TF, and CNTK. Keras can be run seamlessly on GPU and CPU. By using Keras, beginners to deep learning can design and build a neural network. Easy and fast prototyping are a few of the best things that Keras provide (Roy and Rai n.d.).

3.3.7 Matplotlib

Matpoltlib is most of the popular Python libraries for data visualization. Like Pandas, Matplotlib is not directly related to deep learning. It is very handy when a computer programmer needs to visualize a few kinds of patterns in the data. Matplotlib is a 2D plotting library mainly used for generating 2D plots and graphs. PyPlot is a module in this library, which makes it easy for plotting because it has features to font properties, formatting axes, control line styles, etc. It provides several kinds of plots and graphs for data visualization, such as histogram, error charts, bar charts, etc., (Roy and Rai n.d.).

3.3.8 Plotly

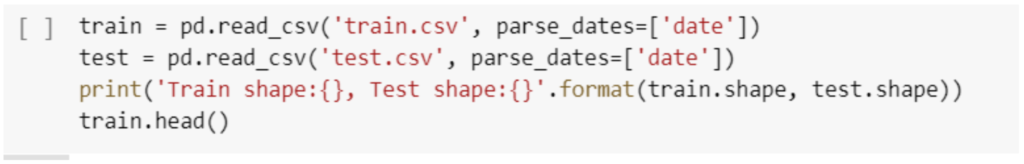
Plotly is one of the famous Python libraries that is used to design graphs, like interactive graphs. Plotly can plot several charts and graphs like histogram, boxplot, barplot, spread plot, etc. Plotly is often used in data analysis and financial analysis. It is popular as an interactive visualization library. Plotly is connected with pandas to create charts and graphs of dataframes directly with the help of cufflink**.** Choropleth is in the plotting of geographical locations around the world (Roy and Rai n.d.).

Chapter 4

Implementation

4.1 Input

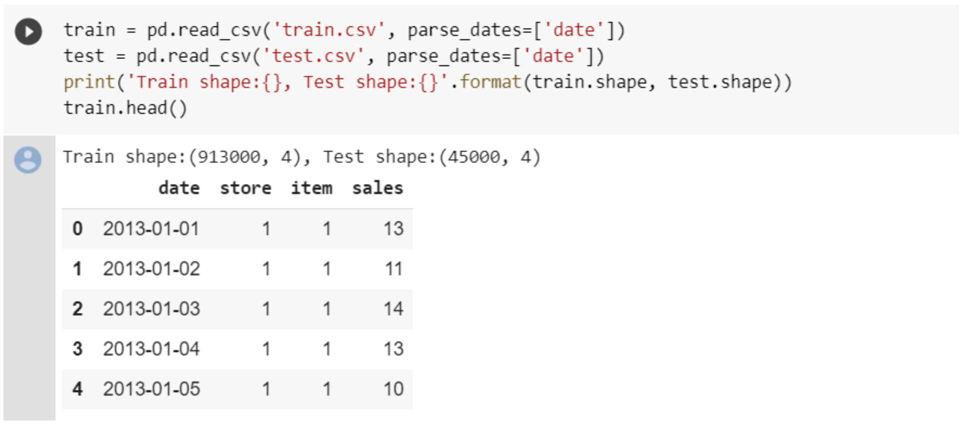
The CSV file is a delimited text file in which a comma separates the values(tabular data) in the columns. Each line of the CSV file is a data record. Each record contains one or more fields, separated by commas. In deep learning, reading data from CSV(comma separated values) is important. A CSV file can be read in full as well as in parts for only a selected group of columns and rows by using the panadas library. The sample code of loading data sets is shown in Figure 5.



```
[ ]  train = pd.read_csv('train.csv', parse_dates=['date'])
     test = pd.read_csv('test.csv', parse_dates=['date'])
     print('Train shape:{}, Test shape:{}'.format(train.shape, test.shape))
     train.head()
```

Figure 5: Sample code of loading datasets.

We use a relatively simple and clean dataset, which includes the retail stores' real-life sales and stock data with four features (Store ID, Product ID, Date, Sales)The dataset consists of 260 weeks of sales data that includes fifty distinct products for ten different stores. The sample code extracting features is shown in Figure 6.



```
train = pd.read_csv('train.csv', parse_dates=['date'])
test = pd.read_csv('test.csv', parse_dates=['date'])
print('Train shape:{}, Test shape:{}'.format(train.shape, test.shape))
train.head()
```

```
Train shape:(913000, 4), Test shape:(45000, 4)
         date  store  item  sales
0  2013-01-01      1     1     13
1  2013-01-02      1     1     11
2  2013-01-03      1     1     14
3  2013-01-04      1     1     13
4  2013-01-05      1     1     10
```

Figure 6: Sample code of extracting features.

This dataset will explain the variance of sales of each product at each store, along with the Time features included in the Date feature.

And for each id, several unique numbers found, this is as follows:

Table 2: Features and Unique Values.

Feature	**Number of Unique values**
Store ID	10 Stores
Product ID	50 Products

As we see, the data is balanced, each item in each store has its data daily from 2013 till the end of 2017, so each item has the same number of recordings in the dataset we have. This means it can be used to train a deep learning model without over fitting one product to another.

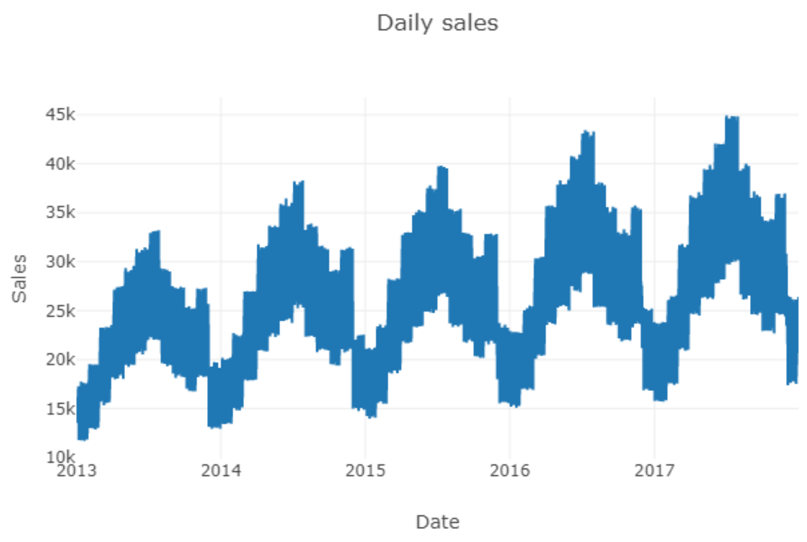


Figure 7: Daily Sales.

4.2 Procedures

The Data is stored in the Pandas data frame, and then the following procedures are done:

- Tidying data
- Cleaning data
- Feature Extraction
- Dataset storage

4.2.1 Tidying Data

Processing the dataset so we can make sure that every record is a unique set of data, having a unique store_id and product_id and date.

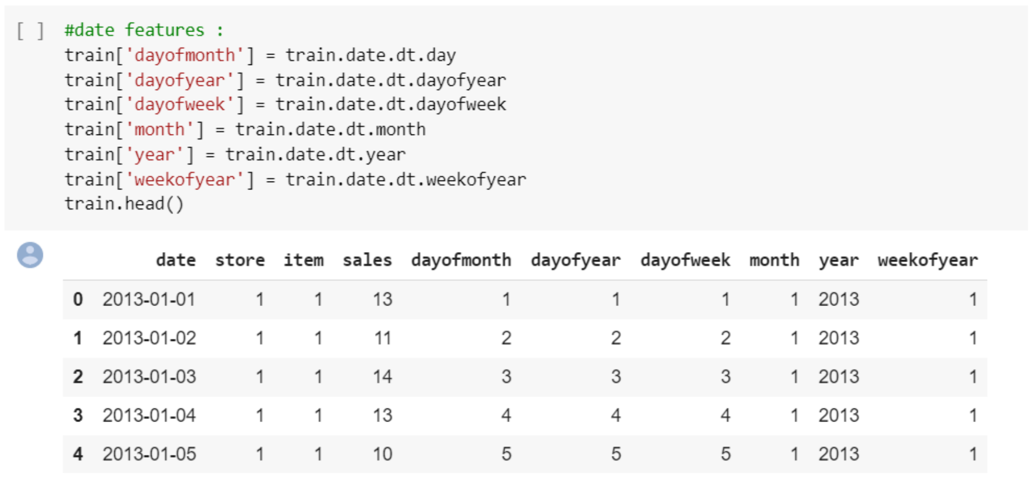
4.2.2 Cleaning Data:

Check for missing data, bad input data, data types, and clean all of them so we can make sure we can perform our feature extraction on a solid base, to have a convenient dataset

4.2.3 Feature Extraction:

For time-series data, the most work done is to detect what feature can be added to help the model predict the target. Extracting Date features and make them input features to the model. This involved extracting all date properties from the DateTime object the dataset having, like

- Day of month
- Day of week
- Week of year
- Week of month
- Day
- Month
- Year



```
[ ] #date features :
    train['dayofmonth'] = train.date.dt.day
    train['dayofyear'] = train.date.dt.dayofyear
    train['dayofweek'] = train.date.dt.dayofweek
    train['month'] = train.date.dt.month
    train['year'] = train.date.dt.year
    train['weekofyear'] = train.date.dt.weekofyear
    train.head()
```

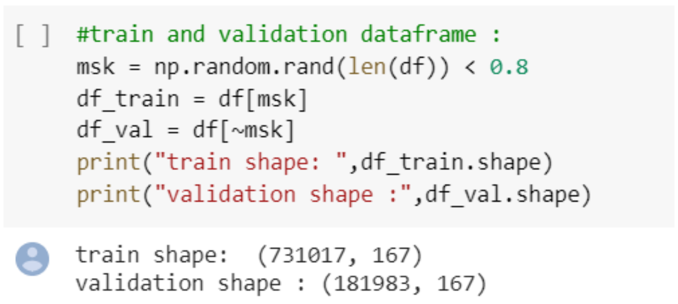
	date	store	item	sales	dayofmonth	dayofyear	dayofweek	month	year	weekofyear
0	2013-01-01	1	1	13	1	1	1	1	2013	1
1	2013-01-02	1	1	11	2	2	2	1	2013	1
2	2013-01-03	1	1	14	3	3	3	1	2013	1
3	2013-01-04	1	1	13	4	4	4	1	2013	1
4	2013-01-05	1	1	10	5	5	5	1	2013	1

Figure 8: Getting the deriving features from the data.

