Cosine-Gated LSTM

Goodluck Oguzie
College of Engineering and Physical Sciences
Aston University
Birmingham, UK
g.oguzie@aston.ac.uk

***Abstract*—This paper introduces the Cosine-Gated Long Short-Term Memory (CGLSTM), a novel architecture that integrates a cosine similarity-based gate with the vailla LSTM framework to improve sequence prediction accuracy. It addresses long-term dependencies in sequence data through experiments across multiple datasets and tasks such as the adding problem, MNIST and Fashion-MNIST classification, IMDB sentiment analysis, and language modelling on the Penn Treebank, the CGLSTM's performance is evaluated against LSTM, Gated recurrent unit, Recurrent Attention Unit, and Transformer models. Additionally, its effectiveness is demonstrated in the SocNavGym environment, highlighting its potential for real-world applications. Results show the CGLSTM model's superior capability in complex sequences, providing evidence that the integration of cosine similarity into LSTM leads to a 17% improvement in predictive accuracy and efficiency, offering a promising solution for various deep learning applications.**

***Index Terms*—Cosine-Gated LSTM, sequence prediction, long-term dependencies, natural language processing, sentiment analysis, language modeling, social navigation**

I. Introduction

Deep Learning architectures, such as Long Short-Term Memory networks (LSTMs) [1], Gated Recurrent Units (GRUs) [2], and Transformer architectures [3] have significantly improved the effectiveness of sequence prediction tasks using deep learning architectures. Among them, LSTMs stand as a foundational architecture, addressing a great number of problems from natural language processing to financial prediction [1]. However, like all models, LSTMs have challenges, particularly a decline in performance in long-term predictions, and they can struggle with capturing intricate patterns in sequential data.

To address these challenges, this paper proposes the Cosine-Gated LSTM (CGLSTM). This model introduces an additional gate that integrates the strength of LSTMs—their ability to capture long-term dependencies in sequence data—with the strength of cosine similarity, specifically its effectiveness in identifying the relevance of data points based on their orientation in vector space. The CGLSTM model aims to mitigate LSTM's limitations: vulnerability to vanishing and exploding gradients and limited contextual understanding. This integration seeks to improve the vanilla LSTM architecture, offering improved predictive accuracy and efficiency across a variety of deep learning applications.

Our experiments show that our model improved the long-term prediction capabilities of the LSTM.

The contributions of this paper are:

1) We propose a novel LSTM-based architecture, the CGLSTM, which integrates a cosine-similarity-based gate. This mechanism allows the model to automatically emphasize important information and deemphasize less relevant data in sequences, improving the LSTM's capacity.
2) We validate the performance of the CGLSTM model across a variety of tasks, demonstrating its capability to capture long-term dependencies on the adding problem and the pixel-by-pixel MNIST handwritten digit recognition task. We further demonstrate the effectiveness of the CGLSTM model in computer vision applications on the Fashion-MNIST image classification tasks. Additionally, we illustrate its performance in natural language processing applications through sentiment classification and language modeling tasks, highlighting its versatility and effectiveness across different domains. We also compare the predictive performance of our model in the SocNavGym environment.

II. Background and Motivation

Our research aims to improve the capabilities of LSTM models for various tasks across different domains. This initiative was inspired by consistent errors observed in our previous research [4] using LSTM models as a world model within the SocNavGym environment [5]. Significant prediction errors emerged as we predicted further into the future, highlighting the necessity for a more robust LSTM model capable of addressing complex tasks.

Our study enhances LSTM's predictive accuracy in environments like SocNavGym [5], exploring long-term dependencies with the adding problem and MNIST tasks. We demonstrate the CGLSTM model's effectiveness in computer vision, specifically with Fashion-MNIST classification, and in natural language processing, through sentiment classification and language modeling tasks. In doing so, we aim to demonstrate the effectiveness of the CGLSTM model across different domains.

III. Related Works

This section explores the evolution and current state of sequence prediction, highlighting their contributions while introducing our Cosine-Gated LSTM models.

A. Long Short-Term Memory Networks (LSTMs)

Traditional RNNs [6] often face challenges during training due to the vanishing gradient problem, where gradients either vanish or explode throughout long sequences in the backpropagation phase [7]. This issue prevents the RNN's ability to learn from long sequences. The LSTM's design mitigates this problem by using gates and a memory cell to manage the flow of information.

Hochreiter and Schmidhuber proposed LSTM networks in 1997 to capture dependencies in sequences for prediction tasks more effectively than their predecessors [8]. The LSTM, unlike vanilla RNNs, effectively retain information across long sequences, mitigating the vanishing gradient problem with their architecture [8].

