Improving Pre-trained CNN-LSTM Models for Image Captioning with Hyper-Parameter Optimization

Nuha M. Khassaf
Informatics Institute for Postgraduate Studies, Iraqi Commission for Computers & Informatics, Iraq
phd202230705@iips.edu.iq (corresponding author)

Nada Hussein M. Ali
Department of Computer Science, College of science, University of Baghdad, Iraq
nada.husn@sc.uobaghdad.edu.iq

Received: 21 July 2024 | Revised: 10 August 2024 | Accepted: 22 August 2024

 DOI: https://doi.org/10.48084/etasr.8455

ABSTRACT

The issue of image captioning, which comprises automatic text generation to understand an image's visual information, has become feasible with the developments in object recognition and image classification. Deep learning has received much interest from the scientific community and can be very useful in real-world applications. The proposed image captioning approach involves the use of Convolution Neural Network (CNN) pre-trained models combined with Long Short Term Memory (LSTM) to generate image captions. The process includes two stages. The first stage entails training the CNN-LSTM models using baseline hyper-parameters and the second stage encompasses training CNN-LSTM models by optimizing and adjusting the hyper-parameters of the previous stage. Improvements include the use of a new activation function, regular parameter tuning, and an improved learning rate in the later stages of training. The experimental results on the flickr8k dataset showed a noticeable and satisfactory improvement in the second stage, where a clear increment was achieved in the evaluation metrics Bleu1-4, Meteor, and Rouge-L. This increment confirmed the effectiveness of the alterations and highlighted the importance of hyper-parameter tuning in improving the performance of CNN-LSTM models in image caption tasks.

Keywords-CNN pre-trained models; LSTM; activation function; hyper-parameters; overfitting

I. INTRODUCTION

Image captioning is a far more challenging problem than picture categorization, which has been the primary focus of computer vision researchers, because the interconnections between the picture's constituent parts must be conveyed in any adequate image description. Caption creation requires visual recognition of the image and the expression of semantic knowledge in a natural language such as English. Utilizing a CNN-LSTM model is one of the techniques used in generating image translation, which is seen as a combination of computer vision and natural language processing [1]. There are still many challenges in achieving high-performance models. These challenges include the need for large computational resources for training, hyper-parameter optimization, different dataset generalization, and adaptation to real-world applications that require high accuracy. Most existing studies have not adequately addressed the issue of choosing and optimizing hyper-parameters to improve performance. Also, the unstable performance of models across different datasets indicates the necessity for further work on domain adaptation and transfer learning [2]. Image caption generation is an active research area that integrates computer vision and natural language processing. CNN-LSTM models are the most popular tools in this area, where CNN is used to extract image features, and LSTM to generate text addressing the vanishing gradient problem, with its unique features being able to hold data values for long periods [2]. Authors in [3] proposed an improved Grey Wolf Algorithm (GWO) to design advanced CNN-LSTM networks for time series analysis. The optimization includes multiple search mechanisms to overcome local stagnation and increase the convergence speed. The enhanced models showed superior performance in time series prediction and classification compared to other traditional and state-of-the-art methods. The proposed approach in [4] concerned generating sentences from visual scenes by deploying a bidirectional LSTM decoder. The encoder relies on Inception v3 to extract object attributes and Places365 to extract scene attributes. The decoder uses bidirectional LSTM to create sentences. The results demonstrated an improvement in generating longer sentences.

Authors in [5] presented a system relying on VGG-16, Xception models to extract features and using an LSTM model to predict the outcome with a fixed-length output vector. The experiments provided satisfactory outcomes of image captioning but still need improvement. Authors in [6] tested feature extraction architectures (VGG, ResNet, and, DenseNet, Inception, ResNet) along with LSTM to explore how modifying the CNN feature extractor affects picture captioning quality. The Xception, Inception, and ResNet models outperformed the VGG model on the image description task. Authors in [7] employed the VGG architecture along with LSTM for image description and the provision of sequential text. This study displayed that layered normalization is superior in RNNs. Although both techniques aim to improve data flow and speed up learning, layered normalization is better suited to the sequential nature of RNNs [7]. Authors in [8] conducted a performance comparison between the Resnet50 and the InceptionV3 models, with LSTM to generate image captions. With a batch size of 128 and Adam optimization, the InceptionV3 model outperformed the Resnet50 model [8]. CNN-LSTM based models have been proven to be effective in image captioning, with their continuous advances in hyper-parameter optimization and training strategies to prevent overfitting and improve the accuracy of image captions.