Figure 8 shows the sample code of extracting features from data. Then feeding them to the neural network this will let the neural network check for the time-related trends and variations that can explain many customer behaviors according to time.

4.2.4 Dataset Storage

The dataset now is ready to be used, so it is stored in pandas dataframe, and then to a CSV file so it can be accessed again. Figure 9 shows the splitting of data into train and test sets.



```
[ ] #train and validation dataframe :
    msk = np.random.rand(len(df)) < 0.8
    df_train = df[msk]
    df_val = df[~msk]
    print("train shape: ",df_train.shape)
    print("validation shape :",df_val.shape)
```

```
train shape:  (731017, 167)
validation shape : (181983, 167)
```

Figure 9: Splitting into the train, test sets.

4.3 Hyperparameters

Hyperparameters are the parameters whose values are set before beginning of the learning process. These parameters determine the network structure (Number of hidden layers, filters) and how network is trained (Learning rate). More number of layers, the more complex dependencies a neural network can recognize. Choosing these parameters correctly is very important for developing a efficient neural network (Brownlee 2018a).

4.3.1 Activation function

Each neuron in our model has an activation function at its core. The activation function of a neuron defines the output of that neuron given an input or set of inputs. Linear activation function may work well, but nonlinear activation functions can handle the complex relationships in the data. In our thesis, we are using a rectified linear activation unit (ReLU). ReLU is acting as a linear function as well as a non-linear function (Brownlee 2018a).

4.3.2 Optimizer

Optimizers are methods or algorithms which are used to update the attributes of the neural network such as learning rate and weight parameters in order to reduce the loss. The loss function guides the optimizer if it is optimizing to get the global minimum.

In our study, we are using Adam optimizer. Adam is Adaptive Moment Estimation. Adam optimizer is one of the most popular gradient descent optimization algorithms. It calculates the individual adaptive learning rate for each parameter from estimates of the first and second moments of the gradients. Adam uses the exponential moving average of the gradients for scaling the learning rate. It is computationally efficient and also has less memory requirements (Doshi 2019; Khandelwal 2019).

4.4 Evaluation Metric

4.4.1 sMAPE

Neural networks are trained using optimization algorithms, which requires a loss function to calculate the model error (Brownlee 2018b). Forecast accuracy in our thesis is measured using sMAPE (Symmetric Mean Absolute Percentage Error). So the least amount of loss indicates that the model is better. Statistically, sMAPE is the average of percentage errors. It is more precise and faster when compared to the other loss functions. sMAPE is also better in protecting from outliers and bias effect.

$$sMAPE = \frac{\sum_{t=1}^{n} |A_t - F_t|}{\sum_{t=1}^{n} (A_t + F_t)}$$

where A_t is the actual value, and F_t is the forecast value.

The absolute difference between A_t and F_t is divided by the sum of absolute values of the actual value A_t and the forecast value F_t. The value of this calculation is summed for every fitted point t and divided again by the number of fitted points n (sMPAE, Wikipedia 2018).

Chapter 5

Convolutional Neural Network (CNN)

It would seem simple for humans to identify numbers, items in a picture and to check for the relationship between the sequence of features, but how do you train the machine to recognize these different features in any block of data? Convolutional neural networks (CNN) can solve this problem. In this Section, a convolutional neural network will be used and fitted on the time-series dataset to predict the sales of a product within certain stores at a certain date. Obviously, human beings can perceive that there is a hierarchy or conceptual structure in such data, and sometimes they cannot, and they can miss, this is called Features in the field of Feature Engineering, these features can be extracted from Convolution Neural Networks because they have feature extraction layers at the beginning of these structures, that can detect those relationships(Brownlee 2019; Pokhrel 2019). So now, we will try to see how to do this using Convolutional Neural Network.

In recent years, the development of neural networks has been extremely rapid in the field of pattern recognition systems. We use a common pattern recognition technique in this article and use convolution to create this technique. This technique will be using the fact of having relationships between the features and itself, also the features and the target.(Mehtha 2019; Mikhailouskaya 2019)

It is achieved by operating directly on raw data, as an alternative of domain-specific or handcrafted features derived from the raw data. The CNN model then learns how to automatically extract the features that are directly useful for the problem being addressed. This way of learning is known as representation learning, and CNN achieves this in such a way that the features are extracted regardless of how they occur in the data, so-called transform or distortion invariance. CNN can learn and automatically extract features from raw input data. A sequence of observations can be treated like a one-dimensional image that a CNN model can read and distill into the most salient elements (Brownlee 2019). In our study we are using 1D CNN model which is variation of CNN model. 1D CNN model is for one dimensional sequence of data like text data, time series data. The difference between 1D CNN model and Standard CNN model is the structure of the input data and how the filter (feature detector) moves across the data. In the rest of the thesis we address 1D CNN model as CNN model.

5.1 Convolutional Neural Network Architecture

A CNN consists of:

- Convolutional Layers
- Processing Layers
- Fully Connected Layers

Convolutional Layer is a layer of units, each unit is considered a feature extractor kernel, this kernel is used to detect certain correlations between certain features of an image, and to explain the variance of the output using the variance of the features, they mask the image, to other variations of itself.

Processing Layer is:

- Pooling Layer
- Flattening Layer

Pooling Layer is used to reduce the size of data, to exclude unnecessary and unused data.

Flattening Layer is used to change the dimensions of the image from two-dimensional to one-dimensional vector, to be able to enter the Fully Connected Layer.

Fully Connected Layers are then activated, fed with the output of the feature extraction process, and then the classification mechanism begins as in the artificial neural networks.

For this, Convolutional Neural Network will be used with the following architecture.

5.2 Structure of the Model

- Convolutional layer having
  - 64 filters
  - Kernel size of 2
  - Activation function – Relu
- Max pooling layer having
  - Pool size = 2
- Convolutional layer having
  - 128 filters
  - Kernel size of 2
  - Activation function – Relu
- Max pooling layer having
  - Pool size = 2
- Flattening layer
- Fully connected layers having
  - 50 nodes

        - Activation function – Relu
    - Output layer having
        - 1 node
    - Compiled using
        - Loss function – sMAPE
        - Optimizer - Adam

5.3 Results and Graphs

In Table 3, as we can see the training loss, validation loss and time elapsed for the CNN model. Figure 10 shows the loss value during the training and testing phases. In Figures 11, 12, 13, we can see the comparison of actual values and predicted values of total sales, total sales per store, total sales per item respectively.

Table 3: Results of CNN.

Training loss	Validation Loss	Time Elapsed for training	PC
6.3447	6.3612	379 seconds	Google Colab CPU

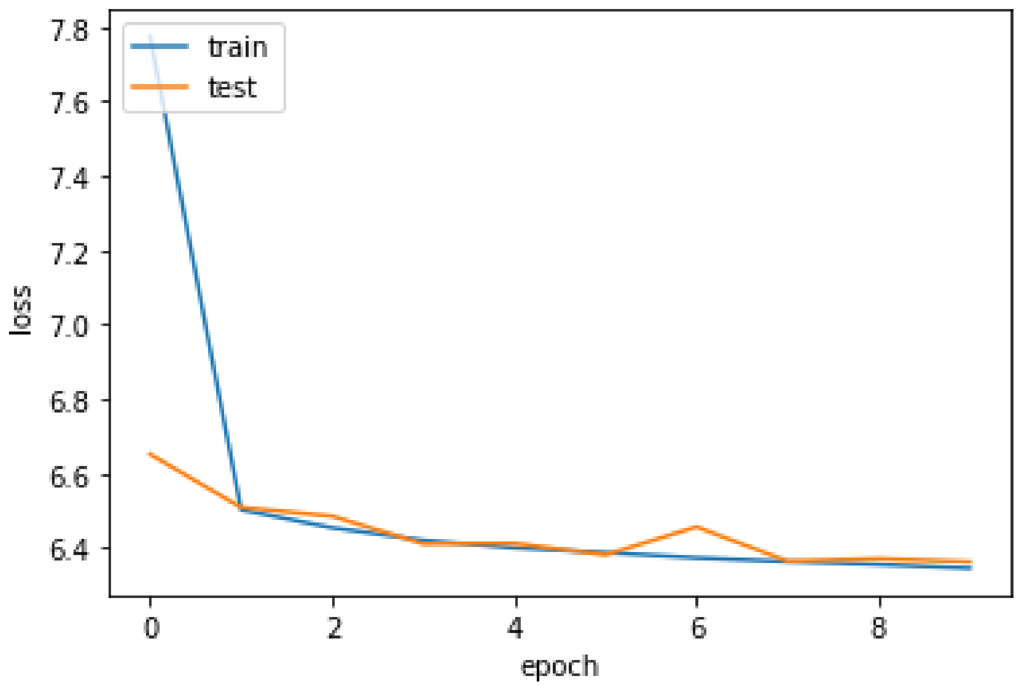


Figure 10: CNN- Training Loss vs. Validation Loss.

