Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 218 (2023) 686–696

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering

Hybrid Architecture using CNN and LSTM for Image Captioning in Hindi Language

Ayush Kumar Poddar[a], Dr. Rajneesh Rani[b]

[a]*Dr B R Ambedkar National Institute of Technology, Jalandhar, Punjab, India*
[b]*Dr B R Ambedkar National Institute of Technology, Jalandhar, Punjab, India*

Abstract

With the advent of deep learning in recent years, the integration of computer vision with natural language processing has garnered a lot of attention. Generating descriptions from images is one of the most intriguing and focused areas of Machine Learning that faces a number of obstacles, particularly when describing images in languages other than English. In image captioning, a computer is trained to understand the visual information of an image and then generate a description based on the image features and reference sentences. Having an application that automatically describes events in their environment and then translates them into a caption or message can make a significant contribution to society. This paper presents a multi-layered CNN-LSTM neural network model that is utilized to recognize and generate Hindi captions for the objects in images. In addition, a variety of models were trained by adjusting hyperparameters and the number of hidden layers to find the optimum model and maximize the likelihood of the resultant Hindi description. Moreover, after testing the effectiveness of our models, it was observed that our model has shown an increase of 34.64% and 29.13% in terms of BLEU score (Unigram) and BLEU score (Bigram) respectively when compared to the existing work done in this field. Image captioning in Hindi can have a multitude of applications in today's society, and it can also provide a user-friendly interface for Hindi speakers.

Keywords: Image Captioning; Encoder Decoder Framework; Deep Learning; Convolutional Neural Network; Recurrent Neural Network.

1. Introduction

The concept of automatically generating descriptive words for images has garnered a lot of interest in the last few years. Generating captions for an image is a crucial endeavor that necessitates a semantic

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: author@institute.xxx

1877-0509
10.1016/j.procs.2023.01.049

understanding of images as well as the ability to construct relevant captions. It can be beneficial for a wide array of domains from assisting visually impaired people to understanding the web contents. It also enables companies to analyze enormous amounts of visual data. Recent advances in object identification and language modeling have enabled the generation of meaningful image captions.

Early attempts relied on template methods, in which information derived from images was utilized to fill specified text templates [2]. Deep learning techniques are used in current systems, which take advantage of the available computational power. One of the best methodologies for generating image captions is to use an encoder-decoder architecture. In this methodology, firstly the image features are extracted using pre-trained CNN models like Alexnet, VGG16, DenseNet, etc., and then the captions for the given image is generated using recurrent neural networks [13]. Many researchers have also proposed solutions to improve the existing encoder-decoder architecture. One of them is the U-net inspired model [10] that uses a symmetric encoder-decoder architecture. This model makes use of multilayer CNN and has demonstrated significant improvement in the image detection and segmentation process.

This paper introduces a CNN-LSTM model to automatically recognize objects in images and generate relevant captions for them. It uses Transfer Learning-based models to recognize objects using deep learning approaches. This model has the ability to do two tasks. The first is to recognize key objects in the image using pre-trained CNN models, and the second is to generate descriptions for the image using RNN. The goal of this study is to offer a multi-layer convolutional neural network model and evaluate the results of these models. The flow of the paper is as follows:

Section 2 of the paper will focus on the existing works done in this field. Section 3 outlines our approach, which covers the proposed model as well as the data pre-processing phase for the image and text data. Section 4 discusses the proposed models for the experimentation. In section 5, the experimental design and results are detailed along with a comparison to earlier investigations. Finally, section 6 concludes the paper along with some of the future works that can be implemented on the existing work.