An LSTM unit houses three distinct gates: the input gate, the output gate, and the forget gate. These gates work together to selectively retain or discard information and determine the output [9].

The **Forget Gate** takes h_{t-1} and x_t to make decisions on discarding information from the previous cell state C_{t-1}, using values between 0 and 1 for data retention or removal. Similarly, the **Input Gate** evaluates h_{t-1} and x_t to determine the information to remove from the current cell state C_{t-1}, and the **Output Gate** determines the next hidden state h_t from the updated cell state while the memory cell retains relevant information throughout the duration of data processing.

The transitions within an LSTM follow these equations:

$$\begin{aligned} f_t &= \sigma(\mathbf{W}f[\mathbf{h}t-1;\mathbf{x}_t]+\mathbf{b}_f),\\ i_t &= \sigma(\mathbf{W}i[\mathbf{h}t-1;\mathbf{x}_t]+\mathbf{b}_i),\\ \tilde{\mathbf{C}}_t &= \tanh(\mathbf{W}C[\mathbf{h}t-1;\mathbf{x}_t]+\mathbf{b}_C),\\ \mathbf{C}t &= f_t \odot \mathbf{C}t-1+i_t\odot\tilde{\mathbf{C}}_t,\\ o_t &= \sigma(\mathbf{W}o[\mathbf{h}t-1;\mathbf{x}_t]+\mathbf{b}_o),\\ \mathbf{h}_t &= o_t\odot\tanh(\mathbf{C}_t), \end{aligned} \tag{1}$$

Here, σ represents the sigmoid function, and $*$ denotes element-wise multiplication. The weights W and biases b correspond to the various gates and the cell state, with $[h_{t-1}, x_t]$ indicating the combined previous hidden state and current input.

B. Gated Recurrent Units (GRUs)

Introduced by Cho et al. in 2014, GRU addresses the vanishing gradient problem in sequence prediction more efficiently than LSTMs by simplifying the architecture [10]. GRU integrate the forget and input gates into a singular update gate and combine the cell state with the hidden state. This simplification results in a model that effectively manages long-range dependencies with lower computational demands [11]. The update gate $\mathbf{z}_t$ and the reset gate $\mathbf{r}_t$, together modulate the information flow within the hidden state $\mathbf{h}_t$. The update gate regulates the degree to which the GRU model updates its activation, and the reset gate determines how much past information to forget [11].

The choice between using LSTM and GRU units can depend on the specific requirements of the application and computational resources. In some cases, GRUs have been found to perform better than LSTMs because they are easier to compute and train due to their simplified structure. However, LSTMs have been more widely used and may perform better on more complex tasks that require modeling longer dependencies.

C. Transformers

Transformers, introduced by Vaswani et al. in 2017, have significantly revolutionized natural language processing. Unlike RNN models that process sequences step-by-step, Transformers process data in parallel and capture relationships between all elements in a sequence simultaneously [12] Transformers have been applied across various domains, particularly in tasks such as machine translation, text generation, and language modeling, demonstrating their efficiency in large-scale applications [12].

The Transformer architecture, consisting of multi-layer decoder and encoder, each made up of a stack of identical layers, has significantly advanced sequence-to-sequence translation. These layers include multi-head self-attention mechanisms and fully connected feed-forward networks. The encoder's multi-head attention layers allow the model to focus on different parts of the input sequence for processing various representation subspaces simultaneously. In the decoder, the encoding is processed through several neural network layers using masked multi-head attention layers. These layers enable the model to focus on prior positions within the sequence, ensuring that predictions at any given position are influenced solely by previously known outputs, thereby generating the final output. The final linear and softmax layer generates the probability distribution of the next token in the sequence. Positional encoding is added to the input embeddings to retain the sequence order information, which is important for the model's understanding of the dataset's inherent structure.

D. Cosine similarity

Cosine similarity, a measure derived from the cosine of the angle between two non-zero vectors, is commonly used in fields like information retrieval and recommendation systems [13], [14], it helps identify users or items with similar preferences. Its ability to focus on orientation rather than magnitude makes it a valuable metric for understanding relationships between entities.

Our research introduces the CGLSTM, which leverages the strengths of cosine similarity to improve the predictive capabilities of the vanilla LSTM model, particularly in scenarios involving long-term predictions and outliers. By combining the robust architecture of vanilla LSTM with the directional focus of cosine similarity, our model effectively addresses challenges related to long-term dependency and sensitivity to outliers in sequence prediction tasks.