In this paper, the proposed approach entails the use of pre-trained CNN models namely Xception, DenseNet121, InceptionV3, MobileNetV2, VGG16, VGG19 with LSTM to generate image captions. The process involves two stages. The first stage includes training the CNN-LSTM models using baseline hyper-parameters, and the second stage encompasses training CNN-LSTM models utilizing optimized parameters that helped reduce overfitting and improve efficiency. In this work, the model parameters are optimized deploying techniques, such as the Swish activation function instead of Rectified Linear Unit (Relu), Gradient Clipping by Norm (ClipNorm) tuning, and learning rate optimization after the fifth epoch.

II. METHODOLOGY

A. Convolutional Neural Networks (CNNs)

A CNN is designed to process data in the form of a 2D matrix. CNNs comprise convolutional, pooling, flattening, and fully connected layers [9]. CNNs can take an image as input, applying weights and biases to different parts of the image. Their ability to extract complex features from images and videos makes them suitable for tasks, such as image classification, object detection and differentiation [10].

B. Long Short-Term Memory (LSTM)

LSTM is a specialized type of Recurrent Neural Network (RNN) that can generate feedback loops by storing information in its internal states [11]. LSTM networks may effectively handle sequential input and record long-range relationships. The LSTM architecture primarily comprises three gates: input, output, and forget. These gates control the flow of data into and out of the cells and the ability to ignore the current value of the cell. The prior hidden states that are passed on to the next step are crucial [12].

C. CNN-LSTM Model

CNN-LSTM models handle sequential data by iterating steps over time and learning long-term dependencies between time steps, so they can learn spatial and temporal properties. A CNN-LSTM model learns from training data using convolutional and LSTM layers. Image captioning utilizes CNN encoders and LSTM decoders [13].

D. Hyper-Parameters

The structure and training method of a network are dictated by hyper-parameter including batch size, embedding dimension, length of generated texts, optimizer type, epoch number, sequence length, etc. [14]. Optimal model performance is achieved through parameter tweaking.

1) Number of Nodes in the Dense Layer

The number of neural nodes in the dense layer affects the model's ability to depict complex patterns.

2) Activation Function

It is used to transform the layer output into a nonlinear form: ReLU is a common and simple activation function, its output is equal to zero if the input is negative and equal to the input if it is positive. It is effective in training and accelerates the deep approximation process [15]:

$$y = \max(0, x) \quad (1)$$

where x represents the image matrix values.

Swish is a new activation function introduced to improve the performance of deep models. It is generally considered more effective than ReLU in improving model accuracy, and reduces the problem of dead neurons to a large extent [16]:

$$y = x * \text{sigmoid}(x) \quad (2)$$

The sigmoid is used to ensure nonlinearity and stimulate different outputs.

3) Dropout Rate

It represents the percentage of units that are dropped. It is determined randomly in each training update [17].

4) Number of LSTM Units

It determines the number of neural units in the LSTM layer, which affects the model's ability to accommodate temporal patterns in texts [18].

5) Word Embedding Size

It represents the specifying resulting dimension of vectors representing words [18].

6) Optimizer

It is an algorithm used to update the model weights with the aim of minimizing the loss function. It controls how the model learns from the data and improves its performance by gradually adjusting the weights during the training process. Adaptive Moment Estimation (Adam) is a widely employed advanced optimizer that combines the advantages of both SGD and adaptive learning rate [19].

7) Learning Rate (LR)

It is deployed in the training process. It has a small positive value and it affects the size of the steps the optimizer takes while adjusting the weights in the model during optimization to minimize the loss function [19].

8) Maximum Slope Value (Gradient Clipping Norm)

It is a hyper-parameter used to limit the length of the slope to prevent very large slope updates that might make the model unstable [20].

9) Batch Size

It defines the number of numerical samples that are passed through the model in one training step.

10) Epoch Number

The epoch number determines the number of times the model will be trained on the full data set.

11) Learning Rate Scheduler

It is a predefined framework used to define how the learning rate will be adjusted during training to maintain stability of updates and improve adaptability [19].

E. Overfitting

Overfitting occurs when there is a significant difference between the model's performance on training data and validation data, which means that the algorithm closely matches the training data, which leads to the model not being able to predict accurately other data. Common mechanisms to prevent overfitting are Early Stopping, Dropout, and Learning Rate Scheduler [21].