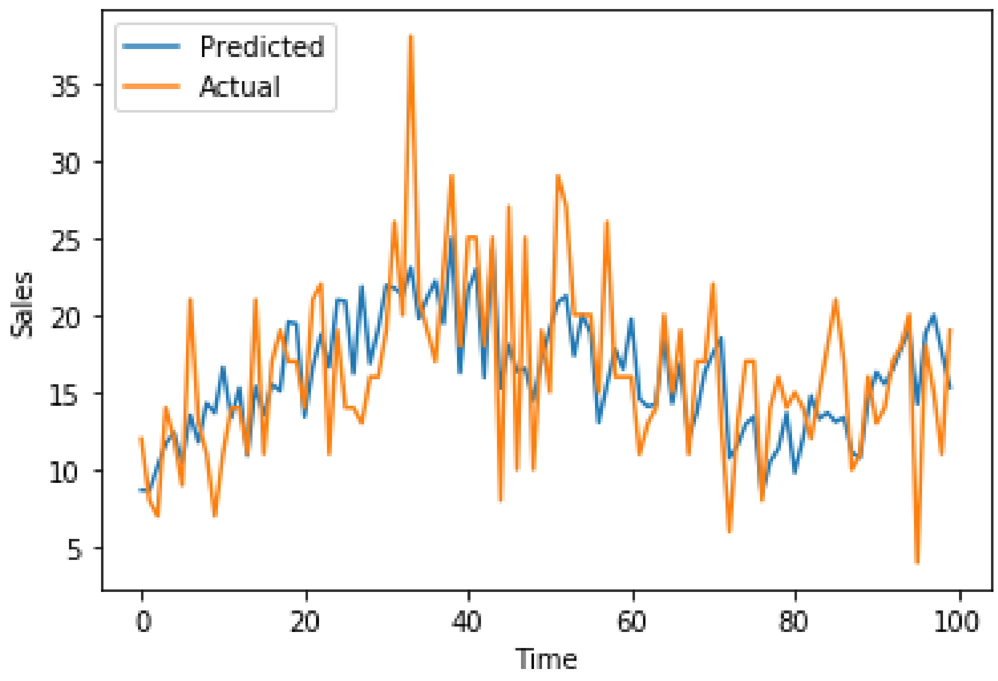


Figure 11: CNN- Predicted Sales vs. Actual Sales.

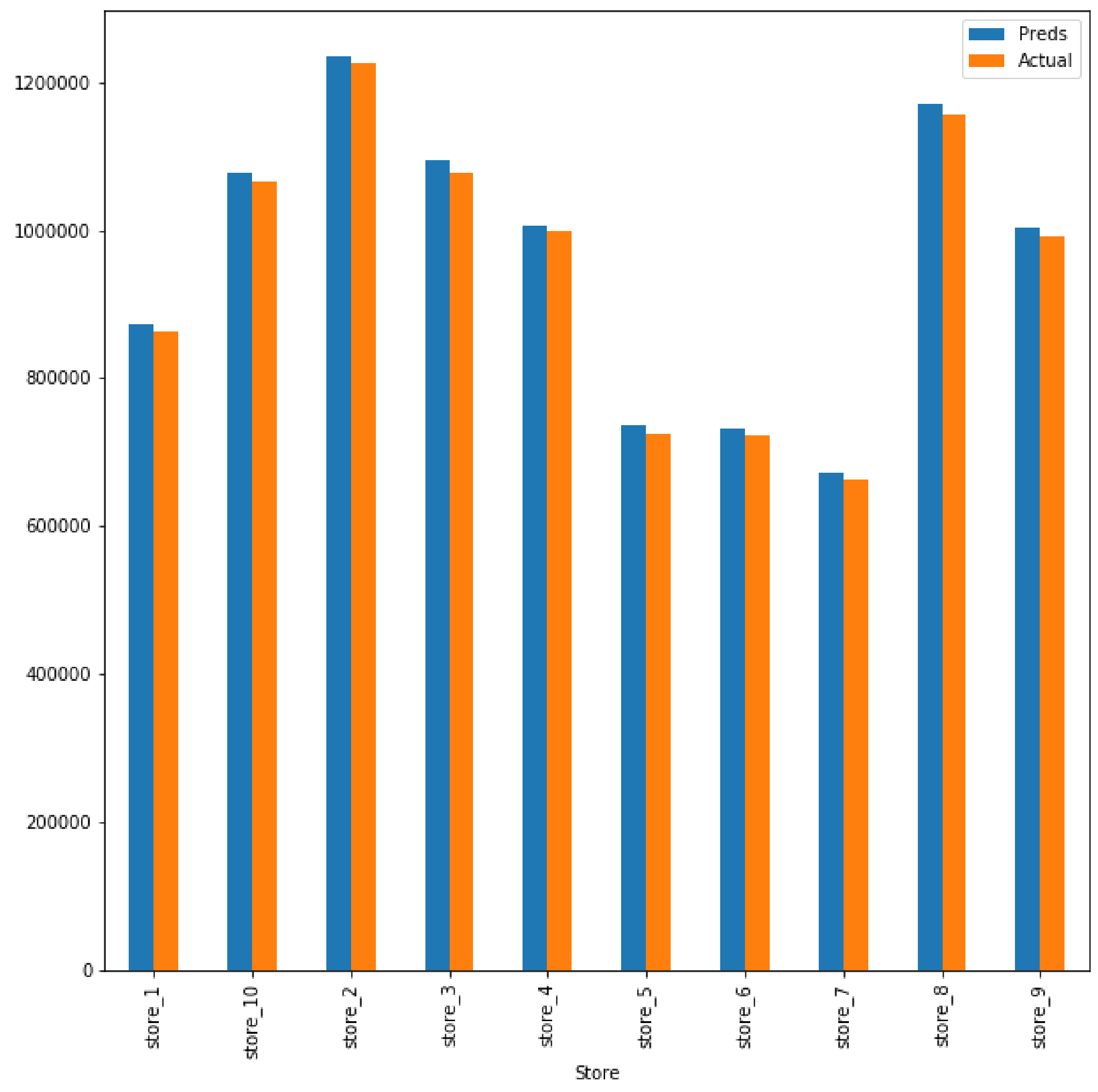


Figure 12: CNN – Predicted Sales Per Store vs. Actual Sales Per Store.

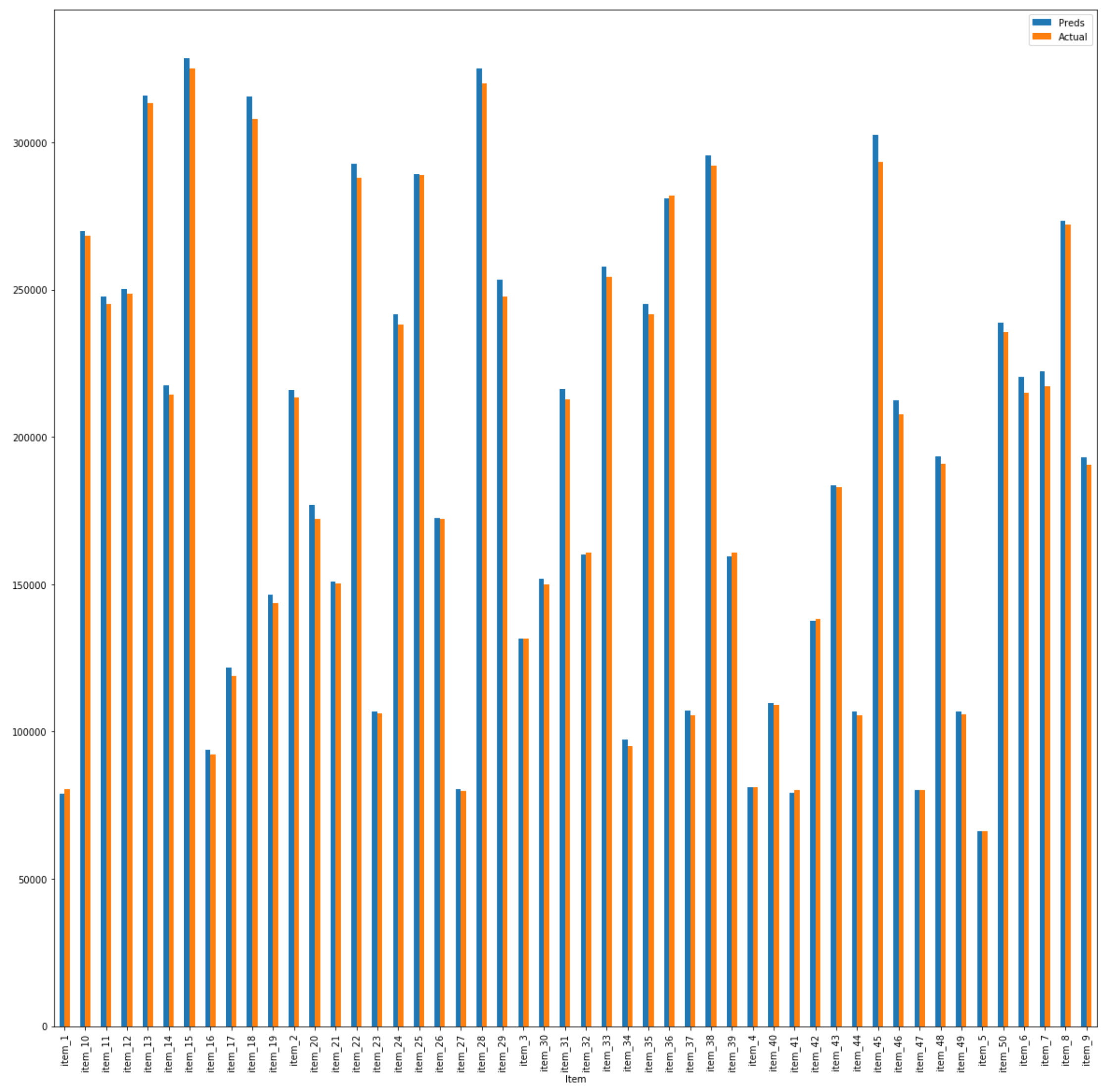


Figure 13: CNN – Predicted Sales Per Item vs. Actual Sales Per Item.

Chapter 6

Long Short-Term Memory (LSTM)

In recent years, there were many problems regarding the Neural networks, like the exploding gradient problem. This problem led the scientists to develop a new type of Neural Networks, which is Recurrent Neural Networks. These networks are found to be having the ability to acquire the relationships on a certain window of time, depending on the type of data it is fed (Kilimci et al. 2019).

LSTM (Long Short-Term Memory) was implemented as a variations of Recurrent Neural Networks architectures, LSTM is considered a very powerful tool to detect the Long and short-term relationships among a dataset, and to exclude some relations when needed, and then it has a very high predictive power when it comes to time-series predictions and modeling. Although CNNs can learn arbitrary mapping functions, LSTM offers efficiency and greater performance for automatically learning the temporal dependencies (Mikhailouskaya 2019; Brown lee 2019).

6.1 Recurrent Neural Network Architecture

An RNN consists of:

- Recurrent Neural Layers (LSTM or RNN)
- Fully Connected Layers

RNN layer is a layer of units, each unit is considered as a RNN cell, and each layer has the property to have a returning sequence or not to, which should be for all layers except the last layer before the Fully Connected layer.