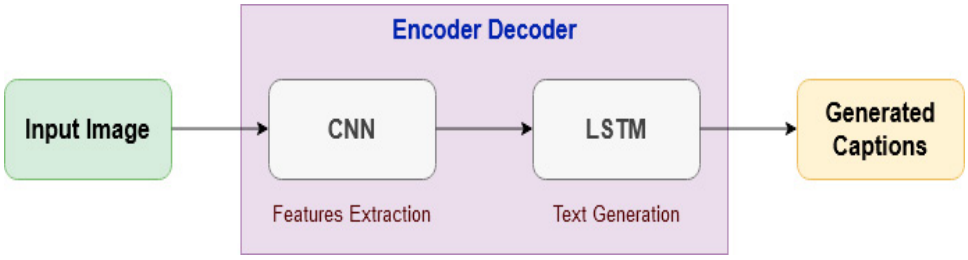


Figure 1. Flowchart of Image Caption Generating Model

2. Related Works

This section examines the state-of-the-art image captioning as well as the various methodologies employed by scholars. Many scholars have recently worked on this domain and offered a variety of methods for generating high-quality captions for images.

In the early phase of image captioning, template-based approaches were formulated. In this approach, there is a fixed number of templates and they are filled on the basis of object detection, scene recognition, attribute categorizations, or other characteristics.

Farhadi et al. [2] proposed a template-based approach where a triplet consisting of object, action, and scene is used to capture the scene elements. Object detection algorithms are utilized initially to estimate the scenes and objects, followed by the use of pre-trained language models to identify prepositions, verbs, and numerous scenarios. Finally, a descriptive caption is created.

Kulkarni et al. [6] suggested a template-based technique that utilizes the conditional random field (CRF) for predicting the best image description. In their approach, a large number of object detectors are firstly used to scan an image and collect the set of high-scoring detections. Thereafter, the CRF is used for generating the descriptions for the image.

The limitation of these template-based methods is that they can only generate captions of a specific length. This led to the rise of the encoder-decoder framework [15]. With the success of the encoder-decoder framework, a variety of neural network-based techniques emerged. This method has garnered a lot of popularity among researchers. Some of the works related to neural network-based approaches include:

Kiros et al. [5] presented an encoder-decoder model that utilizes a feed-forward neural network. Their method predicts the target word based on the extracted features of the image and the preceding word using a multimodal log-bilinear model. In their method, the encoder lets you rank photos and words using the pre-trained CNN (Convolutional Neural Network) models, while the decoder can make up new captions from scratch using the LSTM.

Xiao et al. [16] proposed a three-layer LSTM encoder-decoder model that effectively fused images and textual data to generate relevant descriptions. They investigated how the output from the middle layer of LSTMs can be used to improve the language generating model of the top-most LSTM. Furthermore, they also used a policy gradient optimized strategy to boost their model's performance.

In image captioning, attention mechanisms have recently been incorporated into the existing encoder-decoder neural frameworks. The following are some of the works that utilized attention-based image captioning technique:

Xu et al. [17] suggested a model for generating image descriptions that incorporate visual attention with a single LSTM layer's hidden state. They used typical backpropagation techniques to train their model in a deterministic manner. They also demonstrated how the model can automatically learn to fix its gaze on important items while creating the appropriate sentences in the output sequence through visualization.

Al-Malla et al. [1] proposed an attention-based captioning model. Their image captioning model employs two feature extraction techniques: Xception, a pre-trained CNN model, and YOLOv4, which is an attention module used for detecting objects in images. They also illustrated how the "importance factor" improves the model accuracy by prioritizing forefront large objects over background tiny ones.

Mishra et al. [7] were the first to design a transformer-based architecture for image captioning in the Hindi language. They utilized the MSCOCO dataset, which was then translated into Hindi using the translator. The proposed model uses CNN as the encoder for the feature extraction and the transformer model as a decoder. This model does not have any RNN model, instead, it uses the transformer model. Furthermore, they have also compared their proposed model with numerous baselines. In every baseline, different CNNs and RNNs with spatial attention are used as encoders and decoders respectively.

Rastogi et al. [9] developed a unique method for the classification of leukocytes by utilizing a customized VGG16 feature extractor model. This feature extractor model was trained in 2 steps. In the first step, an augmented dataset with pre-trained layers in a frozen state is utilized to ensure that only the newly added fully connected layer receives weight updates and other layers remain frozen. In the second step, all layers were unfrozen and the model was retrained using the same dataset but with a low learning rate. The experiments showed that this deep learning-based technique can be utilized to extract a wide variety of features with enhanced performance.