IV. CGLSTM Architecture

The CGLSTM aims to improve the vanilla LSTM by integrating cosine similarity into its gating mechanism as illus-

trated in Figure 1. Cosine similarity focuses on the direction of vectors, rather than their magnitude. This metric is particularly useful in sequential data processing where the direction or trend of data is more informative than the size:

The calculation involves the following steps:

1) **Dot Product of A and B**: The dot product is defined as
$$A \cdot B = \sum_{i=1}^{n} A_i B_i \tag{2}$$
where A_i and B_i are the components of vectors A and B respectively, and n represents the dimensionality of the vectors.
2) **Magnitude of a Vector**: The magnitude (or norm) of vector A is given by
$$\|A\| = \sqrt{A \cdot A} = \sqrt{\sum_{i=1}^{n} A_i^2} \tag{3}$$
representing the vector's length from the origin in the space.
3) **Normalization and Cosine Calculation**: The cosine similarity is derived by normalizing the dot product of the vectors by the product of their magnitudes, yielding
$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{4}$$

Applying the definitions from equations (2) and (3) to (4), we can express the cosine similarity more explicitly as

$$\text{Cosine Similarity}(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5}$$

Equation 5 evaluates the cosine of the angle between vectors A and B, providing a similarity score that ranges from -1, indicating opposite directions, to 1, signifying identical directions. This score is derived from the dot product of the vectors, normalized by their magnitudes.

A. Cosine-Gated Integration in LSTM

This section explains the incorporation of cosine similarity into LSTM architecture, as illustrated in Figure 1, to create cosine gates. It involves three main steps:

1) **Input to Previous Output Similarity Calculation (gate-ic)**:
   - The input vector x is transformed through a mapping function to facilitate comparison with the LSTM's hidden states. This transformation is denoted as I_M, which aligns the input vector space with that of the LSTM's previous output.
   - The cosine similarity between I_M and the previous LSTM output *prv_output*, denoted as *gate_ic*, measures their alignment. This similarity is calculated as follows:
$$\text{Cosine Similarity (gate-ic)} = \frac{I_M \cdot \textit{prv\_output}}{\|I_M\|\|\textit{prv\_output}\|} \tag{6}$$
2) **Input to Current Output Similarity Calculation (gate-co)**:
   - Similarly, cosine similarity between I_M and the current LSTM output *output*, denoted as *gate_co*, assesses their alignment, indicating the similarity of the input vector to the current output state.
$$\text{Cosine Similarity (gate-co)} = \frac{\textit{output} \cdot I_M}{\|\textit{output}\|\|I_M\|} \tag{7}$$
3) **Cosine-Gated Integration Mechanism**:
   - This mechanism uses the computed cosine similarity values, *gate_ic* and *gate_co*, to dynamically adjust the integration of the transformed input vector I_M and the LSTM's output *output*. Initially, *gate_ic* is used to weigh the transformed input I_M, which is then added to the LSTM output *output*. This operation integrates the influence of the input's similarity to the previous output state with the current output.
   - Subsequently, the result of this addition is further modulated by *gate_co*, acting as a second layer of gating based on the similarity between I_M and the current LSTM output. This two-step gated integration ensures that the final output reflects a balance between historical data (as represented by *prv_output*) and current observations (as represented by *output*).
   - The modulated output, after being processed through these gating mechanisms, is then concatenated with the original LSTM output. This combined vector is passed through a linear transformation. The output of this transformation is further modulated by *gate_co*, emphasizing the final adjustment based on the current output's relevance. This step ensures that the final output h_t effectively integrates both the immediate and contextually relevant information, refined by the cosine similarity.
   - By incorporating these steps, the integrated-gated mechanism effectively regulates the flow and integration of information within the LSTM. It allows the model to dynamically adjust the influence of new input on the final output, based on its relevance to the current and preceding states. This dynamic adjustment mechanism improves the LSTM's ability to capture temporal dependencies in the sequential data.

By calculating and integrating cosine similarities, the CGLSTM is aimed at recognizing and prioritizing information that aligns with the learned patterns, thereby improving the overall predictive capability of the model for sequential data.

V. Experiments

To evaluate the performance of the CGLSTM model, we conducted comprehensive comparisons with vanilla LSTM, GRU, RAU, and the Encoder Transformer model across five benchmark tasks and a social navigation environment.