Fully Connected Layers are then activated, fed with the output of the RNN layers process, and then the prediction mechanism starts as in the artificial neural network.

For this, the Recurrent Neural Network will be used with the following architecture

6.2 Structure of the Model

- Recurrent Neural Layer having
    - LSTM configuration
    - Fast CUDNN implementation
    - 100 units
    - Return sequences enabled
    - Relu activation function
- Recurrent Neural Layer having
    - LSTM configuration
    - Fast CUDNN implementation
    - 50 units
    - Return sequences disabled
    - Relu activation function
- Output layer having
    - 1 node
- Compiled using
    - Loss function – sMAPE
    - Optimizer - Adam

6.3 Results and Graphs

In Table 4, as we can see the training loss, validation loss, and time elapsed for the LSTM model. Figure 14 shows the loss value during the training and testing phases.

In Figures 15, 16, 17, we can see the comparison of actual values and predicted values of sales, sales per store, sales per item respectively.

Table 4: Results of the LSTM model.

Training loss	Validation Loss	Time Elapsed for training	PC
7.8595	6.9254	3965 seconds	Google Colab CPU

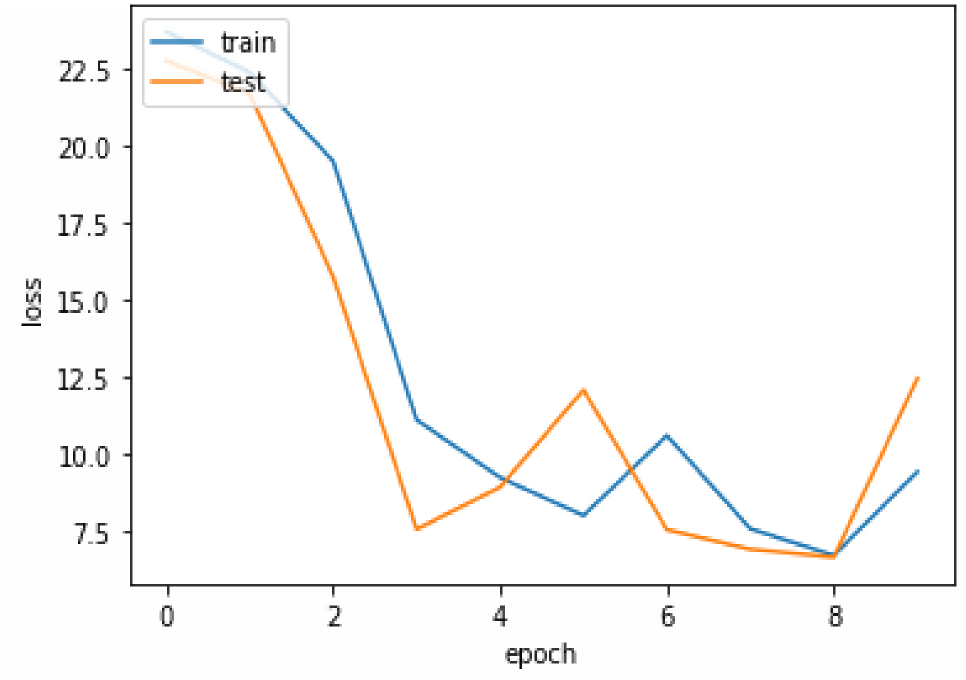


Figure 14: LSTM- Training Loss vs. Validation Loss.

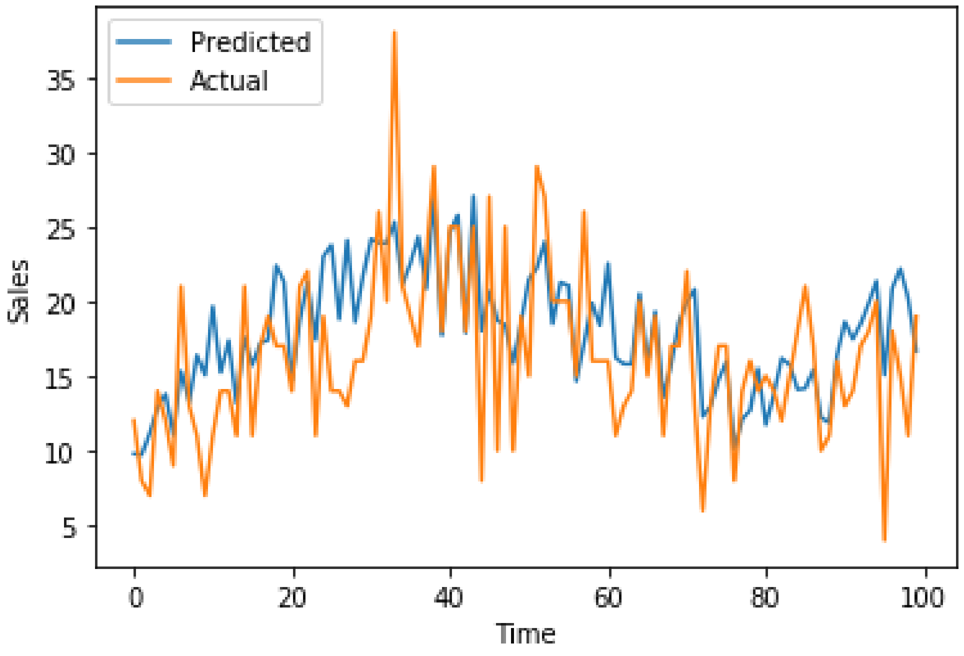


Figure 15: LSTM- Predicted Sales vs. Actual Sales.

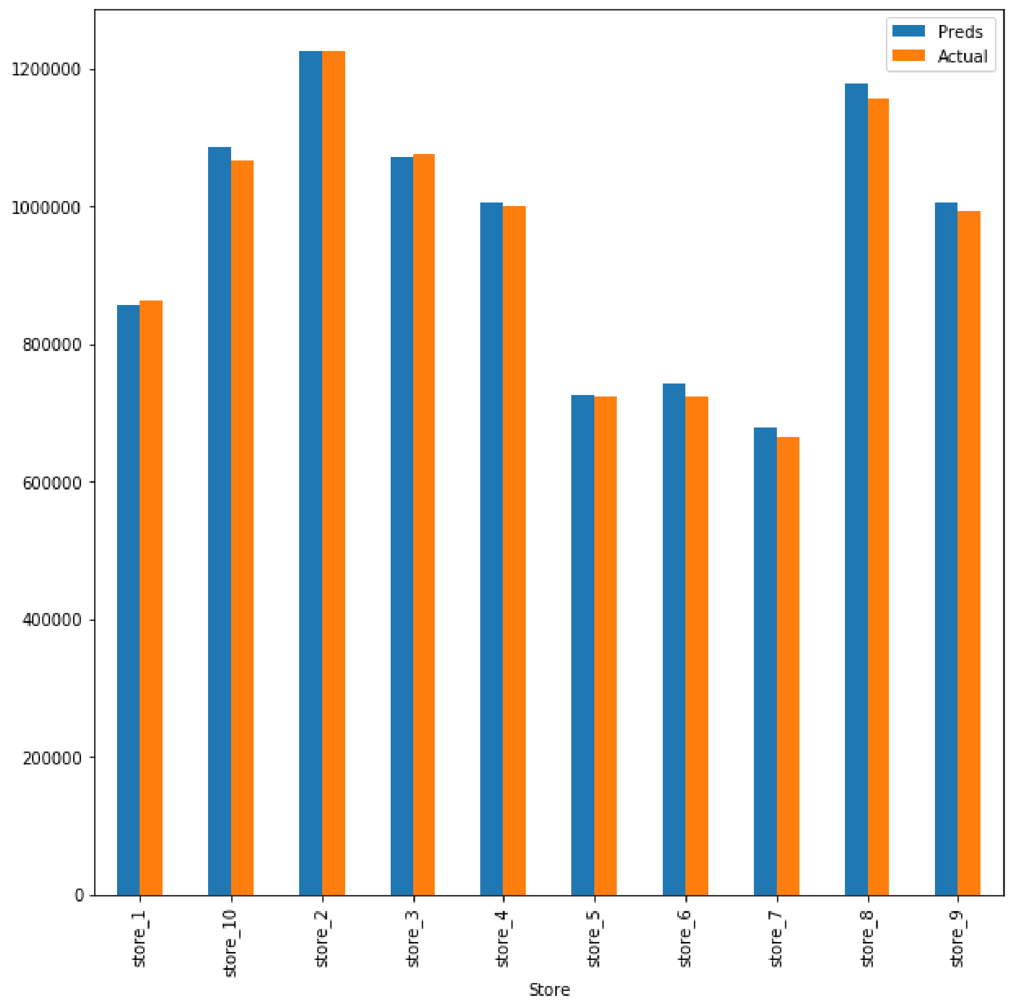


Figure 16: LSTM – Predicted Sales Per Store vs. Actual Sales Per Store.