F. Dataset

The Flickr8K dataset was utilized for training and evaluation. The dataset contains 8,091 visual images, each with five captions. The images represent different scenes and are prepared using multiple labels for each image [22].

G. Pre-Processing

Image resizing is resized to match the requirements of the pre-trained model (e.g. 299 × 299 pixels for Xception). The images are normalized to pixel values between 0 and 1.

H. The Proposed Approach

Image captions are generated deploying the CNN-LSTM models in two stages. At the first stage, the CNN-LSTM models are trained using baseline hyper-parameters. At the second stage, the models are trained utilizing optimized hyper-parameters. The experiments were conducted on 6 CNNs pre-trained on the ImageNet dataset. Features were extracted from the last layer after removing the last fully connected layer on the flickr8k dataset. The considered models are summarized in Table I.

- Xception: It is an architecturally advanced ANN model developed by Google and contains 71 layers. It is based on the idea of dimension alternation, which employs multilevel processes to separate categories using images with a size of 299 × 299 pixels [23].
- DenseNet121: It is a model from the DenseNet family, and contains 121 layers. This model is characterized by a dense interconnection between layers, where the outputs of all the advanced layers are passed to the subsequent ones, which enhances the exchange of information between layers. It uses 224×224 pixel images [24].
- MobileNetV2: It is a model designed specifically for devices with limited resources such as mobile phones. It consists of 53 layers and is highly efficient. It relies on powerful technologies such as Depth-wise Separable Convolution and used images with a size of 224 × 224 pixels [25].
- InceptionV3: It is a model developed by the Google team, and is based on the "Inception" concept that uses multi-level processes to analyze images in an efficient way. It contains 48 layers and uses 299×299 images. It has good performance and high generalizability [26].
- VGG-16: It is a model from the VGG (Visual Geometry Group) series and it consists of 16 layers containing an average depth. It is easy to understand and to apply, and has performed well on many taxonomic tasks. It uses images with a size of 224 × 224 pixels [7].
- VGG-19: It is similar to the VGG-16 but has 19 layers, increasing the depth of the model. Increasing the number of layers can improve performance and enhance the model's ability to extract features. It uses 224 × 224 pixel images [27].

TABLE I. UTILIZED CNN MODELS

CNN Model	Number of layers	Image size	Feature Vector
Xception	71	299x299	2048
DenseNET121	121	224x224	1024
MobileNetV2	53	224x224	1280
InceptionV3	48	299x299	2048
VGG-16	16	224x224	2096
VGG-19	19	224x224	2096

III. IMPLEMENTATION

The CNN-LSTM models were implemented to generate image captions. They take the image as input and provide the text as output. Models were trained applying the Flickr8K dataset, which consists of 8091 images. Each image is described with five sentences of content. The dataset was divided into 6000 images for training, 1091 for validation, and 1000 for testing.

A. Modeling

Two stages were used based on hyper-parameter modification.

- First Stage: The CNN-LSTM model architecture is created, as shown in Figure 1, with the following parameter values: dropout (0.40), dense layer units (256), embedding size (256), LSTM units (256), optimizer (Adam), batch size (32), epochs (10), activation function (ReLU), and loss function (categorical-cross entropy).
- Second Stage: The CNN-LSTM model architecture is created, as evidenced in Figure 2, with the following

optimized hyper-parameter values: dropout (0.40), dense layer units (512), embedding size (512), LSTM units (512), optimizer (Adam), batch size (16), epochs (10), learning rate (0.0010), learning rate scheduler (lr_schedule). It halves the learning rate after 5 epochs), clipnorm (1.0), activation function (Swish), and the loss function (categorical-cross entropy).

