**2. PART I:**

**Title: The Evolution of Generative AI: Unveiling the Contributions of RNN and its Variations like LSTM**

Introduction:

Generative Artificial Intelligence (AI) has undergone remarkable advancements in recent years, offering unprecedented capabilities in generating text, images, and music that exhibit human-like creativity. Central to this progress are Recurrent Neural Networks (RNNs) and their specialized variant, Long Short-Term Memory (LSTM) networks. This essay delves into the pivotal contributions of RNNs and LSTMs to the emergence and evolution of generative AI, elucidating their significance in shaping the landscape of artificial creativity.

The Role of Recurrent Neural Networks (RNNs):

RNNs constitute a fundamental architecture in the realm of sequence modeling, capable of capturing temporal dependencies within data. Their cyclic connections enable them to maintain an internal state, thereby exhibiting memory of past information. This inherent ability to process sequential data makes RNNs indispensable in generative tasks such as language modeling, speech recognition, and time series prediction.

In the context of generative AI, RNNs serve as the cornerstone for various applications. They facilitate the creation of language models that can generate coherent and contextually relevant text. By leveraging the sequential nature of language, RNN-based models generate realistic sentences, paragraphs, and even entire articles, mimicking the style and structure of human-authored content. Furthermore, RNNs have been instrumental in generating music, where they learn patterns and melodies from existing compositions to compose original pieces that resonate with human preferences.

Despite their efficacy, traditional RNNs encounter challenges in learning long-term dependencies due to the vanishing or exploding gradient problem. This limitation inhibits their ability to retain and utilize information over extended sequences, hindering the generation of coherent and meaningful outputs.

Enter Long Short-Term Memory Networks (LSTMs):

To address the shortcomings of traditional RNNs, LSTM networks were introduced, representing a significant leap in sequence modeling capabilities. LSTMs incorporate specialized memory cells equipped with gating mechanisms that regulate the flow of information, allowing them to retain relevant information over prolonged sequences while mitigating the effects of gradient vanishing or explosion.

The introduction of LSTMs revolutionized generative AI by enabling the creation of more sophisticated and context-aware models. In natural language generation, LSTMs excel at capturing long-range dependencies, enabling the generation of coherent narratives and nuanced language expressions. Moreover, in image generation tasks, LSTMs facilitate the synthesis of visually compelling content by learning intricate patterns and structures present in the data.

Beyond their utility in standalone applications, LSTMs have also catalyzed advancements in multimodal generative AI, where multiple modalities such as text, images, and audio are integrated to produce rich and immersive outputs. By leveraging LSTMs' ability to encode and decode diverse data modalities, researchers have achieved breakthroughs in generating multimodal content that transcends individual mediums, fostering novel forms of creative expression.

Conclusion:

The advent and advances of generative AI owe much to the contributions of RNNs and their specialized variant, LSTM networks. From text and music generation to image synthesis and multimodal creativity, RNNs and LSTMs have played pivotal roles in pushing the boundaries of artificial creativity. As researchers continue to refine and innovate upon these architectures, the future of generative AI holds the promise of ever more sophisticated and human-like creative endeavors.

**3. PART II: A Time-series Data Set (10 Points)**

**SUBMISSION REQUIREMENT #1:**

**Dataset: Lottery Mega Millions Winning Numbers Beginning 2002**

**Introduction:**

The dataset contains the Mega Millions winning numbers starting from 2002 to the current year. Mega Millions is a popular lottery game played in multiple states in the United States. Each draw involves selecting five numbers from a pool of 70 and one Mega Ball number from a pool of 25. The dataset is provided by the Texas Lottery Commission and is available in CSV format.

**Data Description:**

**The dataset contains the following columns:**

Draw Date: Date on which the draw took place.

WB1: First white ball number selected.

WB2: Second white ball number selected.

WB3: Third white ball number selected.

WB4: Fourth white ball number selected.

WB5: Fifth white ball number selected.

MB: Mega Ball number selected

Megaplier: Megaplier value

There are a total of 8 columns and 1,846 rows in the dataset. The dataset contains data from 2002 to 2022.