1) The adding problem [1], [15];

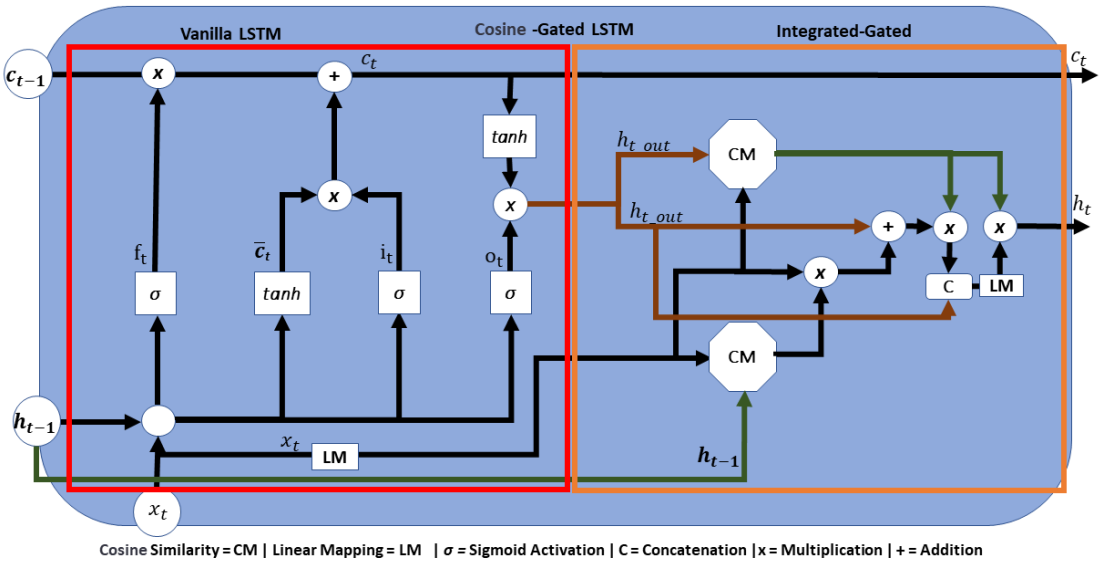

Fig. 1. Cosine-Gated LSTM (CGLSTM) Architecture. The left side shows the vanilla LSTM, while the right side illustrates our integrated Cosine Gate.

2) The row-wise MNIST digit recognition [16];
3) The sequential Fashion-MNIST item classification [17];
4) Sentiment analysis on IMDB movie reviews [18];
5) Word-level language modeling on the Penn Treebank corpus [19].
6) The SocNavGym [5];

These tasks are designed to challenge each model's ability to handle complex sequence data. Our evaluation criteria extended beyond mere accuracy to include training time, testing time, and statistical significance tests. This approach to performance evaluation allowed us to capture an overall view of each model's strengths and weaknesses. The hyperparameters used on each experiment are outlined in Table II, and the configuration details of the models, including the number of trainable parameters, are summarized in Table I.To ensure the reproducibility of our experiments, we used fixed seeds for random number generation across all models. Each model was run three times with three different seeds, and the mean results were used.

While Transformer-based models have shown impressive results across various tasks, their typically larger number of parameters can make them more resource-intensive [12]. We included the Encoder Transformer model in our experiments, despite its architectural differences from RNNs, to provide a broader perspective on the state-of-the-art performance in sequence modeling tasks.

TABLE I
NUMBER OF TRAINABLE PARAMETERS FOR OUR MODELS

Model	Number of Trainable Parameters
CGLSTM	118,794
GRU	61,962
LSTM	82,186
RAU	105,866
Encoder Transformer	601,638

A. The Adding Problem

The adding problem evaluates the ability of models to capture long-term dependencies, with our study focusing on sequences of length $T = 1000$ as established in previous research [1], [15]. An initial learning rate of 1×10^{-3} was used and reduced every 20,000 steps in accordance with practices used in related research [20]. The dataset was divided into a training set, a validation set, and a test set, each set containing 100,000, 50,000, and 50,000 examples respectively, to evaluate our model performance. The optimization for all models was conducted using the Adam optimizer.

The CGLSTM models showcased superior performance , outperforming the GRU, LSTM, Encoder Transformer, and RAU model in both validation and test MAE as shown in Table III. The CGLSTM model's predictive accuracy, indicated by significantly lower MAE values, validates its effectiveness in modeling long-term dependencies. Despite requiring approximately 5% more time for training and testing compared to the GRU model, the CGLSTM model's improvements in sequence prediction tasks justify this computational expenditure. Statistical analysis as shown in Table III through t-tests confirmed the CGLSTM model's performance advantage, with p-values less than 0.05 signifying statistical significance, except for RAU, which showed promising but not statistically significant results against the CGLSTM (p-value of 0.114).

TABLE II
SUMMARY OF HYPER-PARAMETERS USED IN OUR EXPERIMENT.