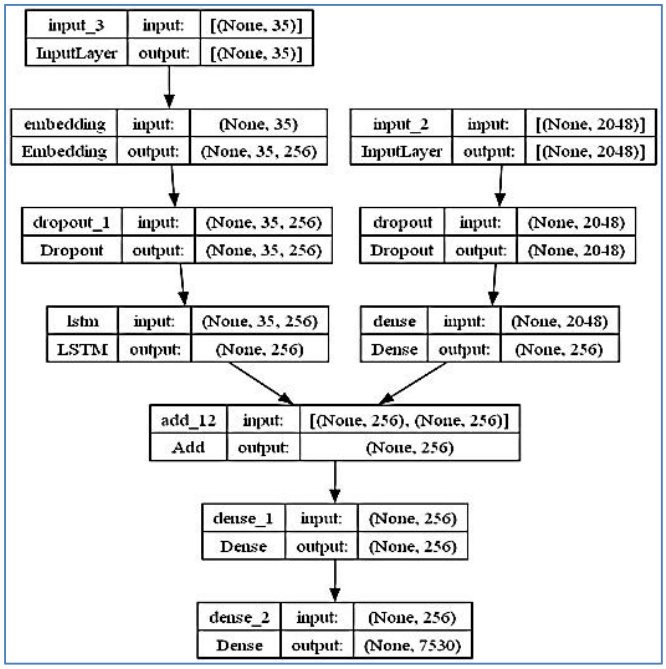

Fig. 1. CNN-LSTM model architecture - first stage.

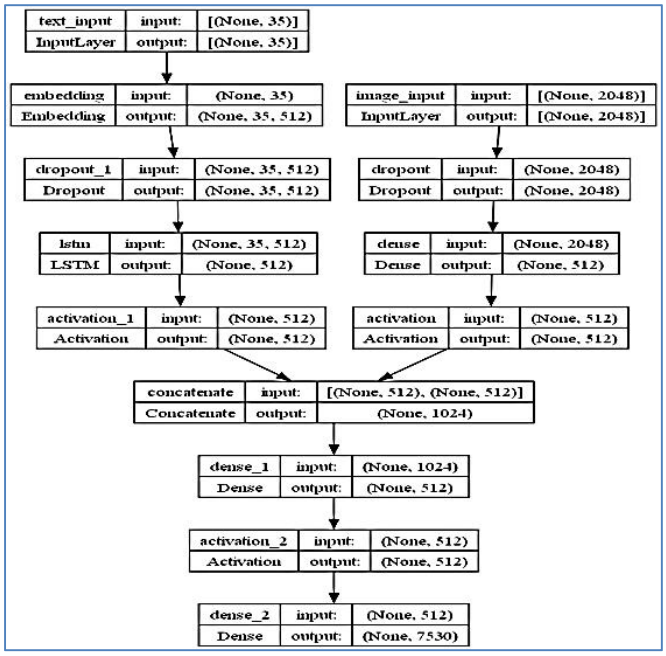

Fig. 2. CNN-LSTM model architecture - second stage.

B. Training

The cross-entropy loss function value was calculated for training and validating images during model training in both stages. Table II and Figure 3 depict the results for the first stage, and Table III and Figure 4 illustrate the results for the second stage.

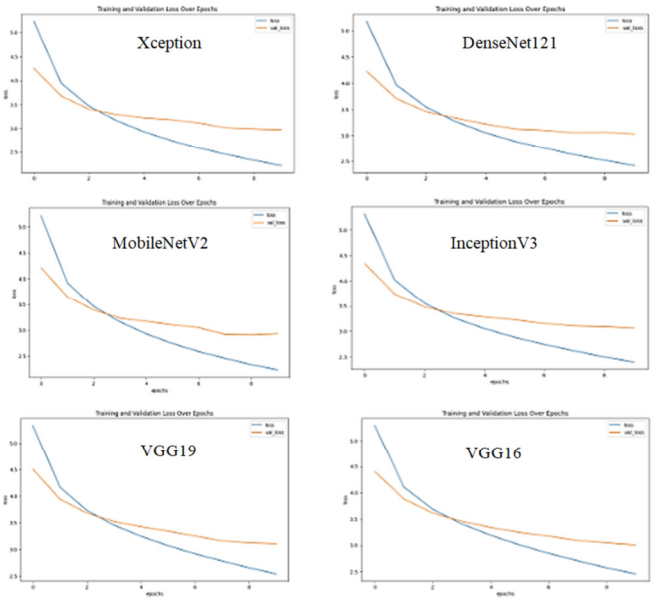

Fig. 3. Loss plot of the CNN-LSTM models - first stage.