**Analysis:**

The dataset provides information about the winning numbers of the Mega Millions lottery game. It can be used to analyze the frequency of occurrence of certain numbers, patterns of winning numbers, and the frequency of occurrence of Megaplier values. It can also be used to evaluate the effectiveness of different lottery strategies.

**Limitations:**

The dataset only provides information about the winning numbers of the Mega Millions lottery game. It does not provide information about the total number of players, the amount of money bet, or the distribution of players across different states. Therefore, the dataset cannot be used to estimate the odds of winning the lottery.

**Conclusion:**

In conclusion, the Mega Millions Winning Numbers dataset provides valuable information about the winning numbers of the Mega Millions lottery game. It can be used for a variety of purposes, including analyzing the frequency of occurrence of certain numbers and patterns, and evaluating the effectiveness of different lottery strategies. However, it is important to keep in mind the limitations of the dataset, which do not allow us to estimate the odds of winning the lottery.

LINK: <https://catalog.data.gov/dataset/lottery-mega-millions-winning-numbers-beginning-2002>

**4. PART III: RNN: Simple RNN with Sine Wave Data (10 Points)**

**SUBMISSION REQUIREMENT #2**

An input layer, one or more hidden layers, and an output layer make up the fundamental structure of a neural network. 70 neurons are spread throughout the layers of the RNN architecture shown below. There are 4 neurons in the input layer, 70 neurons in the hidden layer, and 1 neuron in the output layer. Data is first received by the input layer, processed by the hidden layers, and then produced by the output layer.

Based on the size of the input data and the task being carried out, the number of neurons in the input and output layers are chosen. In the meantime, testing with various numbers and balancing the complexity leads to the appropriate number of neurons in the hidden layers.

Chart, line chart

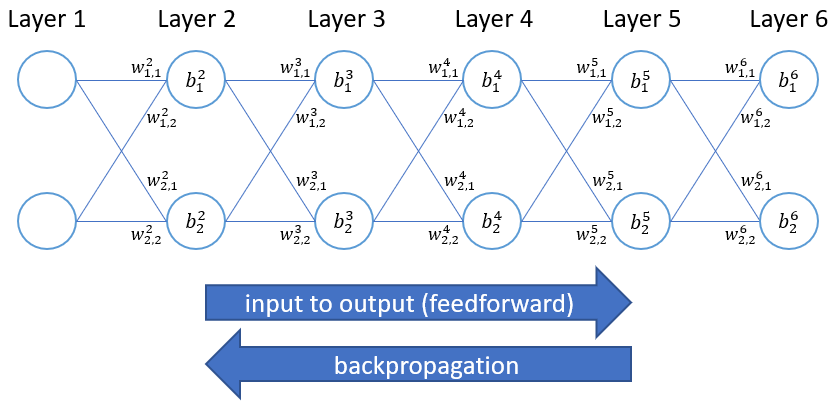
Description automatically generated

The sine wave dataset was made by dividing the range, which were then plotted to show a fit between -1 and +1 on the graph. Next, we imported the required libraries. An additional 20% of the dataset was put aside for testing before being divided into training and test datasets. The SimpleRNN model has two layers, a SimpleRNN layer with 70 neurons and a fully-connected layer with a single neuron. We utilized a TimeseriesGenerator with a lookback of time steps and a batch size of 1.

Diagram

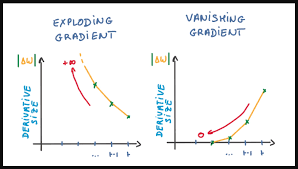
Description automatically generated with low confidence

**Vanishing Gradient Problem:** A well-known difficulty that can occur during the training of neural networks, especially ones with several layers, is the vanishing gradient problem. This issue is mostly caused by the network's activation functions and the large number of layers involved. The problem arises because the gradients computed about the parameters of the network's prior layers may reduce dramatically during backpropagation. In networks with many layers, this multiplication can result in the gradients reducing significantly as they travel backward through the layers. This is because the gradients are multiplied by the derivatives of the activation functions of each layer. This scenario might result in the earlier layers of the network receiving very small gradients, which can make it challenging to efficiently update their weights. This might make learning extremely slow and even prevent the network from learning entirely, resulting in poor performance.