3. Detailed Methodology

This section explains the environmental setup and detailed methodology of implementing the image captioning model. Let's explore them in further subsections:

3.1. Environmental Setup

The implementation of the project was performed on a Windows machine having Intel Core i7 processor. The script was written using Python 3.9 in Jupyter Notebook. Moreover, the image captioning model requires a significant amount of memory during the training phase of the model. As a result, all of the experiments are run on 16 GB of RAM. For language modeling, the project also leveraged high-level APIs like Keras. Keras helps to implement almost all neural networks such as CNN and RNN or the combination of both CNN and RNN with ease. Furthermore, some of the python libraries that were used for running the neural network model include Tensorflow, Numpy, Matplotlib, TQDM, Pillow, and NLTK.

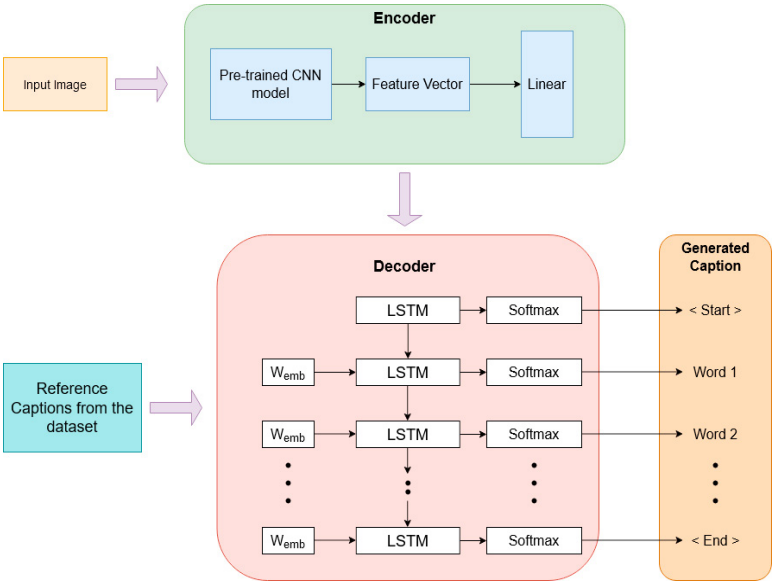


Figure 2. Encoder Decoder Framework

3.2. Phases of Encoder-Decoder model

To generate a caption for a given image, two pieces of information are taken as input: image data and text data. Previous research has demonstrated that combining CNN and RNN can give a rich visual description. As a result, the encoder-decoder model in the proposed study uses the CNN-RNN paradigm. The proposed work employs two neural network models: CNN is used for extracting image features and LSTM is used to translate the image features into sentences. The structure of the image caption generating model is shown in Figure 2.

The encoder-Decoder model for image captioning can be constructed in three key phases:

- Feature Extraction
- Text Preprocessing
- Language Modeling

Before fitting the image and text data into the model, both image and text data must be pre-processed and transformed. In the next subsections, we'll look at how image and textual data are preprocessed in our methods before being fitted to the encoder-decoder model.

3.2.1. Feature Extraction

The image captioning process begins with feature extraction. It reduces the dimensionality of the RGB channel into a latent space representation. Initially, the image is converted into the vector format before being fed into the neural network. A pre-trained CNN [3] is utilized to convert images into fixed-size vectors. VGG-16, DenseNet, MobileNet, and a variety of other pre-trained CNN models are available for feature extraction.

For this experiment, we used the VGG-16 model to test several model structures. Additionally, CNN has been pre-trained to avoid overfitting in image captioning models. The final layer of the CNN model is used to predict the image features. As a result, the CNN's last layer is removed, and the model is now implemented using the second last layer, which returns the image feature in the vector format [14].

3.2.2. Text Preprocessing

Text processing is also one of the key steps that is used for generating high-quality captions. It is necessary to pre-process text data and convert it to a numerical form before feeding it to the neural network.

For the text preprocessing, firstly, all numeric words, special letters, and punctuations are eliminated during the text preprocessing phase. In addition, some of the words that aren't very useful are also eliminated from the lexicon. Secondly, the text data is annotated with the start and end tags. This helps the machine determine where the sentence begins and ends. Below is an example of a caption with a start and end tag.

startseq सड़क पर भूरी बिल्ली देख रही है **endseq**

Moreover, text cannot be processed directly by the neural network. The text must be converted into a numerical form. This can be easily done using the Keras tokenizer.