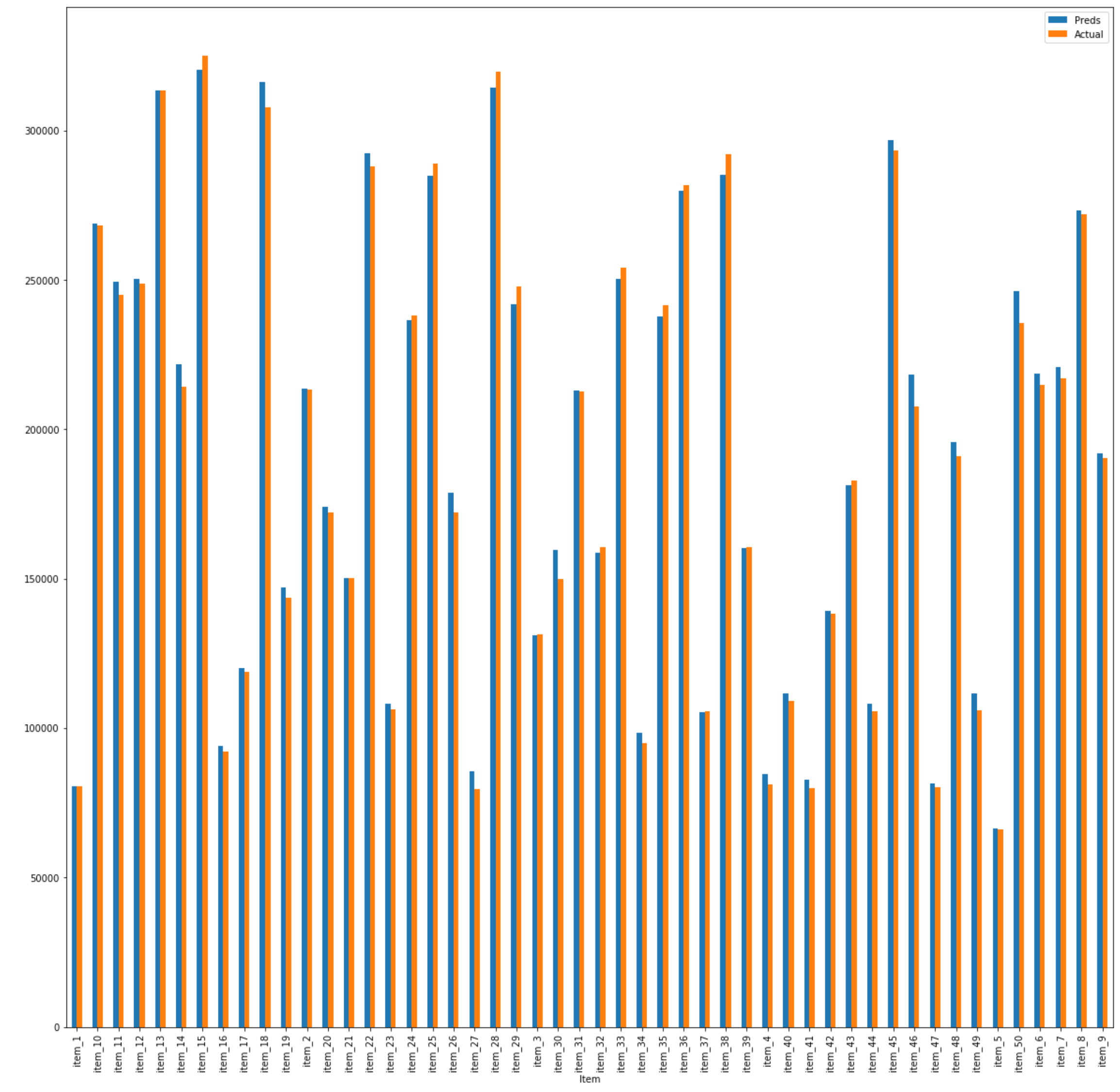


Figure 17: LSTM – Predicted Sales Per Item vs. Actual Sales Per Item.

Chapter 7

Hybrid Model (CNN-LSTM)

Lately, the idea of having a hybrid model has been rising because of our requirements like feature extractors and time-series properties. So as we stated in previous sections, CNN is used as a feature extraction neural network that can feed into fully connected neural network helping it acquiring relationships. Also RNN (“LSTM”) is used as a time-series predicting model since it can extract the time-dependent relationships. So what if we use both to detect the features and the time-series dependent properties?

From this, the hybrid (CNN-LSTM) model was developed. It is used for many projects like OCR (Optical Character Recognition), License Plate recognition, ID Card text extraction, Prices Predictions, and Forecasting. So, we will use the hybrid model to tackle demand forecasting and then compare it among the single type model.

7.1 Hybrid Neural Network Architecture

A CNN-LSTM consist :

- Convolutional Neural Network
- Processing layers
- Recurrent Neural Layers (LSTM)
- Fully Connected Layers

Convolutional Layer is a layer of units, and each unit is considered a feature extractor kernel, this kernel is used to detect certain correlations between certain features of an image, and to explain the variance of the output using the variance of the features, they mask the image, to other variations of itself.

Pooling Layer is used to reduce the size of data, to exclude unnecessary and unused data.

Flattening Layer is used to change the dimensions of the image from a two-dimensional to a one-dimensional vector.

RNN layer is a layer of units, each unit is considered an RNN cell, and each layer has the property to have a returning sequence or not to, which should be for all layers except the last layer before the Fully Connected layer.

Fully Connected Layers are then activated, fed with the output of the feature extraction process, and then the classification mechanism begins as in the ANN.

7.2 Structure of the Model

- Convolutional layer is having
  - 64 filters
  - Kernel size of 2
  - Activation function – Relu
- Max pooling layer is having
  - Pool size = 2
- Convolutional layer is having
  - 128 filters
  - Kernel size of 2
  - Activation function – Relu
- Max pooling layer is having
  - Pool size = 2
- Recurrent Neural Layer is having

        - LSTM configuration
        - Fast CUDNN implementation
        - 50 units
        - Return sequences enabled
        - Relu activation function
    - Output layer having
        - 1 node
    - Compiled using
        - Loss function – sMAPE
        - Optimizer – Adam

7.3 Results and Graphs

In Table 5, as we can see the training loss, validation loss, and time elapsed for the CNN-LSTM model. Figure 18 shows the loss value during the training and testing phases. In Figures 19, 20, 21, we can see the comparison of actual values and predicted values of sales, sales per store, sales per item respectively.

Table 5: Results of CNN-LSTM.

Training loss	Validation Loss	Time Elapsed for training	PC
6.4015	6.3665	617 seconds	Google Colab CPU

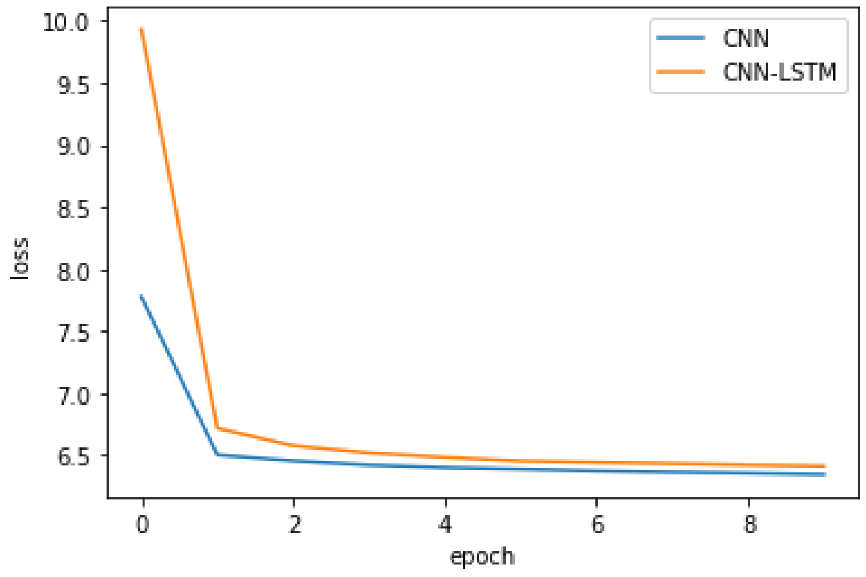


Figure 18: CNN-LSTM vs. CNN- Training Loss.

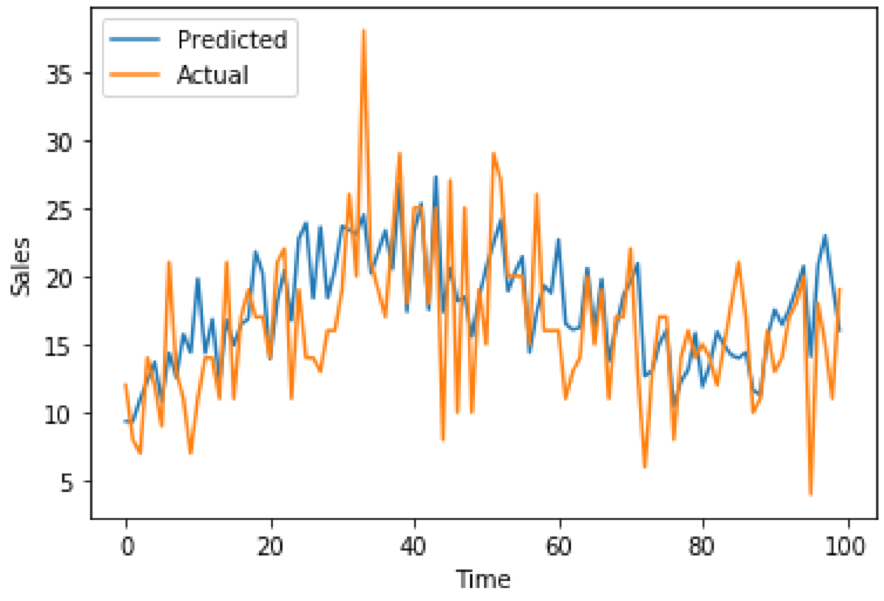


Figure 19: CNN-LSTM - Predicted Sales vs. Actual Sales.

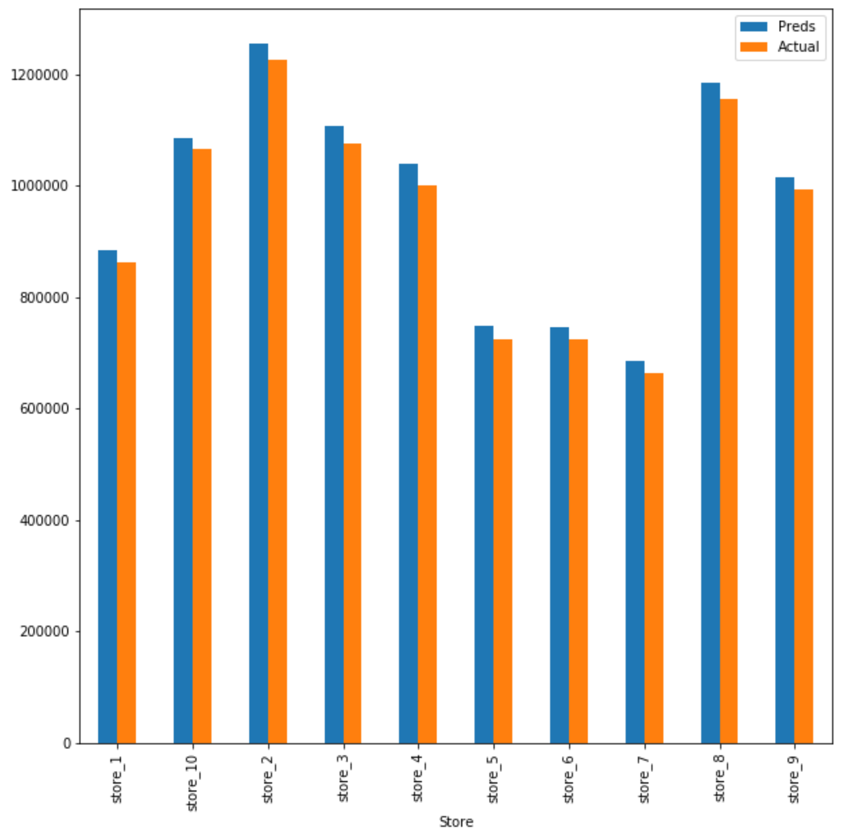


Figure 20: CNN-LSTM – Predicted Sales Per Store vs. Actual Sales Per Store.