**REFERENCE:** [**https://i.stack.imgur.com/ycYu6.png**](https://i.stack.imgur.com/ycYu6.png)

**Exploding Gradient Problem** is a challenge that emerges in neural networks during backpropagation when the weights of the network encounter exponential growth because of a significant gradient increase during training. The network becomes unstable because of this. Gradient clipping, which adjusts values to stay inside a threshold limit, and batch normalization, which scales all values to a consistent level, can both be used to alleviate this issue. Utilizing activation functions correctly is necessary to solve this problem.



**REFERENCE:**  [**https://tinyurl.com/ytt4p2jz**](https://tinyurl.com/ytt4p2jz)

**Discuss the limitations of the SimpleRNN neural network:**

**Difficulty with high-dimensional input data:** Due to the need for many parameters, processing high-dimensional data, such as videos and images, with Simple RNNs can be computationally expensive. As a result, significant memory utilization and computing resources may be required, making prolonged training a need for obtaining desired performance.

**Difficulty with non-linear data:** Modeling straightforward linear relationships between inputs and outputs is a strength of Simple RNN models. However, dealing with complex non-linear patterns may present difficulties, necessitating the use of more advanced models like LSTM or GRU networks to capture the intricate dependencies between the input and output sequences.

**Short-term memory:** Because of its memory constraints, SimpleRNN is unable to capture temporal dependencies that arise over lengthy periods. As a result, the network could find it challenging to learn intricate patterns that have a large time lag between input and output, which poses a serious obstacle when working with time-series data.

**Exploding gradients:** Gradient values may abruptly increase during the neural network training process, which creates the issue of exploding gradients. This could prevent the network from converging to an optimal solution by causing weight updates to be larger than desired levels.

**Vanishing Gradient:** SimpleRNN models are known to have the vanishing gradient problem, which happens when the gradients of the loss function with respect to the weights get too small during backpropagation. This may prevent weight updates, causing the network to learn slowly or ineffectively.

**Explain how the LSTM neural network can provide powerful solutions to both gradient problems: (Vanishing and Exploding) and address the limitations of the SimpleRNN neural network.**

The difficulties with SimpleRNNs, such as the issues with vanishing and exploding gradients, are addressed by Long Short-Term Memory (LSTM) neural networks. The LSTM architecture has grown in favor in a variety of applications, including speech-to-text and stock price prediction, since it can recognize patterns in time series data. The network uses unique gating mechanisms and memory cells to keep a longer-term memory of previous inputs and to selectively update or forget this information. This capability makes it possible to model complicated dependencies more successfully in sequential data, which improves time-series analysis performance. As a result, the LSTM architecture has gained popularity as a model for time-series analysis in the deep learning space.

A picture containing text, clock

Description automatically generated

In neural networks, gradients are used to guide weight changes over the course of learning, helping the network create increasingly accurate representations of the input data. Disappearing gradients occur when gradient values are too small, making it difficult for the network to transmit information over long sequences and recall events that occurred several time steps earlier. Contrarily, bursting gradients happen when gradient values rise too quickly, causing the network to become unstable and learning to stop entirely. Both issues might impair the network's functionality and hinder its ability to identify complex patterns in sequential data. Various methods, such as gradient clipping or the use of specific activation functions, have been employed as a result.

Due to its capability to keep a longer-term memory of previous inputs and to selectively update or forget this knowledge, the LSTM neural network is well suited for modeling and interpreting sequential data. Because of this, it is a popular option for many time series data applications, including speech recognition and natural language processing.

**6. PART V: RNN: LSTM with Time-Series Data (20 Points)**

Chart

Description automatically generated

**5. PART** **IV:** **RNN: LSTM with Time-Series Data (20 Points)**

Summary of Core Parameters

* How much of the data (20%) will be used for testing?
* How many LSTM layers are there? LSTM 3 layers
* Number of neurons in each layer of the LSTM? 40 neurons are present in each LSTM layer.
* Is there a DropOut layer? Yes.
* The proportion to decrease is 0.2 if the layer is DropOut.
* Time-series input sequence length: 45
* For testing and forecasting, the batch size must be ONE.
* The number of epochs for training must be 50.

