Walmart Sales Prediction

Introduction:

Walmart, as one of the premier retail chains in the US, relies heavily on accurate sales predictions to ensure a smooth customer experience and maintain a healthy bottom line. Given the vast size and geographical spread of Walmart, coupled with the numerous factors influencing sales, accurate sales forecasting can be quite challenging.

The Dataset:

The dataset provided for this project comes from 45 Walmart stores across various regions. The core objective is to harness this data to predict sales and demand more effectively, particularly during peak events and holidays, which traditionally see a spike in sales.

- *Dataset URL:* https://www.kaggle.com/datasets/yasserh/walmart-dataset/data

Features:

- Store: Represents one of the 45 different Walmart stores.
- Date: The specific date of sales data.
- Holiday_Flag: A binary indicator, with '1' denoting a holiday and '0' signifying a non-holiday.
- Temperature: The recorded temperature on the given date.
- Fuel_Price: The cost of fuel on the particular day.
- CPI (Consumer Price Index): Measures the average change in prices over time that consumers pay for a basket of goods and services.
- Unemployment: The unemployment index which provides insights into the overall economic health.

Target Variable:

- Weekly_Sales: Represents the sales figure for a given store on a specific date.

Data Preprocessing:

Before employing model training, it's crucial to preprocess the data, ensuring its quality and format align with the requirements of the modeling process. The following steps were undertaken:

- ***Handling Missing Values:***

  A comprehensive check was conducted across the dataset to identify and address any missing values. Result: No missing values were found, ensuring that the dataset is complete and reducing the chances of inaccuracies during modelling.
- ***Date Format Manipulation:***

  The 'Date' column provided in the dataset was split into three distinct columns to capture the Year, Month, and Day of the Week. This granular breakup of the date provides more specific temporal information, potentially aiding the model in identifying sales patterns related to particular months or days of the week.

- ***One-Hot Encoding:***

  Certain categorical variables such as Store, Year, Month, and Day of the Week were transformed using one-hot encoding. The one-hot encoding was done using the `pandas.get_dummies' method. This technique converts categorical variables into a format that can be better understood, transforming each category into its own binary column.

- ***Data Scaling:***

  The MinMax Scaler was employed to normalize the dataset values to fall within the range of [0,1]. Scaling is vital for many algorithms, ensuring that all features have the same scale, which can lead to faster convergence and improved model performance.

Experimental Setup:

- Optimization Algorithm: Adam optimizer.
- Loss Function: Mean Squared Error.
- Used ReLU in the hidden layer and sigmoid in the output layer.
- Number of Epochs: 500.
- Batch Size: Randomly choosen mini-batches, each containing 10% of the training dataset.
- EarlyStopping function has been implemented on h-h-1 network to halt training when MSE on training data does not improve more than 0.0001) in ten successive weight update iterations (min-batch presentations).
- An alternate EarlyStopping function has been implemented for remaining networks to halt training when MSE reaches E1.

Results & Discussion:

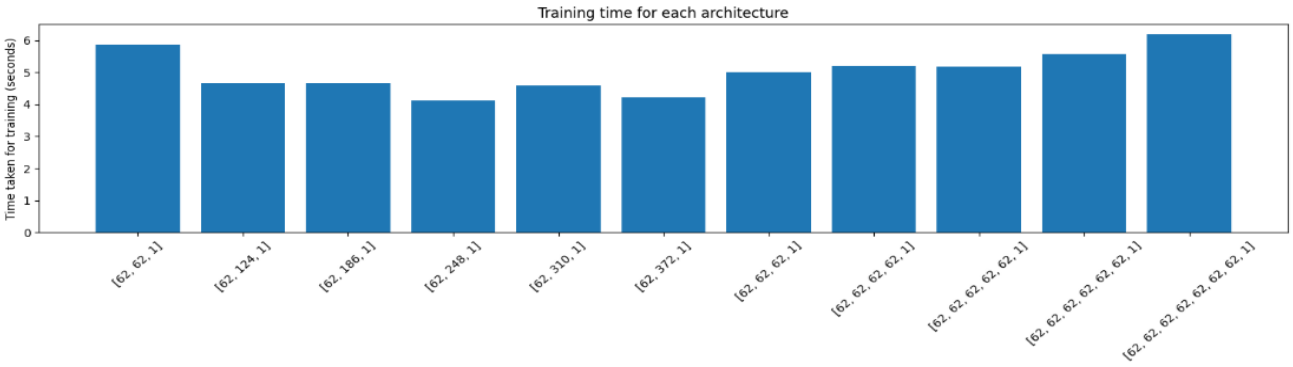
	Architecture	No. of weight updates needed to reach training MSE=E1	Time taken for training (seconds)	MSE on test data at the end of training
0	[62, 62, 1]	433125	5.859171	0.001105
1	[62, 124, 1]	497406	4.663211	0.001168
2	[62, 186, 1]	664062	4.670596	0.001126
3	[62, 248, 1]	613149	4.115913	0.001148
4	[62, 310, 1]	878639	4.591067	0.001219
5	[62, 372, 1]	942310	4.216984	0.001158
6	[62, 62, 62, 1]	530145	5.011729	0.001255
7	[62, 62, 62, 62, 1]	596106	5.207839	0.001209
8	[62, 62, 62, 62, 62, 1]	626976	5.191841	0.001153
9	[62, 62, 62, 62, 62, 62, 1]	681471	5.572876	0.001224
10	[62, 62, 62, 62, 62, 62, 62, 1]	822150	6.193226	0.001293

- ***Capacity vs Complexity:***

  As we increase the number of neurons in a single layer or add more layers, the network's capacity to learn complex relationships increases. However, this can also introduce the possibility of overfitting.