Parameter	MNIST	Fashion-MNIST	IMDB	Adding Problem	Language Modeling
Hidden Size	128	128	128	128	128
Epochs	213	213	100	35	30
Batch Size	128	128	128	128	128
Learning Rate	1e-3	1e-3	1e-3	1e-3	1e-3

B. The Row-wise MNIST Handwritten Digits Recognition

In this experiment, we evaluated performance on the task of row-wise MNIST handwritten digit recognition. The MNIST dataset comprises 28x28 pixel grayscale images, which we transformed into sequences of length 28. Each row of an image was treated as a step in the sequence, simulating a row-wise unfolding of the image for the recurrent models. The models were tasked with classifying the digit after sequentially processing all 28 rows.

The training dataset comprised 55,000 examples, while the validation and test sets contained 5,000 and 10,000 examples, respectively. The models were evaluated based on their classification accuracy. The detailed hyperparameters used in this experiment, such as hidden units and learning rate, are consistent across models and are listed in Table II. The models were trained using the Adam optimizer with a learning rate of 1×10^{-3}.

The CGLSTM model achieved the best accuracy of 99.07%, highlighted in Table III. Despite requiring about 5% more time compared to the GRU model, it demonstrated substantial improvement in performance. These results demonstrate the strengths and limitations of each model in the classification tasks.

C. FashionMNIST Classification Task

The FashionMNIST dataset, comprising 28x28 pixel grayscale images of clothing items, was divided into training, validation, and testing subsets. A balanced validation set was created to ensure equal representation of each class. The training set included 55,000 images, with 5,000 images reserved for validation and 10,000 images for the test set. The detailed hyperparameters used in this experiment are listed in Table II. Model performance was evaluated based on classification accuracy, training time, and testing time. Additionally, t-tests were conducted to statistically compare the performance of CGLSTM against other models. The detailed results, including mean validation accuracy, mean test accuracy, mean training time, mean testing time, and statistical comparisons, are presented in Table III.

The results, as shown in Table III, indicate that although the CGLSTM model required a longer training time of about 43.7% more compared to the GRU model, it achieved the highest accuracy of **90.12%** on the Fashion-MNIST dataset.

D. Sentiment Analysis on IMDB Movie Reviews

Our study extended to analyzing sentiments of movie reviews using the widely recognized IMDB dataset [18]. This collection features an equal division of 25,000 positive and negative reviews, each review averaging 231 words [21]. The dataset is divided into sets for training (25,000 reviews), validation (10,000 reviews), and testing (15,000 reviews), maintaining a balanced representation of both positive and negative sentiments. The hidden sizes, epochs, batch sizes, and learning rates used for the models are outlined in Table II.

The CGLSTM model, achieving a test accuracy of 86.30%, demonstrated competitive performance in the sentiment analysis on the IMDB movie reviews dataset, as evidenced by the results presented in Table III. Despite requiring approximately 5% longer training and testing times compared to the average of other models, the CGLSTM model shows a notable improvement in test accuracy. This improvement, supported by T-test statistics and p-values, indicates the model's robustness and effectiveness for sentiment analysis tasks.

E. Word-level Language Modeling on the Penn Treebank Corpus

In this experiment, we tested our CGLSTM on language modeling capabilities using the Penn Treebank (PTB) corpus [19]. The PTB corpus contains a vocabulary of 10,000 words, including 929,589 training words, 73,760 validation words, and 82,430 testing words. This dataset is widely used for various NLP tasks, such as syntactic analysis and part-of-speech tagging.

We measured model performance using perplexity, defined by the equation:

$$\log(\text{perplexity}(S)) = -\frac{1}{m}\sum_{i=1}^{m}\log(P(x_i|x_1, x_2, \ldots, x_{i-1})), \tag{8}$$

where $S = (x_1, x_2, \ldots, x_m)$ represents the words in a sentence, and m is the sentence length. A model achieves better performance with lower perplexity values.

The CGLSTM model demonstrated competitive performance in word-level language modeling, as evidenced by its test perplexity however the Encoder Transformer model exhibited superior performance, achieving the lowest perplexity, which indicates the highest predictive accuracy.

F. SocNavGym: Scaling to Somewhat Realistic Scenarios

We further evaluate its application in more realistic settings, we extended our experiment to the SocNavGym environment, aiming to compare the CGLSTM model's adaptability and performance in scenarios closer to real-world applications against vanilla LSTM, GRU and the Encoder Transformer only.

TABLE III
MODEL PERFORMANCE ON MNIST, FASHION-MNIST, IMDB, PENN TREEBANK, AND THE ADDING PROBLEM. LOWER MAE AND PERPLEXITY INDICATE BETTER PERFORMANCE FOR THE ADDING PROBLEM AND PENN TREEBANK, RESPECTIVELY.