3.2.3. Language Modeling

In the third step of image captioning, the preprocessed text and the extracted image features are fed into the image captioning model to generate the descriptions. In order to achieve this, different language models such as LSTM, GRU, or a combination of other existing models have been frequently employed in the literature. For this paper, we employed LSTM recurrent networks [13].

The Long Short Term Memory layer is also a type of RNN. It is mostly used to transfer data from one cell to the next and to generate a complete word. It can handle a variety of data, including video and sound, in addition to images. A conventional LSTM consists of three gates i.e. input gate, output gate, and forget gate. These gates of the LSTM control the movement of information. For categorization, processing, and prediction, time-series data is ideal for LSTM networks.

Let's look at how the LSTM is used to generate the image captions for the given image:

Figure 3. An image of a cat

Caption -> सड़क पर भूरी बिल्ली देख रही है।

The vocabulary for the given caption will be:

Vocabulary -> सड़क , पर, भूरी , देख , रही, बिल्ली, है, startseq, endseq

Output Caption: सड़क पर भूरी बिल्ली देख रही है

From table 1, it is clear that firstly, the target word is predicted using the LSTM, and then the target word is concatenated to the partial caption.

4. Proposed Models for our Experimentation

In our experimentation, various models are implemented having different layers and parameters. We have used the VGG-16 as the pre-trained CNN model. Let's look at some of the encoder-decoder models:

4.1. Model 1

The structure of model 1 can be explained as follows:

Table 1. Partial Captions generating in every iteration

Iterations	Image Features Vector	Partial Caption	target Word
1	image vector	startseq	सड़क
2	image vector	startseq सड़क	पर
3	image vector	startseq सड़क पर	भूरी
4	image vector	startseq सड़क पर भूरी	बिल्ली
5	image vector	startseq सड़क पर भूरी बिल्ली	देख
6	image vector	startseq सड़क पर भूरी बिल्ली देख	रही
7	image vector	startseq सड़क पर भूरी बिल्ली देख रही	है
8	image vector	startseq सड़क पर भूरी बिल्ली देख रही है	endseq

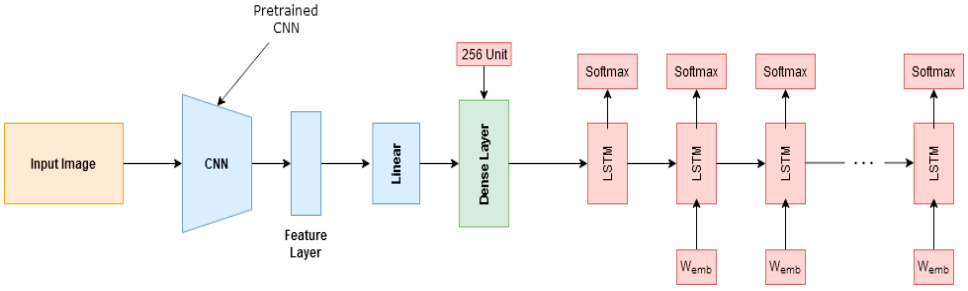


Figure 4. Model 1

1. **Image Feature Input:** Initially, VGG-16 is used to build an image feature vector, which is fed into the encoder model. With this, the image encoder's input shape is set to the size of the extracted image feature. Also, a dropout layer with a dropout rate of 0.3 is added. Finally, using the ReLU activation function in the dense layer, the image feature vector is compressed to 256 elements.

2. **Text Feature Input:** The description of the image is used as a second input in the text encoder model. It predicts an input sequence, which is further transmitted via the encoding layer. Following that, there's a dropout layer with a 0.3 dropout rate. In the final stage, the LSTM layer is employed, which has 256 memory units and produces a 256-element vector output.

3. **Decoder:** Finally, the feature extractor and sequence models are fed into the decoder model. Both input models yield a 256-element vector, which is then added together. The information is subsequently passed on to a 256 neurons dense layer. It then makes a softmax forecast of a word at the end of the process. Thereafter, the model is run for 25 epochs and the Loss vs Epochs graph for model 1 is generated. The graph is depicted in fig 8.