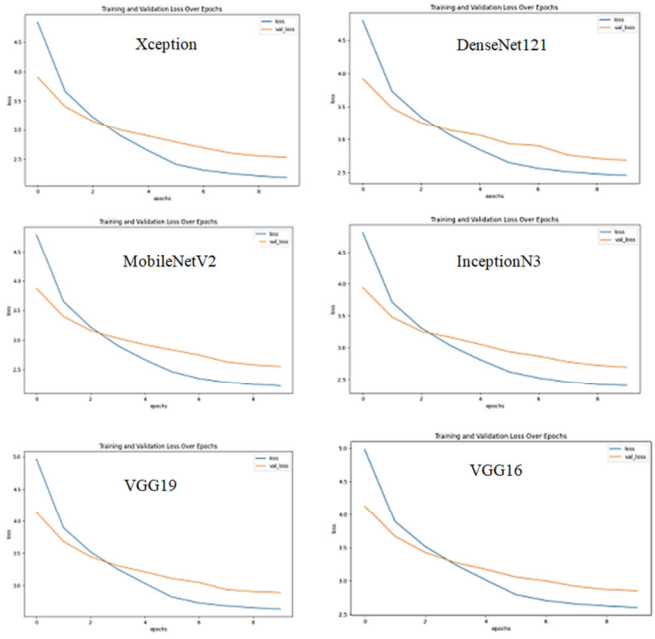

Fig. 4. Loss plot of the CNN-LSTM models - second stage.

TABLE II. LOSS TABLE OF THE CNN-LSTM MODELS FOR THE FIRST STAGE

CNN-LSTM model	Cross entropy-loss/ Training	Cross entropy-loss/ validation
Xception-LSTM	2.2136	2.9738
DenseNet121-LSTM	2.4149	3.0284
MobileNetV2-LSTM	2.2360	2.9310
InceptionV3-LSTM	2.3970	3.0666
VGG19-LSTM	2.5419	3.1088
VGG16-LSTM	2.4596	3.0098

TABLE III. LOSS TABLE OF THE CNN-LSTM MODELS FOR THE SECOND STAGE

CNN-LSTM model	Cross entropy-loss/ Training	Cross entropy-loss/ Validation
Xception-LSTM	2.1588	2.5006
DenseNet121-LSTM	2.4308	2.6546
MobileNetV2-LSTM	2.1872	2.5247
InceptionV3-LSTM	2.3690	2.6626
VGG19-LSTM	2.6068	2.8607
VGG16-LSTM	2.5737	2.8235

B. Discussion

This study included training pre-trained CNN models, namely Xception, DenseNet121, InceptionV3, MobileNetV2, VGG16, VGG19 along with LSTM in two stages. The second stage involved optimizing and adjusting the parameters based on the performance results derived from the first stage. The Swish function proved to be superior to the Relu activation function, which the former replaced, as it gives the model better nonlinear learning ability and improves its performance. Clipnorm also helped stabilize the training process by preventing very large updates that could cause the models to become unstable, while it was used with the Adam optimizer to set the maximum weight update rate to 1.0. Monitoring the learning rate and modifying after each fifth epoch led to improved performance and stability. These modifications were based on repeated experiments and observations of the model behavior. Table II-III and Figures 3-4 illustrate an effective comparative analysis of the two stages. At the first stage, it is observed that the models start to overfit after about 5 epochs, and the performance starts to decline, despite the use of methods, such as Dropout (0.40), to reduce overfitting and tuning Adam optimizer with default settings. At the second stage, Adam learning rate was set to 0.001 with a learning rate function that halves the rate after 5 epochs and the batch size was reduced to 16 instead of 32, which improved the learning process. Having used the concatenate function to layer the fusion from CNN and LSTM instead of the add function to combine layers at the first stage, gave the models a more complex representation of the extracted information.

C. Evaluation Metrics and Results

The considered models were evaluated with the Bleu1-4, Meteor, and Rouge-L metrics. These metrics are used to evaluate the quality of the text generated as outputs of the models compared to the reference texts [28]. Bleu measure is based on the concept of n-grams to calculate the degree of similarity between two texts, where the linguistic sequences of words are compared. Bleu uses four sub-indices to compare the linguistic sequences of words to evaluate the smoothness and structure of the generated sentence. Rouge-L measure deploys the recall metric to evaluate the quality of the linguistic model. It measures the proportion of common words between the two texts to the total number of words in the original text. The Meteor measure compares the two texts at the level of root words and synonyms to provide a more comprehensive and accurate assessment [29]. The comparison results are portrayed in Tables IV and V. The positive effect of modifying the hyper-parameters on the performance of the models can be clearly seen. All models achieved improvement in their performance after the modification. It should be noted that the Xception-LSTM model benefited greatly from the modification, overperforming the other models. DenseNet121-LSTM, MobileNetV2-LSTM, and InceptionV3 models also achieved clear improvement, confirming the flexibility of these models and their ability to benefit from the modifications. VGG16 and VGG19 models achieved less improvement, which may be due to the limitations of their architecture to benefit from such modifications.