Task	**LSTM**		**GRU**		**RAU**		**Encoder Transformer**		**CGLSTM**	
	Accuracy									
	Mean Val	Mean Test	Mean Val	Mean Test	Mean Val	Mean Test	Mean Val	Mean Test	Mean Val	Mean Test
MNIST	98.83%	98.42%	**98.93%**	98.70%	98.88%	98.86%	90.48%	90.93%	**98.93%**	**99.07%**
Fashion-MNIST	89.80%	89.26%	89.87%	**89.68%**	89.68%	89.45%	86.13%	85.16%	**90.68%**	**90.12%**
IMDB	**100%**	85.58%	**100%**	86.48%	**100%**	86.16%	99.96%	83.94%	**100%**	**86.30%**
PTB	103.43	105.98	104.95	107.51	103.13	105.55	**100.25**	**103.38**	101.47	104.43
Adding Problem	0.3336	0.3353	0.2634	0.2644	0.0389	0.0391	0.0481	0.0472	**0.0223**	**0.0225**
	Time (s)									
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
MNIST	2469.96	1.66	**1740.63**	**1.18**	2029.27	1.34	3674.42	1.77	2911.37	1.59
Fashion-MNIST	0.8051	1.5729	**0.6139**	**0.5658**	0.6420	1.2850	1.0169	1.9596	0.8791	1.7313
IMDB	1412.98	2.25	**980.19**	**1.22**	1211.66	1.66	4438.73	5.89	1889.96	2.88
PTB	1723.70	3.30	**1738.99**	**3.19**	895.40	5.46	2248.68	5.19	2747.75	6.06
Adding Problem	3.8540	0.0009	**3.0150**	**0.0009**	3.0777	0.0016	5.4688	0.0012	3.9615	0.0018
	Statistical Test									
	T-test Statistic	p-value	T-test Statistic	p-value	T-test Statistic	p-value	T-test Statistic	p-value		
MNIST	-1.47	0.276	-1.81	0.192	-3.56	0.063	-78.67	<0.001		
Fashion-MNIST	-4.20	0.026	-1.39	0.244	-2.20	0.098	-20.95	<0.001		
IMDB	-3.844	0.019	-0.947	0.413	-0.947	0.413	-5.476	0.021		
PTB	2.42	0.15	3.23	0.07	-0.94	0.38	-78.67	<0.001		
Adding Problem	21.038	<0.001	20.391	<0.001	-1.848	0.114	-2.990	0.031		

The SocNavGym [5] a social navigation environment for testing our model, simulating social navigation scenarios with dynamic obstacles that include an adjustable number of mobile humans and static obstacles flowerpots, tables, and laptops, offering a closer approximation to real-world applications. In our experiment, we used four humans, a table, and a flowerpot.

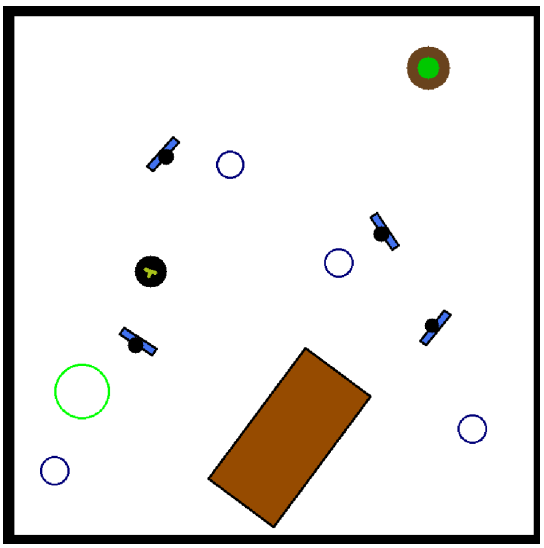

Fig. 2. Screenshot of SocNavGym, the environment used for the experiments [5]. Blue entities represent humans, blue circles indicate humans' goals (which are non-observable to the robot), green circles represent the robot's goals, and black-green circles represent robot agents.

Performance Metrics

For this experiment, we selected Mean Absolute Error (MAE) and Mean Squared Error (MSE) as the primary performance metrics to provide insights into prediction accuracy. For the training procedures and parameter settings, we used a learning rate of 1e-3, a hidden state size of 128, and early stopping to prevent overfitting.

The goal here is to evaluate and validate our proposed model, the CGLSTM, in a social navigation environment.

G. SocNavGym Environment

In the SocNavGym environment, we evaluated the performance of various models with a particular focus on the CGLSTM model. This experiment includes analysis on inference time, training and validation losses, the number of parameters, and predictive accuracy.

Table IV presents the Mean Absolute Error (MAE) for each model at different future time steps (k=1, k=3, k=5, and k=10), along with the prediction time in seconds. The CGLSTM model consistently exhibits the lowest MAE across all k-values, indicating its superior capability to predict future states with better accuracy within the SocNavGym environment.