4.2. Model 2

The structure of model 2 can be explained as follows:

1. **Image Feature Input:** Initially, VGG-16 is used to build an image feature vector, which is fed into the encoder model. With this, the image encoder's input shape is set to the size of the extracted image feature. Also, a dropout layer with a dropout rate of 0.5 is added. Finally, using the ReLU activation function in the dense layer, the image feature vector is compressed to 128 elements.

2. **Text Feature Input:** The reference captions of the images are used as input in the text encoder model. It predicts an input sequence, which is further transmitted via the encoding layer. Following that, there's a dropout layer with a 0.5 dropout rate. In the final stage, the LSTM layer is employed,

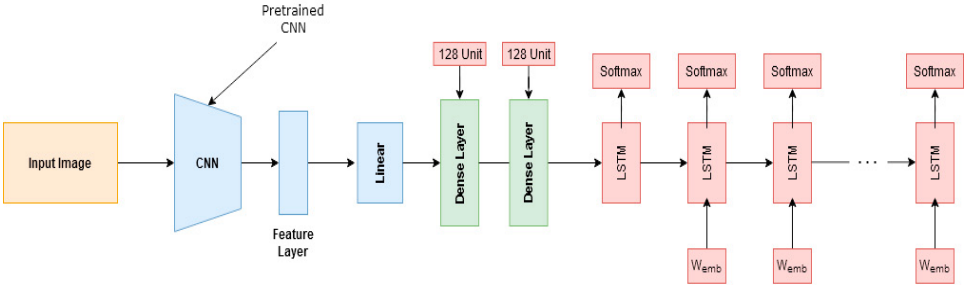


Figure 5. Model 2

which has 128 memory units and produces a 128-element vector output.

3. **Decoder:** Finally, the feature extractor and sequence models are fed into the decoder model. Both input models yield a 128-element vector, which is then added together. The information is subsequently passed on to a 128 neurons dense layer. It then makes a softmax forecast of a word at the end of the process. Thereafter, the model is run for 25 epochs and the Loss vs Epochs graph for model 2 is generated. The graph is depicted in fig 8.

4.3. Model 3

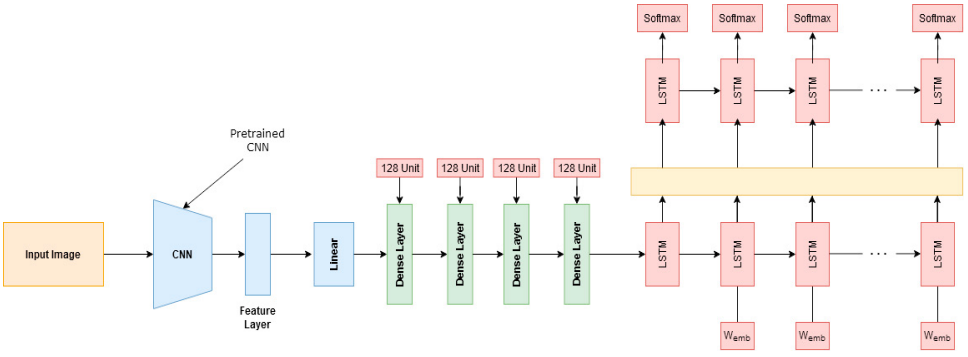


Figure 6. Model 3

The structure of model 3 can be explained as follows:

1. **Image Feature Input:** Initially, VGG-16 is used to build an image feature vector, which is fed into the encoder model. With this, the image encoder's input shape is set to the size of the extracted image feature. Also, a dropout layer with a dropout rate of 0.5 is added. Finally, using the ReLU activation function in the dense layer, the image feature vector is compressed to 128 elements in 4 subsequent dense layers.

2. **Text Feature Input:** The reference captions of the images are used as input in the text encoder model. It predicts an input sequence, which is further transmitted via the encoding layer. Following that, there's a dropout layer with a 0.5 dropout rate. In the final stage, two parallel LSTM layers are employed, which have 128 memory units and produce a 128-element vector output.