TABLE IV. RESULTS OF CNN-LSTM MODELS - FIRST STAGE

CNN-LSTM Model	Bleu1	Bleu2	Bleu3	Bleu4	Meteor	Rouge-L
Xception-LSTM	0.4422	0.2590	0.1626	0.1071	0.2888	0.1999
DenseNET121-LSTM	0.4449	0.2638	0.1690	0.1115	0.2821	0.1942
MobileNetV2-LSTM	0.4389	0.2548	0.1622	0.1070	0.2709	0.1931
InceptionV3-LSTM	0.4312	0.2614	0.1683	0.1124	0.2886	0.1884
VGG19-LSTM	0.3862	0.2163	0.1350	0.0882	0.2321	0.1595
VGG16-LSTM	0.3721	0.2061	0.1272	0.0825	0.2329	0.1606

TABLE V. RESULTS OF CNN-LSTM MODELS - SECOND STAGE

CNN-LSTM Model	Bleu1	Bleu2	Bleu3	Bleu4	Meteor	Rouge-L
Xception-LSTM	0.5267	0.3324	0.2261	0.1589	0.3602	0.2509
DenseNET121-LSTM	0.5144	0.3253	0.2225	0.1570	0.3438	0.2484
MobileNetV2-LSTM	0.5111	0.3241	0.2196	0.1547	0.3496	0.2454
InceptionV3-LSTM	0.4973	0.3122	0.2129	0.1504	0.3481	0.2398
VGG19-LSTM	0.4362	0.2585	0.1749	0.1251	0.2905	0.2039
VGG16-LSTM	0.4490	0.2691	0.1832	0.1299	0.3007	0.2124

The proposed models were compared with previous studies (Table VI), where the default baseline parameters were used. The results obtained at the second stage (Table V) generally outperform those of previous studies in most Bleu indicators, especially in Blue3 and Blue4, reflecting a noticeable improvement in text generation accuracy. In contrast, previous studies exhibited some higher values in Blue1, so, the performance of the proposed models represents a more comprehensive improvement thanks to the adjustments made to the parameters.

TABLE VI. RESULTS OF CNN-LSTM MODELS - PREVIOUS STUDIES

Method	Bleu1	Bleu2	Bleu3	Bleu4	Meteor
VGG16-LSTM [5]	0.46	0.32	0.10	0.04	-
Xception-LSTM [6]	44.0	25.9	14.7	8.0	15.1
DenseNet121-LSTM [6]	41.0	23.0	12.3	6.3	14.4
VGG19-LSTM [6]	41.6	23.9	13.1	7.1	13.9
InceptionV3-LSTM [8]	0.53	0.35	0.18	0.09	-

IV. LIMITATIONS

The proposed approach requires computational resources, particularly for training and hyper-parameter optimization. Careful selection of parameters is necessary to balance performance and computational cost. Regarding generalization, the model performance may vary across different datasets, requiring further work on adaptation when real-world applications needing high accuracy, like assistive technologies that are used to help people with disabilities perform daily tasks and facilitate communication and learning, are demanded.

V. CONCLUSIONS AND FUTURE WORK

This study uses several pre-trained CNN models, namely Xception, DenseNet121, InceptionV3, MobileNetV2, VGG16, and VGG19, which perform best in feature extraction with sequence text prediction utilizing LSTM decoding to generate image captions. The testing results obtained from the two stages employing the Bleu1-4, Meteor, and Rouge-L evaluation metrics, showed noticeable performance improvements due to modifications performed in the hyper-parameters at the second stage. Among these modifications, replacing the Relu activation function with Swish provided tnhancements in the training stability and performance. Moreover, adding Clipnorm with a value of 0.1 and adjusting the learning rate were effective in reducing overfitting, which contributed to achieving more stable performance. These improved results provide an important starting point for future research and development to enhance CNN-LSTM models, either by modifying their architecture, e.g. including attention modules, or by optimizing their hyper-parameters. Bayesian optimization could help find optimal hyper-parameters. Random search allows a wider range of values for testing hyper-parameters, potentially leading to the discovery of more efficient settings, while the use of differential evolution could provide powerful and effective performance improvements.