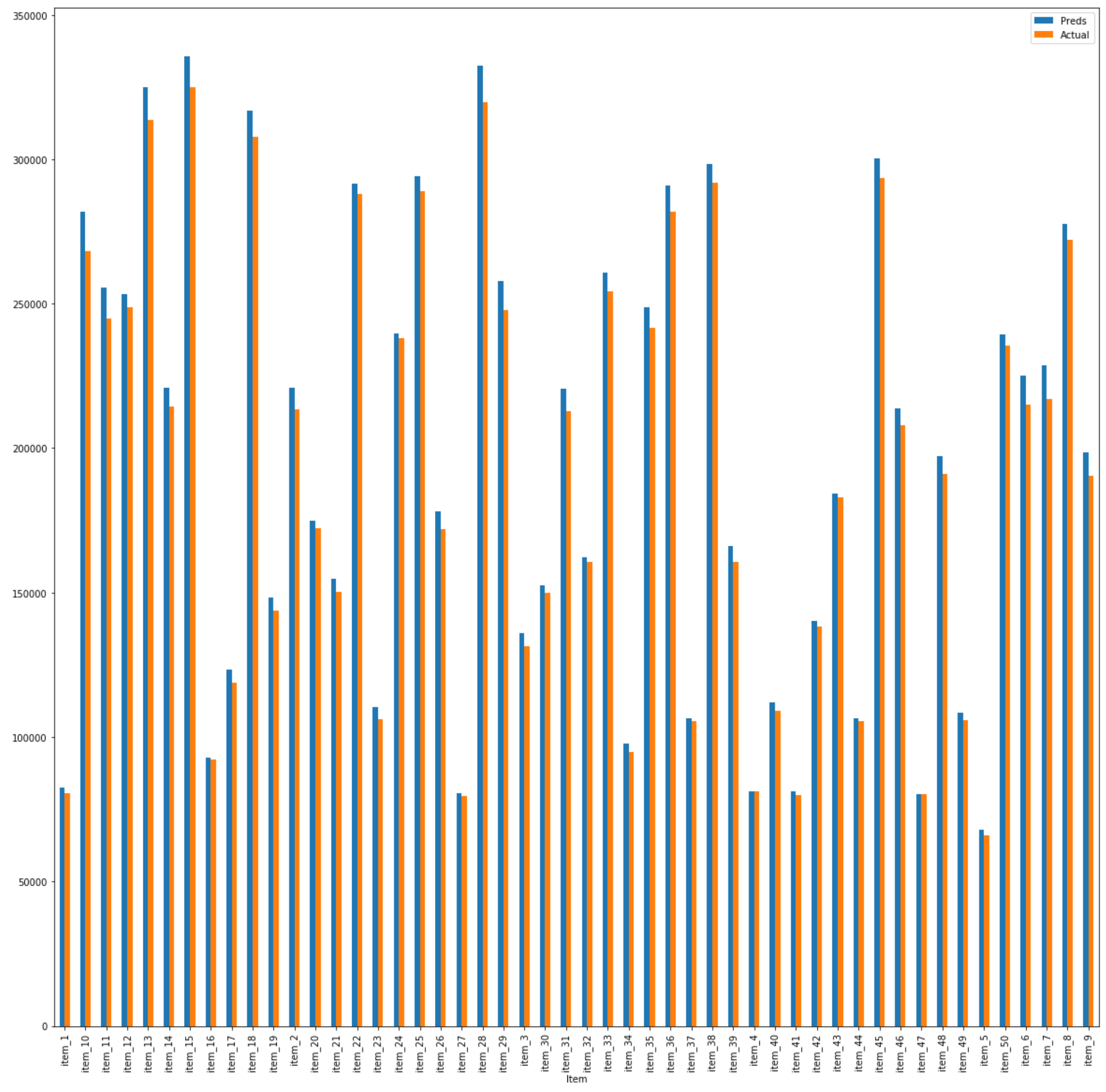


Figure 21: CNN-LSTM – Predicted Sales Per Item vs. Actual Sales Per Item.

Chapter 8

Results and Discussion:

As a result, the inclusion of a deep learning approach to the time-series problem provides solution for demand forecasting. In our study, the forecast accuracy is measured using the sMAPE loss function. The least amount of loss indicates that the model is better. We observe the following results:

In the training phase, the error of CNN is 6.344, LSTM is 7.852, and CNN-LSTM is 6.401. In the validation phase, the error of CNN is 6.361, LSTM is 6.925, and CNN-LSTM is 6.366. As we can see, CNN has the least error value, CNN-LSTM has a second-best error value, and LSTM has the highest error value among these three models.

We also observe that the time elapsed for the training of CNN is 379 seconds, for the CNN-LSTM (hybrid) model is 671 seconds, and for LSTM is 3965 seconds. As shown CNN has the best elapsed time.

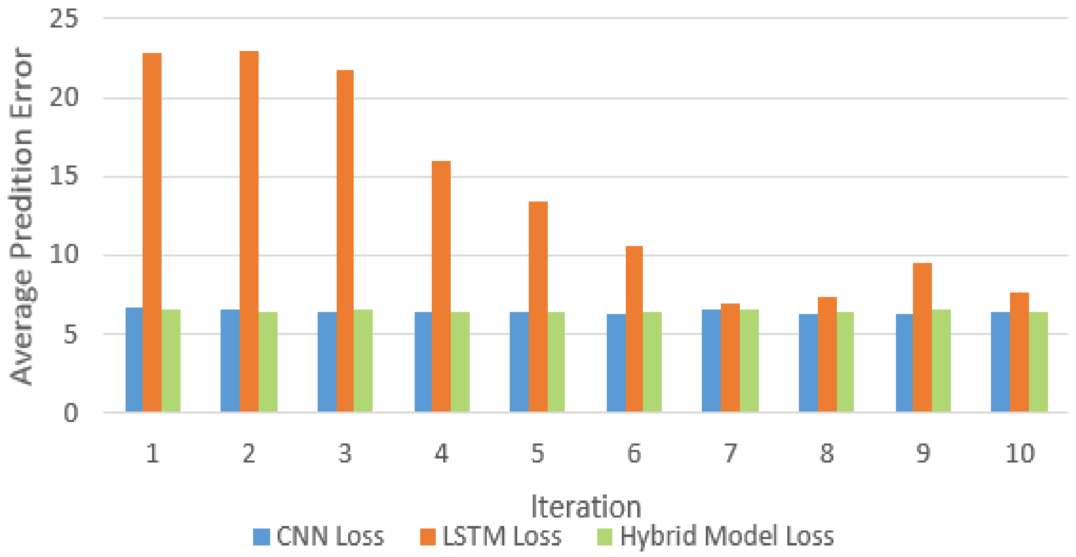


Figure 22: Comparison of Average Prediction Error Value of the Three Models.

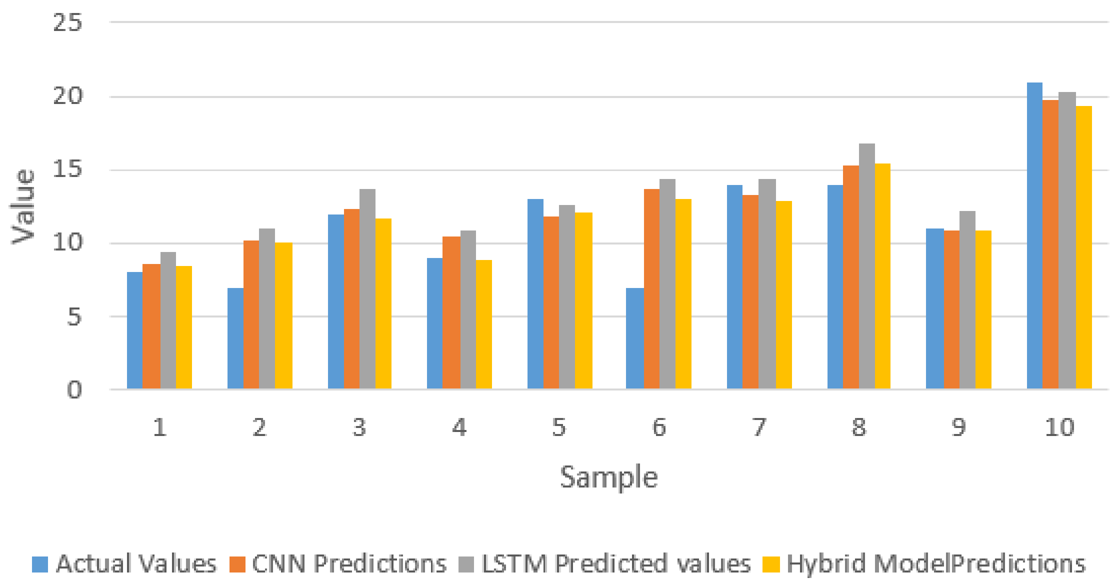


Figure 23: Comparison of Predicted and Actual Values of the Three Models.

As we can see, Figure 22 shows that CNN is better than LSTM and CNN-LSTM in predicting target sales values faster and precise.

CNN-LSTMs were developed for visual time series prediction problems and the application of generating textual descriptions from sequences of images, so it is not performing better for tabular time series data. Also, Data set size is a very crucial part of training neural networks. Larger datasets help to learn the model and its parameters better, it improves the optimization process. Also, Figure 23 and Table 6 show the samples of actual and predicted values by CNN, LSTM, and CNN-LSTM.