Chart

Description automatically generated

Starting from scratch, we imported fundamental libraries including pandas, NumPy, matplotlib, Sequential, LSTM, Dropout, Dense, TimeseriesGenerator, and MinMaxScaler. The dataset, with the name df\_ALL and seven variables, was imported. But to build the model, we only need the 'Close' variable. To see the trend, we plotted the data after extracting the 'Close' variable and giving it the name df. while the remaining data were used for training. According to the summary statistics, the stock's low and high closing prices were, respectively.

Chart

Description automatically generated

After that, we used MinMaxScaler to normalize the data so that the features are all on a comparable scale, which would help machine learning algorithms perform better. Then, we created timeseries batches with a batch size and a 45-character input sequence for each training and testing timeseries batch. Three LSTM layers, each with 50 LSTM cells, and a fully connected layer for the final prediction were created into an LSTM model with a single feature. With a dropout rate of 0.2, a dropout layer was added after each LSTM layer. An activation function that is frequently used in deep learning models, Relu, was used for the LSTM layers.

After that, we tested the prediction by building a TimeseriesGenerator and visualizing it. To compare the prediction with the actual closure price, we also prepared another plot. For predicting, we created an LSTM model with 50 epochs. We saw a reduction in the loss also in this caseWe used normalized data to train our model, and to evaluate the findings, we had to rescale the data. To do it, we made advantage of the inverse\_transform function. We generated a data frame named forecast\_df with the forecast values contained in a single column and a timestamp index. The Close and Forecast plots were then combined.

Chart, line chart

Description automatically generated

To compare the prediction and actual values for the following we imported the second dataset for the time frame. To compare the two, we combined the actual close and forecast into a data frame called, then plotted it. The stock prices seem to have been predicted by the model, although they did not come close to the actual prices.

**7. PART VI: Redesign the Neural Network (10 Points)**

Summary of Core Parameters

* How much of the data (20%) will be used for testing?
* How many LSTM layers are there? LSTM 3 layers
* Number of neurons in each layer of the LSTM? 50 neurons are present in each LSTM layer.
* Is there a DropOut layer? Yes.
* The proportion to decrease is 0.2 if the layer is DropOut.
* Time-series input sequence length: 45
* For testing and forecasting, the batch size must be ONE.
* The number of epochs for training must be 30.

Chart, line chart

Description automatically generated

'Project # 1' and 'Project # 2' were given to the first and second LSTM models, respectively, to facilitate comparison. We altered a few significant parameters in the "Project # 2" project to enhance the model's performance. The LSTM network's architecture and behavior are controlled by these variables, which directly determine how well it can infer patterns from the input data and produce precise predictions. In this model, the testing partition was left alone.

Chart

Description automatically generated

In contrast to Project #1's 40 neurons, Project #2's 50 neurons make up each LSTM layer. With less complexity and a lower risk of overfitting, the model may be easier to train.In Project #2, the time-series input sequence is 20 characters long, which is less than Project #1's (45) length. More recent patterns can be captured by a shorter input sequence, which can also lessen the influence of potentially irrelevant earlier data. This could make it easier for the model to adjust as the data changes. Compared to Project # 1's 22 participants, Project # 2's training batch size is 22, which is same. Since the model can process more examples simultaneously with a larger batch size, training can be more effective. Overall training time may be shortened as a result.

**Count of epochs:** Compared to Project #1's 50, Project #2's 30 epochs for training. To improve accuracy and lower the chance of underfitting, more epochs might help the model discover more intricate patterns in the data.

Chart

Description automatically generated

The modifications made for Project #2 are intended to strike a better balance between the model's complexity and efficiency, which can aid in its ability to learn from the data more precisely and effectively while also lowering the risk of overfitting. This is crucial when discussing stock prices, which are quite susceptible to changes in the world economy. The input sequence has been condensed to 20 to accommodate for even the smallest changes in the data.

We can see that Project #2's total parameters decreased from Project #1's. This can lower the possibility of overfitting and increase the model's capacity for generalization. Since we slightly changed the number of epochs, the difference in loss might not be very noticeable. In contrast to the first and last epochs, the loss in Project #2 was lower.

Chart, line chart

Description automatically generated

Additionally, these modifications were reflected in the prediction, and Project #2 has produced better results than Project #1 as a result.