REFERENCES

[1] C. P. Chaudhari and S. Devane, "Capturing Semantic Knowledge In Object Localization In Captioning Images," in *International Conference on Communication information and Computing Technology*, Mumbai, India, Jun. 2021, pp. 1–4, https://doi.org/10.1109/ICCICT50803.2021.9510175.

[2] S. M. Al-Selwi, M. F. Hassan, S. J. Abdulkadir, and A. Muneer, "LSTM Inefficiency in Long-Term Dependencies Regression Problems," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 30, no. 3, pp. 16–31, May 2023, https://doi.org/10.37934/araset.30.3.1631.

[3] H. Xie, L. Zhang, and C. P. Lim, "Evolving CNN-LSTM Models for Time Series Prediction Using Enhanced Grey Wolf Optimizer," *IEEE Access*, vol. 8, pp. 161519–161541, Jan. 2020, https://doi.org/10.1109/ACCESS.2020.3021527.

[4] D. Agughalam, P. Pathak, and P. Stynes, "Bidirectional LSTM approach to image captioning with scene features," in *Thirteenth International Conference on Digital Image Processing*, Singapore, Singapore, Dec. 2021, vol. 11878, pp. 81–88, https://doi.org/10.1117/12.2600465.

[5] J. Basnet, S. Kumari, and M. Rathore, "Image caption generator using CNN and LSTM," *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 8, no. 2, pp. 489–495, 2022.

[6] M. A. Al-Malla, A. Jafar, and N. Ghneim, "Pre-trained CNNs as Feature-Extraction Modules for Image Captioning: An Experimental Study," *ELCVIA. Electronic letters on computer vision and image analysis*, vol. 21, no. 1, pp. 1–16, 2022, https://doi.org/10.5565/rev/elcvia.1436.

[7] P. R. Devi, M. T. Deepak, M. Lohitha, M. S. C. Raju, and K. Venkata, "Image Caption Generator Using VGG and LSTM For Visually Impaired," *International Journal of Advances in Engineering and Management*, vol. 5, no. 4, pp. 576–583, 2023, https://doi.org/10.35629/5252-0504576583.

[8] H. Priyambudi and A. Hadinegoro, "Performance Analysis RESNET50 and INCEPTIONV3 Models for Caption Image Generator," *JURTEKSI (Jurnal Teknologi dan Sistem Informasi)*, vol. 9, no. 3, pp. 521–528, Jun. 2023, https://doi.org/10.33330/jurteksi.v9i3.2277.

[9] A. A. Ali and F. A. A. Dawood, "Deep Learning of Diabetic Retinopathy Classification in Fundus Images," *Journal of Engineering*, vol. 29, no. 12, pp. 139–152, Dec. 2023, https://doi.org/10.31026/j.eng.2023.12.09.

[10] M. M. H. Milu, M. A. Rahman, M. A. Rashid, A. Kuwana, and H. Kobayashi, "Improvement of Classification Accuracy of Four-Class Voluntary-Imagery fNIRS Signals using Convolutional Neural Networks," *Engineering, Technology & Applied Science Research*, vol. 13, no. 2, pp. 10425–10431, Apr. 2023, https://doi.org/10.48084/etasr.5703.

[11] H. S. Abdullah, N. H. Ali, and N. A. Z. Abdullah, "Evaluating the Performance and Behavior of CNN, LSTM, and GRU for Classification and Prediction Tasks," *Iraqi Journal of Science*, vol. 65, no. 3, pp. 1741–1751, Mar. 2024, https://doi.org/10.24996/ijs.2024.65.3.43.

[12] S. Gupta, S. Agnihotri, D. Birla, A. Jain, T. Vaiyapuri, and P. S. Lamba, "Image Caption Generation and Comprehensive Comparison of Image Encoders," *Fusion: Practice and Applications*, vol. 4, no. 2, pp. 42–55, Jan. 2021, https://doi.org/10.54216/FPA.040202.

[13] B. Deepika, S. P. Reddy, S. G. Satya, and K. R. Kumar, "Image Caption Generator," in *International e-Conference on Advances in Computer Engineering and Communication Systems*, Hyderabad, India, Sep. 2023, pp. 360–370, https://doi.org/10.2991/978-94-6463-314-6_35.