Table 6: Comparison of Predicted and Actual Values of the Three Models.

	CNN		LSTM		CNN-LSTM	
S.no	Predicted	Actual	Predicted	Actual	Predicted	Actual
1	8.646827	8	8.47835	8	9.328511	8
2	10.19299	7	10.09834	7	10.97618	7
3	12.39516	12	11.63054	12	13.67024	12
4	10.47596	9	8.899753	9	10.82941	9
5	11.78841	13	12.13036	13	12.5747	13
6	13.66984	7	13.0495	7	14.39933	7
7	13.33816	14	12.82147	14	14.34704	14
8	15.26849	14	15.41805	14	16.83186	14
9	10.91612	11	10.80813	11	12.24012	11
10	19.74852	21	19.36611	21	20.22315	21

Chapter 9

Conclusion and Future Work

9.1 Conclusion

In this study, we applied deep learning models to address problems of demand forecasting such as to optimize the stocking, carry fewer financial risks, have a precise inventory control system, increase the sales, and customer loyalty. We have developed forecasting models using neural networks and applied them to a sales data set of retail stores. Our deep learning approaches analyze, learn complex interactions, and patterns from historical data.

We observe that:

- CNN provides 6.36% sMAPE average prediction error.
- LSTM provides 6.92% sMAPE average prediction error.
- CNN-LSTM provides 6.36% sMAPE average prediction error.

CNN and CNN-LSTM perform very closely in terms of loss, but CNN is slightly better in terms of time elapsed.

As we showed the hybrid model (CNN-LSTM) did not improve on the loss value, because, the focus of classification is different as CNN works according to features and LSTM works according to the temporal patterns. Our results indicate that One Dimensional Convolutional Neural Network (1D CNN) can outperform the Long Short-Term Memory Neural Network across time-series problems.

We conclude that CNN model has predicted the least average error. Also, CNN has the least time elapsed. Considering all the parameters, CNN would be the best deep learning model for demand forecasting.

9.2 Future Work

One important possibility for future research is to improve the set of features by collecting data from other sources such as social media, shopping trends, social events, economic studies, and location-based demographic data of stores. A new variety of data sources contributions to deep learning can be observed. Also, in the future, we can implement neural networks and artificial intelligence in the social media marketing domain. Furthermore, another objective would be applying deep learning methods to maximize the product adoption on social networks.

References

Alsharif Mohammed H., Younes Mohammad K., and Kim Jeong. 2019. “Time Series ARIMA Model for Prediction of Daily and Monthly Average Global Solar Radiation: The Case Study of Seoul, South Korea.” *Symmetry* vol. 11, no. 2 pp. 240.

Bagheri Mohammad Ali, Gao Qigang, and Escalera Sergio. 2013. “A genetic-based subspace analysis method for improving error-correcting output coding.” *Pattern Recognition*, vol. 46, no. 10, pp. 2830–2839.

Bai Shaojie, Kolter Zico J., and Koltun Vladlen. 2018. “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.” Released 12 April 2018. Accessed 26 August 2019. Available : https://arxiv.org/pdf/1803.01271.pdf

Bengio Yoshua. 2012. “Practical Recommendations for Gradient-Based Training of Deep Architectures.” *Neural Networks: Tricks of the Trade*, Lecture Notes in Computer Science, vol. 7700, Springer, Berlin, Heidelberg.

Bengio Yoshua, Courville Aaron, and Vincent Pascal. 2013. “Representation learning: a review and new perspectives.*” IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828.

Box George E. P., Jenkins Gwilym M., Reinsel Gregory C., and Ljung Greta M., 2015. *Time Series Analysis: Forecasting and Control.* 5th Edition: Wiley Series in Probability and Statistics, John Wiley & Sons, Inc. New York, NY, USA.

Bradlow Eric T., Gangwar Manish, Kopalle Praveen, and Voleti Sudhir. 2017. “The Role of Big Data and Predictive Analytics in Retailing.” *Journal of Retailing*, vol. 93, no. 1, pp. 79–95.

Brownlee Jason. 2019. *Deep Learning for Time Series Forecasting*. E-book, Machine Learning Mastery Pty. Ltd.

Brownlee Jason. 2018a. “How to Develop Convolutional Neural Network Models for Time Series Forecasting.” *Deep Learning for Time Series, Machine Learning Mastery (blog).* Released November 2018. Accessed 08 August 2019. https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/

Brownlee Jason. 2018b. “When to Use MLP, CNN, and RNN Neural Networks.”, *Deep Learning, Machine Learning Mastery (blog)*. Released July 2018. Accessed 01 August 2019. https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

Dave Kuhlman. 2011. *A Python Book: Beginning Python, Advanced Python, and Python Exercises*. Platypus Global Media. Accessed 08 July 2019. Available at: http://www.davekuhlman.org/python_book_01.html

Demand forecasting. 2019. *Wikipedia(Website)*. Last modified 14 January 2019. Accessed 15 July 2019. https://en.wikipedia.org/wiki/Demand_forecasting

Doshi Sanket. 2019. “Various Optimization Algorithms for Training Neural Network.” *Neural Networks, Medium (blog)*. Released 13 January 2018. Accessed 5 August 2019. https://medium.com/@sdoshi579/optimizers-for-training-neural-network-59450d71caf6

Fildes Robert, and Petropoulos Fotios. 2015. “Simple versus complex selection rules for forecasting many time series.” *Business Research*, vol. 68, no. 8, pp. 1692–1701.

García Fernando Turrado, Villalba Luis Javier García, and Portela Javier. 2012. “Intelligent system for time series classification using support vector machines applied to supply-chain.” *Expert Systems with Applications*, vol. 39, no. 12, pp. 10590–10599.

Grewal Dhruv, Roggeveen Anne L., and Nordfalt Jens. 2017. “The Future of Retailing.” *Journal of Retailing*, vol. 93, no. 1, pp. 1–6.

Gujarati Damodar N. 2003. *Basic Econometrics*. McGraw Hill Ryerson Ltd, (July 2017): 5th edition, New York, NY, USA.

Haykin Simon O. 2008. *Neural Networks and Learning Machines*. Prentice Hall, Pearson Education, Inc., New Jersey, NJ, USA.

Hinton Geoffrey E., Osindero Simon, and Teh Yee-Whye. 2006. “A Fast Learning Algorithm for Deep Belief Networks.” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554.

Hyndman Rob J., and George Athanasopoulos. 2018. *Forecasting: Principles and Practice.* OTexts, Melbourne, Australia.

Khandelwal Renu. 2019 "Overview of different Optimizers for neural networks." *Neural Networks, Data Driven Investor.* Released 3 February 2018. Accessed 25 August 2019. https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3

Kilimci Zeynep Hilal, Akyuz A. Okay, Uysal Mitat, Akyokus Selim, Uysal M. Ozan, Bulbul Berna Atak, and Mehmet Ali Ekmis. 2019. "An Improved Demand Forecasting Model Using Deep Learning Approach and Proposed Decision Integration Strategy for Supply Chain." *Complexity*, vol. 2019, article id: 9067367, 15 pages. doi: https://doi.org/10.1155/2019/9067367.

Kim Sangwook, Yu Zhibin, Kil Rhee Man, and Lee Minho. 2014. "Deep learning of support vector machines with class probability output networks," *Neural Networks*, vol. 64, pp. 19–28.

Neural Networks Concepts. n.d. *Machine Learning Glossary(blog),* Read the Docs, Inc & Contributors. Accessed 6 September 2019. https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html

McGoldrick Peter J, and Andre Elisabeth. 1997. "Consumer misbehavior: promiscuity or loyalty in grocery shopping." *Journal of Retailing and Consumer Services*, vol. 4, no. 2, pp. 73–81.

Mehtha Anukrathi. 2019. "Comprehensive Guide to Types of Neural Networks." *Machine Learning, Digital Vidya Blog.* Released 25 January 2019. Accessed 9 September 2019. https://www.digitalvidya.com/blog/types-of-neural-networks/

Mikhailouskaya Irene. 2019. “Demand Forecasting Using Traditional and Contemporary Data Science.” *Data Analytics, Science Soft (blog).*Released 20 February 2019. Accessed 10 September 2019. https://www.scnsoft.com/blog/demand-forecasting-with-data-science#deep-learning-in-detail

Olteanu Madalina, Ridgway James. 2012. “Hidden Markov models for time series of counts with excess zeros.” *European Symposium on Artificial Neural Networks*, Belgium. pp.133-138. Article id: hal-00655588.
Available at : https://hal.archives-ouvertes.fr/hal-00655588/document

Pokhrel Sabina. 2019. “Beginners Guide to Convolutional Neural Networks,” *Machine Learning, Towards Data Science Inc(blog).*Released 19 September 2019. Accessed 10 October 2019. https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d

Qi Zhiquan, Wang Bo, Tian Yingjie, and Zhang Peng. 2016. “When ensemble learning meets deep learning: a new deep support vector machine for classification.” *Knowledge-Based Systems*, vol. 107, pp. 54–60.

Roy Rahul, and Rai Akanksha. n.d. “Best Python libraries for Machine Learning,” *Python Libraries, GeeksforGeeks (blog).* Accessed 9 October 2019. https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/

Scikit-learn, n.d. *Scikit-learn Developers*. Accessed 19 September 2019. https://scikit-learn.org/stable/

So Geoffery. 2019. “Should We Abandon LSTM for CNN?”, *AI/ML at Symantec, Medium (blog).*Released 23 March 2019. Accessed 26 August 2019. https://medium.com/ai-ml-at-symantec/should-we-abandon-lstm-for-cnn-83accaeb93d6

Symmetric mean absolute percentage error (sMAPE). 2018. *Wikipedia(website)*. Last modified 17 May 2018. Accessed 19 September 2019. https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error

Vieira Sandra, Pinaya Walter Hugo Lopez, Mechelli Andrea. 2019. “Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications.” *Neuroscience & Biobehavioral*, vol. 74, part A, pp. 58-75.