The prediction time also provides insight into the computational efficiency of the models. While the GRU model shows the fastest prediction time, the CGLSTM model, despite having a slightly higher prediction time of about 2% when compared to GRU, achieves significantly better predictive accuracy of about 5% across all evaluated future time steps. This balance between accuracy and efficiency highlights the effectiveness of the CGLSTM model in real-time prediction tasks within dynamic environments such as SocNavGym.

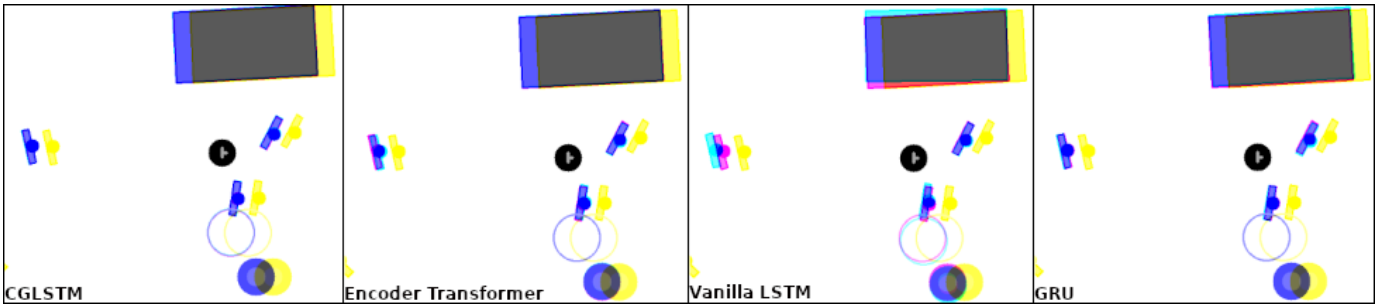

Fig. 3. Visual comparison of predictive performance in the SocNavGym environment, highlighting the color-coded accuracy of predictions: Blue indicates an accurate prediction aligned with the actual future states, red signifies discrepancies between the model's predictions and the actual future states, and yellow represents the current state. Specifically, (a) the CGLSTM model demonstrates precise alignment, indicating accurate predictions. (b) The Transformer model exhibits high predictive accuracy, with some inconsistency marked by red. (c) The Vanilla LSTM model, predominantly in red, shows a high level of inaccuracy in predictions. (d) The GRU model also shows inaccurate predictions but performs slightly better than the Vanilla LSTM model.

TABLE IV
COMPARISON OF PREDICTION TIME AND ACCURACY FOR VARIOUS MODELS IN SOCNAVGYM

Model	Prediction Time (s)	MAE (k=1)	MAE (k=3)	MAE (k=5)	MAE (k=10)
Vanilla LSTM	0.00137	1.65e-2	5.21e-2	9.61e-2	1.30e-1
GRU	**0.00126**	1.88e-2	5.53e-2	9.49e-2	1.35e-1
Encoder Transformer	0.02793	1.22e-2	3.73e-2	6.75e-2	1.12e-1
CGLSTM	0.00738	**8.40e-3**	**3.33e-2**	**6.41e-2**	**8.26e-2**

We visually represent and compare the predictive performances of the various models. In the figures, the blue color indicates an accurate prediction that aligns the model's predictions with the actual future states. In contrast, the red color signifies discrepancies between the model's predictions and the actual future states, highlighting areas where the model's predictions were not accurate. The yellow color represents the current state.

In Figure 3, we visually compare the predictive performance of various models in the SocNavGym environment. The color-coding in the figure indicates the accuracy of predictions: blue signifies accurate predictions that align with the actual future states, red denotes discrepancies between the model's predictions and the actual future states, and yellow indicates the current state.

The CGLSTM model demonstrates effective prediction accuracy, as indicated by the predominant blue color. The Encoder Transformer model also shows high predictive accuracy with blue representations, though slightly less consistently than the CGLSTM model, as evidenced by the presence of some red areas. The Vanilla LSTM and GRU models exhibit more red areas, suggesting a higher level of inaccurate predictions, with the Vanilla LSTM model displaying this more prominently. These visual representations from the SocNavGym Environment illustrate that the model architecture significantly influences predictive accuracy. The CGLSTM model stands out for effectively predicting environmental dynamics. The general trend observed across all models is that a higher frequency of blue color correlates with a model's ability to accurately predict future states, reinforcing the importance of model architectures for complex, dynamic scenarios.