3. **Decoder:** Finally, the feature extractor and sequence models are fed into the decoder model. Both input models yield a 128-element vector, which is then added together. The information is subsequently passed on to a 128 neurons dense layer. It then makes a softmax forecast of a word at the end of the

process. Thereafter, the model is run for 25 epochs and the Loss vs Epochs graph for model 3 is generated. The graph is depicted in fig 8.

4.4. Model 4

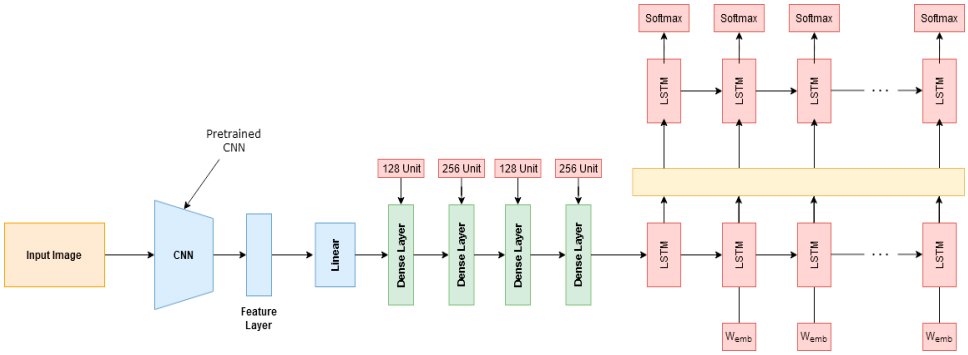


Figure 7. Model 4

The structure of model 4 can be explained as follows:

1. **Image Feature Input:** Initially, VGG-16 is used to build an image feature vector, which is fed into the encoder model. With this, the image encoder's input shape is set to the size of the extracted image feature. Also, a dropout layer with a dropout rate of 0.5 is added. Finally, using the ReLU activation function in the dense layer, the image feature vector is compressed to 128 and 256 elements subsequently in 4 dense layers.

2. **Text Feature Input:** The reference captions of the images are used as input in the text encoder model. It predicts an input sequence, which is further transmitted via the encoding layer. Following that, there's a dropout layer with a 0.5 dropout rate. In the final stage, two parallel LSTM layers are employed, which have 256 memory units and produce a 256-element vector output.

3. **Decoder:** Finally, the feature extractor and sequence models are fed into the decoder model. Both input models yield a 128-element vector, which is then added together. The information is subsequently passed on to a 256 neurons dense layer. It then makes a softmax forecast of a word at the end of the process. Thereafter, the model is run for 25 epochs and the Loss vs Epochs graph for model 4 is generated. The graph is depicted in fig 8.