**8. PART VII: Compare Network Performance (10 Points)**

We made a few modifications that directly impact the predictions to enhance the model's performance. We used a 45-character input sequence length in Project #1; however, 20 characters were used instead in Project #2. Since stock prices are at stake, it is preferable to comprehend the short-term trends that drive volatility. To adjust the percentage of samples processed simultaneously, we slightly changed the batch size. The length of the samples generated fluctuated as we altered the input sequence and batch size. We had 22 for Project #1 and 22 for Project #2.

|  |  |
| --- | --- |
| Chart  Description automatically generated | Chart  Description automatically generated |

We did not adjust the number of features because we are only dealing with one, the "Close" pricing. Additionally, we left the dropout layer (0.2) and the number of LSTM layers (3 layers) unchanged. However, we altered the number of neurons in each layer. In Project #1, we employed 40 neurons in each LSTM layer; in Project #2, we used 50 neurons instead. The overall number of trainable parameters was influenced by these adjustments in Project #2. Overfitting risk may be decreased as a result.

In Project #2, the epoch count was reduced from 50 to 30. Even though both projects' losses dropped from the first to the last period, Project #2 showed no improvement. Between the two projects, there were some predictions that varied, as well.

|  |  |
| --- | --- |
| Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated |

The second model performed better than the first model when compared to the real values. Even the predicted closing price showed some improvement. In Project #2, the discrepancy between actual close and forecast has decreased, as seen in the table above. The second model fared better than the model created for Project #1, even though it does not precisely match the actuals. Additionally, several factors affect stock prices, including a company's financial stability, interest rates, political factors, etc. As a result, rather than depending solely on one feature, we must consider numerous other features when training the model.

**9. PART VIII: Project Report (20 Points)**

**Submission Requirements #7:**

A neural network's architecture, several layer types, and a method to count the number of neurons in a neural network were all covered in Section II. Layers of interconnected nodes or neurons make up neural networks, which process and change the data they receive as input. The connectivity and arrangement of these neurons are referred to as the neural network's architecture. The feedforward neural network architecture is the most popular kind of neural network, and it has three layers: an input layer, one or more hidden layers, and an output layer. A specific number of neurons are present in each layer, and weights connect the neurons in one layer to those in the next layer. The total of the neurons in each layer is needed to determine the number of neurons in a neural network architecture. An illustration would be a feedforward neural network input neurons, 3 hidden layers. A SimpleRNN with two layers that uses Sine Wave was created, trained, and assessed. In a neural network design, a sine wave pattern is followed by a Simple RNN with Sine Wave Data. Comparing this architecture to more complex ones like LSTMs or GRUs, such as those that require more training and implementation effort, is one of its benefits. It is thus a decent option for little tasks or for teaching.

In Section III, we learned about SimpleRNN’s drawbacks, including things like memory limits, disappearing gradients, and inflating gradients. We explained the function of each gate in the LSTM neural network and how it gets around these restrictions.

In Section IV, we created an LSTM neural network that can forecast Target Corporation stock values. We discovered the significance of TensorFlow and Keras in timeseries neural networks in addition to the fundamental libraries. We learned how to choose the amount of input time steps, use MinMaxScaler to normalize the data, choose the batch size for TimeseriesGenerator, and construct, train, and test the LSTM model. Additionally, we learned the value of the Dropout layer, the relu activation function, and how to put together a model. We gained practical experience in training a model and determining the quantity of training epochs needed.

We demonstrated our capacity for independent thought in Section V by experimenting with different modifications to the project finished in Section IV. We recognized the effects these modifications could have on a model's performance. The number of layers' neurons, the quantity of input time steps, and the batch size were a few of the adjustments applied.

In conclusion, the subjects discussed above are crucial for our understanding of RNNs. They offer a solid foundation in neural network architecture and operation, as well as the methods and tools needed to create and train them successfully. By learning these ideas and methods, we can use them to address a variety of issues in the current world.

**10. PART IX: Final Presentation Videos: YouTube Links**

**Submission Requirement #8:**

Part 1

https://youtu.be/wcYHZbP1Blg?si=FcBQJ31iUx\_RewKB

Part 2 –

https://youtu.be/yXbRe2Lnglc?si=oo\_A3YdXXvAUhgcq