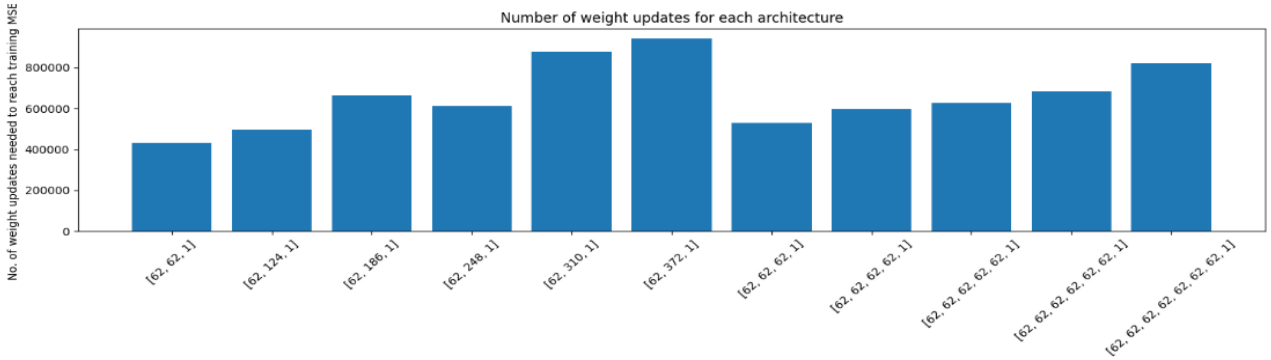
- ***Training Time:***

  Generally, adding more neurons or layers increases the total number of weights in the network. The more weights there are, the more weight updates must be performed, which can lead to longer training times. This is evident in the given results as architectures with more neurons and layers take longer to train. For example, the architecture [62, 62, 62, 62, 62, 62, 62, 1] took the longest time (6.193 seconds).



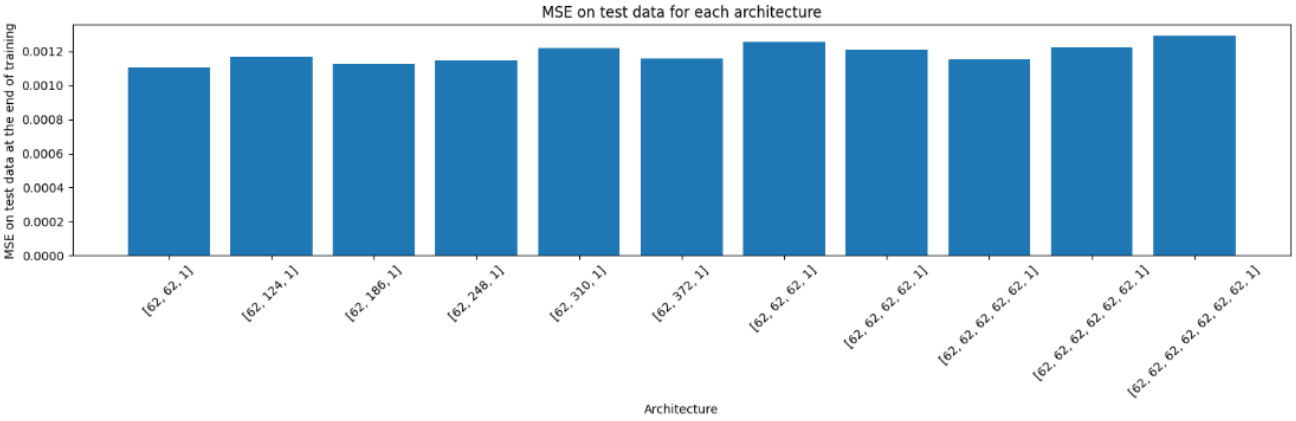
- ***Convergence (No. of Weight Updates):***

  More complex networks might need more weight updates to converge to a desired training error (MSE=E1). In the provided data, we see this trend, with more complex architectures generally requiring more weight updates to reach the target MSE.



- ***Test Performance (MSE on Test Data):***

  A more complex model does not necessarily result in better performance on unseen data. In the given results, the test MSE does not show a clear trend of improvement with increased complexity. The architecture with the lowest test MSE is [62, 62, 1], which is not the most complex. This suggests that the additional complexity in later architectures might be leading to overfitting or might not be adding meaningful representational power for this dataset.



Conclusion:

In summary, based on the provided results:

- Increasing the number of hidden nodes and layers does make the model more computationally intensive, as evidenced by increased training time and more weight updates.
- There isn't a clear linear relationship between complexity and test performance. In fact, some simpler architectures perform better on the test set than more complex ones.
- Overfitting might be a concern with very deep or wide networks, especially if trained on limited data.
- The optimal architecture often depends on the specific dataset, task, and available computational resources. It's always beneficial to test multiple architectures and choose the one that provides the best balance between training efficiency and predictive performance.