Sequence length significantly impacts the performance of the CGLSTM model. Longer sequences provide more context and aid in capturing long-term dependencies but also increase computational complexity. In the adding problem with sequence length $T = 1000$, the CGLSTM outperformed other models in MAE. Similarly, in the row-wise MNIST digit recognition task with sequence length 28, the CGLSTM achieved the highest accuracy, demonstrating its robustness across varying sequence lengths. Our experiments revealed performance differences across various domains. In time series forecasting (adding problem), the CGLSTM had the lowest MAE. In image classification (row-wise MNIST, Fashion-MNIST), it achieved the highest accuracy. For NLP tasks (IMDB sentiment analysis, Penn Treebank language modeling), it showed competitive results with improved test accuracy and perplexity scores. In the SocNavGym environment (social navigation prediction), it exhibited robust performance. These results highlight the model's versatility and effectiveness across different domains, influenced by the nature of dependencies (temporal vs. contextual), as seen in Table III.

VI. CONCLUSION

In this paper, we introduced a novel RNN model, called the Cosine-Gated Long Short-Term Memory (CGLSTM), designed to improve sequence prediction tasks by integrating a cosine similarity-based gate within the traditional LSTM framework. This mechanism enables the model to dynamically focus on the most relevant information in sequences, thereby improving its ability to handle long-term dependencies and outliers effectively.

Through a series of experiments across diverse datasets and tasks, that include the adding problem, row-wise MNIST and Fashion-MNIST image classification, sentiment analysis on IMDB movie reviews, word-level language modeling on the

Penn Treebank corpus, and evaluation within the SocNavGym environment, we demonstrated the superior performance of the CGLSTM model compared to vanilla LSTM, GRU, RAU and the Encoder Transformer model.

The experimental results shows that the CGLSTM model consistently outperformed its counterparts in terms of predictive accuracy across various tasks. Specifically, the CGLSTM model exhibited a 9.5% improvement in test MAE over the GRU model in the adding problem experiment, a 9.9% improvement in test accuracy over LSTM and GRU models in row-wise MNIST handwritten digits recognition, a 0.44% improvement in mean test accuracy over LSTM in the FashionMNIST classification task, and a 2.36% improvement in test accuracy over GRU in sentiment analysis on IMDB movie reviews. Additionally, the CGLSTM model showcased competitive performance in word-level language modeling and outperformed the Transformer model in the SocNavGym environment with about 2.3% improvement in test MAE.

These findings highlight the advantages of the CGLSTM model, which not only offers superior predictive accuracy but also maintains a balance between model complexity and computational efficiency. This balance is important for the practical deployment of RNN models in real-world applications, especially where computational resources may be limited.

In conclusion, the CGLSTM model presents a promising advancement in sequence modeling, with potential applications in robotics, natural language processing, computer vision, and other fields.

For future work, we aim to integrate cosine similarity into GRU to improve its performance on sequence learning tasks. We will further explore variations in the cosine-similarity-based gating mechanism and investigate its application in additional domains beyond those studied in this paper. Additionally, we plan to extend the CGLSTM model to more complex scenarios such as multi-task learning and transfer learning, enhancing its applicability and robustness in diverse environments.

ACKNOWLEDGMENT

The implementation of the Cosine-Gated LSTM model is available on GitHub: https://github.com/goodluckoguzie/CosineGatedLSTM.

REFERENCES

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

[4] Goodluck Oguzie, Anikó Ekárt, and Luis J. Manso. Predictive world models for social navigation. In *Advances in Computational Intelligence Systems, Contributions Presented at the 22nd UK Workshop on Computational Intelligence*, Advances in Intelligent Systems and Computing (AISC), United Kingdom, 2023. Springer. In Press.

[5] Aditya Kapoor, Sushant Swamy, Pilar Bachiller, and Luis J. Manso. Socnavgym: A reinforcement learning gym for social navigation. In *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 2010–2017, 2023.

[6] Stephen Grossberg. Recurrent neural networks. *Scholarpedia*, 8(2):1888, 2013.

[7] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2000.

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[11] Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[13] Quan Duong, Mika Hämäläinen, and Khalid Alnajjar. Tfw2v: An enhanced document similarity method for the morphologically rich finnish language. *arXiv preprint arXiv:2112.12489*, 2021.

[14] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*, pages 325–341. Springer, 2007.

[15] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pages 1120–1128. PMLR, 2016.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[18] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[19] Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[20] Zhaoyang Niu, Guoqiang Zhong, Guohua Yue, Li-Na Wang, Hui Yu, Xiao Ling, and Junyu Dong. Recurrent attention unit: A new gated recurrent unit for long-term memory of important parts in sequential data. *Neurocomputing*, 517:1–9, 2023.

[21] Sida I Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, 2012.