[14] S. K. Shukla, S. Dubey, A. K. Pandey, V. Mishra, M. Awasthi, and V. Bhardwaj, "Image Caption Generator Using Neural Networks," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 7, no. 3, pp. 1–7, May 2021, https://doi.org/10.32628/CSEIT21736.

[15] H. K. Dhahir and N. H. Salman, "A Review on Face Detection Based on Convolution Neural Network Techniques," *Iraqi Journal of Science*, vol. 63, no. 4, pp. 1823–1835, Apr. 2022, https://doi.org/10.24996/ijs.2022.63.4.39.

[16] "machine-learning-articles/why-swish-could-perform-better-than-relu.md," *GitHub*. https://github.com/christianversloot/machine-learning-articles/blob/main/why-swish-could-perform-better-than-relu.md.

[17] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System," *IEEE Access*, vol. 10, pp. 99837–99849, Jan. 2022, https://doi.org/10.1109/ACCESS.2022.3206425.

[18] B. Subedi and B. Krishna Bal, "CNN-Transformer based Encoder-Decoder Model for Nepali Image Captioning," in *19th International Conference on Natural Language Processing*, New Delhi, India, Dec. 2022, pp. 86–91.

[19] N. Landro, I. Gallo, and R. La Grassa, "Mixing ADAM and SGD: a Combined Optimization Method," *arXiv e-prints*. Nov. 01, 2020, https://doi.org/10.48550/arXiv.2011.08042.

[20] V. V. Mai and M. Johansson, "Stability and Convergence of Stochastic Gradient Clipping: Beyond Lipschitz Continuity and Smoothness," in *38th International Conference on Machine Learning*, Jul. 2021, pp. 7325–7335.

[21] C. Ma, Y. Liu, J. Deng, L. Xie, W. Dong, and C. Xu, "Understanding and Mitigating Overfitting in Prompt Tuning for Vision-Language Models," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 9, pp. 4616–4629, Sep. 2023, https://doi.org/10.1109/TCSVT.2023.3245584.

[22] R. Mulyawan, A. Sunyoto, and A. H. M. Muhammad, "Pre-Trained CNN Architecture Analysis for Transformer-Based Indonesian Image Caption Generation Model," *JOIV : International Journal on Informatics Visualization*, vol. 7, no. 2, pp. 487–493, May 2023, https://doi.org/10.30630/joiv.7.2.1387.

[23] I. Taneja and S. Maggu, "Generating Captions for Images Using Neural Networks," *IRE Journals*, vol. 6, no. 12, pp. 214–218, 2023.

[24] R. Khan, M. S. Islam, K. Kanwal, M. Iqbal, M. I. Hossain, and Z. Ye, "A Deep Neural Framework for Image Caption Generation Using GRU-Based Attention Mechanism." arXiv, Mar. 03, 2022, https://doi.org/10.48550/arXiv.2203.01594.

[25] I. I. Amal, D. H. Widyantoro, and A. Umam, "MobileNet-based Neural Image Caption Model in Title Generation for Product's Images," in *7th International Conference on Advance Informatics: Concepts, Theory and Applications*, Tokoname, Japan, Sep. 2020, pp. 1–6, https://doi.org/10.1109/ICAICTA49861.2020.9428886.

[26] R. D. Dondapati, T. Sivaprakasam, and K. V. Kumar, "Dermatological Decision Support Systems using CNN for Binary Classification," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14240–14247, Jun. 2024, https://doi.org/10.48084/etasr.7173.

[27] S. Mundargi and M. H. Mohanty, "Image Captioning using Attention Mechanism with ResNet, VGG and Inception Models," *International Research Journal of Engineering and Technology*, vol. 7, no. 9, pp. 3791–3801, 2020.

[28] M. Bhalekar and M. Bedekar, "D-CNN: A New model for Generating Image Captions with Text Extraction Using Deep Learning for Visually Challenged Individuals," *Engineering, Technology & Applied Science Research*, vol. 12, no. 2, pp. 8366–8373, Apr. 2022, https://doi.org/10.48084/etasr.4772.

[29] Z. Ren, S. Gou, Z. Guo, S. Mao, and R. Li, "A Mask-Guided Transformer Network with Topic Token for Remote Sensing Image Captioning," *Remote Sensing*, vol. 14, no. 12, Jan. 2022, Art. no. 2939, https://doi.org/10.3390/rs14122939.