List of Appendices

Appendix A

CNN Model

Please note the indentations are necessary for the following code in the appendices.

```
import warnings
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from keras import optimizers
from keras.utils import plot_model
from keras.models import Sequential, Model
from keras.layers.convolutional import Conv1D, MaxPooling1D,Conv2D
from keras.layers import Dense, LSTM, RepeatVector, TimeDistributed, Flatten
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
#import plotly.plotly as py
#import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot

%matplotlib inline
warnings.filterwarnings("ignore")
init_notebook_mode(connected=True)
```

```
# Set seeds to make the experiment more reproducible.
from numpy.random import seed
tf.random.set_seed(1)
seed(1)

#Loading data :
train = pd.read_csv('train.csv', parse_dates=['date'])
test = pd.read_csv('test.csv', parse_dates=['date'])
print('Train shape:{}, Test shape:{}'.format(train.shape, test.shape))
train.head()

train.describe()
#date features :
train['dayofmonth'] = train.date.dt.day
train['dayofyear'] = train.date.dt.dayofyear
train['dayofweek'] = train.date.dt.dayofweek
train['month'] = train.date.dt.month
train['year'] = train.date.dt.year
train['weekofyear'] = train.date.dt.weekofyear
train.head()

# dummies features :
```

```
df=pd.get_dummies(train, columns =
['store','item','dayofmonth','dayofweek','month','weekofyear'])
df.head()

#train and validation dataframe :
msk = np.random.rand(len(df)) < 0.8
df_train = df[msk]
df_val = df[~msk]
print("train shape: ",df_train.shape)
print("validation shape :",df_val.shape)

df_train.drop('date',axis=1,inplace=True)
df_val.drop('date',axis=1,inplace=True)
y_train = df_train['sales'].values
y_test = df_val['sales'].values
X_train = df_train.drop('sales', axis=1).values
X_test = df_val.drop('sales', axis=1).values

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)


#CNN

X_train_series = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))

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

print('Train set shape', X_train_series.shape)

print('Validation set shape', X_test_series.shape)


def sMAPE(y_true, y_pred):

    #Symmetric mean absolute percentage error

    return 100 * K.mean(K.abs(y_pred - y_true) / (K.abs(y_pred) + K.abs(y_true)), axis=-
1)


import keras.backend as K

model_cnn = Sequential()

model_cnn.add(Conv1D(filters=64, kernel_size=2, activation='relu',
input_shape=(X_train_series.shape[1], X_train_series.shape[2])))

model_cnn.add(MaxPooling1D(pool_size=2))


model_cnn.add(Conv1D(filters=128, kernel_size=2, activation='relu'))

model_cnn.add(MaxPooling1D(pool_size=2))
model_cnn.add(Flatten())
model_cnn.add(Dense(50, activation='relu'))
model_cnn.add(Dense(1))
model_cnn.compile(loss= sMAPE, optimizer='adam')
model_cnn.summary()

cnn_history = model_cnn.fit(X_train_series, y_train, validation_data=(X_test_series,
y_test), epochs= 10,verbose=2,batch_size = 256)

import matplotlib.pyplot as plt

plt.plot(cnn_history.history['loss'])
plt.plot(cnn_history.history['val_loss'])
plt.title('CNN Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

Appendix B

LSTM Model

Please note the indentations are necessary for the following code in the appendices.

```
import tensorflow as tf
import matplotlib.pyplot as plt
from keras import optimizers
from keras.utils import plot_model
from keras.models import Sequential, Model
from keras.layers.convolutional import Conv1D, MaxPooling1D,Conv2D
from keras.layers import Dense, LSTM, RepeatVector, TimeDistributed, Flatten
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
#import plotly.plotly as py
#import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot


%matplotlib inline
warnings.filterwarnings("ignore")
init_notebook_mode(connected=True)


# Set seeds to make the experiment more reproducible.
from numpy.random import seed
```

```
tf.random.set_seed(1)

seed(1)


#Loading data :

train = pd.read_csv('train.csv', parse_dates=['date'])

test = pd.read_csv('test.csv', parse_dates=['date'])

print('Train shape:{}, Test shape:{}'.format(train.shape, test.shape))

train.head()


train.describe()

#date features :

train['dayofmonth'] = train.date.dt.day

train['dayofyear'] = train.date.dt.dayofyear

train['dayofweek'] = train.date.dt.dayofweek

train['month'] = train.date.dt.month

train['year'] = train.date.dt.year

train['weekofyear'] = train.date.dt.weekofyear

train.head()


# dummies features :

df = pd.get_dummies(train, columns =

['store','item','dayofmonth','dayofweek','month','weekofyear'])

df.head()
```

```
#train and validation dataframe :

msk = np.random.rand(len(df)) < 0.8

df_train = df[msk]

df_val = df[~msk]

print("train shape: ",df_train.shape)

print("validation shape :",df_val.shape)


df_train.drop('date',axis=1,inplace=True)

df_val.drop('date',axis=1,inplace=True)

y_train = df_train['sales'].values

y_test = df_val['sales'].values

X_train = df_train.drop('sales', axis=1).values

X_test = df_val.drop('sales', axis=1).values


from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()


X_train = scaler.fit_transform(X_train)


X_test = scaler.transform(X_test)
```

```
#CNN

X_train_series = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))

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

print('Train set shape', X_train_series.shape)

print('Validation set shape', X_test_series.shape)


def sMAPE(y_true, y_pred):

    #Symmetric mean absolute percentage error

    return 100 * K.mean(K.abs(y_pred - y_true) / (K.abs(y_pred) + K.abs(y_true)), axis=-
1)


import keras.backend as K

from keras.layers import LSTM


model_lstm = Sequential()


model_lstm.add(LSTM(100, input_shape=(X_train_series.shape[1],
X_train_series.shape[2]),return_sequences = True))


#model_lstm.add(LSTM(100,return_sequences = True))


model_lstm.add(LSTM(50))
model_lstm.add(Dense(1))


model_lstm.compile(loss= sMAPE, optimizer= 'adam')


model_lstm.summary()
model_lstm = model_lstm.fit(X_train_series, y_train, validation_data=(X_test_series,
y_test), epochs= 10, verbose=2, batch_size = 64)
import matplotlib.pyplot as plt
plt.plot(model_lstm.history['loss'])
plt.plot(model_lstm.history['val_loss'])
plt.title('LSTM loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.plot(cnn_history.history['loss'])
plt.plot(model_lstm.history['loss'])
plt.title('LSTM vs. CNN training loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['CNN', 'LSTM'], loc='upper right')
plt.show()
plt.plot(cnn_history.history['val_loss'])
plt.plot(model_lstm.history['val_loss'])

plt.title('LSTM vs. CNN validation loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['CNN', 'LSTM'], loc='upper right')

plt.show()
```

Appendix C

Hybrid Model (CNN-LSTM)

Please note the indentations are necessary for the following code in the appendices.

```
import warnings
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from keras import optimizers
from keras.utils import plot_model
from keras.models import Sequential, Model
from keras.layers.convolutional import Conv1D, MaxPooling1D,Conv2D
from keras.layers import Dense, LSTM, RepeatVector, TimeDistributed, Flatten
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
#import plotly.plotly as py
#import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot

%matplotlib inline
warnings.filterwarnings("ignore")
init_notebook_mode(connected=True)
```

```
# Set seeds to make the experiment more reproducible.
from numpy.random import seed
tf.random.set_seed(1)
seed(1)

#Loading data :
train = pd.read_csv('train.csv', parse_dates=['date'])
test = pd.read_csv('test.csv', parse_dates=['date'])
print('Train shape:{}, Test shape:{}'.format(train.shape, test.shape))
train.head()

train.describe()
#date features :
train['dayofmonth'] = train.date.dt.day
train['dayofyear'] = train.date.dt.dayofyear
train['dayofweek'] = train.date.dt.dayofweek
train['month'] = train.date.dt.month
train['year'] = train.date.dt.year
train['weekofyear'] = train.date.dt.weekofyear
train.head()

# dummies features :
```

```
df = pd.get_dummies(train, columns =
['store','item','dayofmonth','dayofweek','month','weekofyear'])
df.head()

#train and validation dataframe :
msk = np.random.rand(len(df)) < 0.8
df_train = df[msk]
df_val = df[~msk]
print("train shape: ",df_train.shape)
print("validation shape :",df_val.shape)

df_train.drop('date',axis=1,inplace=True)
df_val.drop('date',axis=1,inplace=True)
y_train = df_train['sales'].values
y_test = df_val['sales'].values
X_train = df_train.drop('sales', axis=1).values
X_test = df_val.drop('sales', axis=1).values

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)

#CNN
X_train_series = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_series = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
print('Train set shape', X_train_series.shape)
print('Validation set shape', X_test_series.shape)

def sMAPE(y_true, y_pred):
    #Symmetric mean absolute percentage error
    return 100 * K.mean(K.abs(y_pred - y_true) / (K.abs(y_pred) + K.abs(y_true)), axis=-
1)

import keras.backend as K
model_cnn_lstm = Sequential()

model_cnn_lstm.add(Conv1D(filters=64, kernel_size=2, activation='relu',
input_shape=(X_train_series.shape[1], X_train_series.shape[2])))
model_cnn_lstm.add(MaxPooling1D(pool_size=2))

model_cnn_lstm.add(Conv1D(filters=128, kernel_size=2, activation='relu'))
model_cnn_lstm.add(MaxPooling1D(pool_size=2))
model_cnn_lstm.add(LSTM(50, activation='relu'))

model_cnn_lstm.add(Dense(1))

model_cnn_lstm.compile(loss=sMAPE, optimizer='adam')

cnn_lstm_history = model_cnn_lstm.fit(X_train_series, y_train,

validation_data=(X_test_series, y_test), epochs=10, verbose=2,batch_size = 64)

import matplotlib.pyplot as plt


plt.plot(cnn_lstm_history.history['loss'])

plt.plot(cnn_lstm_history.history['val_loss'])

plt.title('CNN-LSTM Model Loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

plt.plot(cnn_history.history['loss'])

plt.plot(cnn_lstm_history.history['loss'])

plt.title('CNN-LSTM vs. CNN training loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['CNN-LSTM', 'CNN'], loc='upper right')

plt.show()

plt.plot(cnn_history.history['val_loss'])
plt.plot(cnn_lstm_history.history['val_loss'])

plt.title('CNN-LSTM vs. CNN validation loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['CNN-LSTM', 'CNN'], loc='upper right')

plt.show()
```