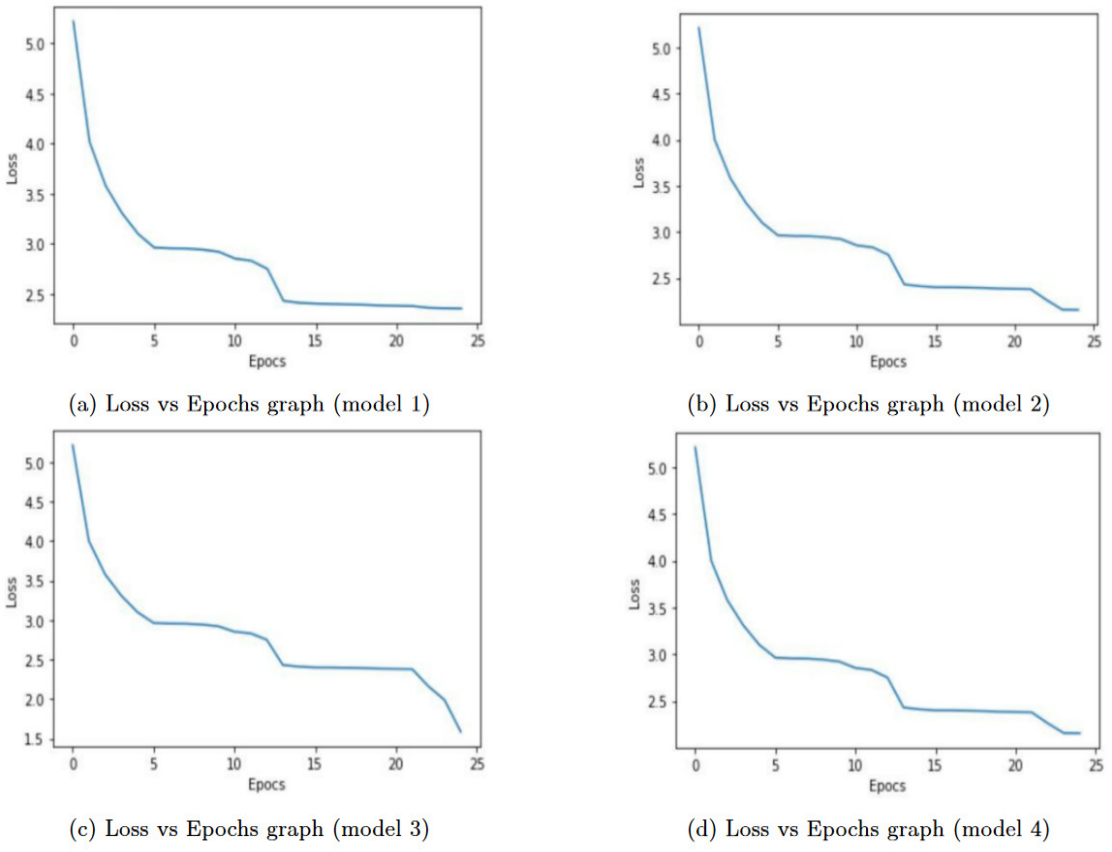
5. Experimental Results and Analysis

This section introduces you to the dataset that is considered for our experiment and its analysis. Later, we demonstrate the effectiveness of our proposed model on the given dataset and present comparisons with the other state-of-the-art works.

5.1. Experimental Results

Experiments were performed on the Flickr8k Hindi dataset. Flickr8k Hindi dataset consists of 5 Hindi captions for each image. This dataset consists of 8,000 training images and 1,000 images are reserved for testing and validation. Moreover, for the experimentation purpose, 25 epochs were run for each of the models, and then the result is generated. The generated loss vs epoch graph is depicted in figure 8.

To test the effectiveness of our model, we have compared the results with A. Rathi's Encoder-Decoder model for the Flickr8k Hindi dataset. Compared with A. Rathi's result, it was observed that our model has achieved a higher BLEU score [8] for the preprocessed image and text data. Thus, our experiment has



(a) Loss vs Epochs graph (model 1)

(b) Loss vs Epochs graph (model 2)

(c) Loss vs Epochs graph (model 3)

(d) Loss vs Epochs graph (model 4)

Figure 8. Generated Loss vs Epochs graph for various models

achieved a state-of-the-art result in terms of the BLEU score. Table 2 depicts the proposed image captioning model's BLEU score in comparison to scores obtained by A. Rathi on the Flickr8k Hindi dataset.

Table 2. Comparison of proposed work with the existing work

Model	Dataset	BLEU Score (Unigram)	BLEU Score (Bigram)
Rathi A. [11]	Flickr8k Hindi Dataset	0.4136	0.278
Proposed_model 1	Flickr8k Hindi Dataset	0.51688	0.29309
Proposed_model 2	Flickr8k Hindi Dataset	0.52022	0.31973
Proposed_model 3	Flickr8k Hindi Dataset	0.55257	0.35166
Proposed_model 4	Flickr8k Hindi Dataset	**0.55698**	**0.35914**

5.2. Results Comparison

In this section, the results obtained by the proposed work are graphically compared with the existing work i.e. A. Rathi's model. From figure 9, it can be observed that our proposed models i.e. model-1, model-2, model-3, and model-4 have shown significant improvements in terms of BLEU scores. Now, let's analyze the resultant graphs in terms of BLEU scores for the Flickr8k Hindi Dataset.

- In comparison to the existing work, it is observed that our model-1 has shown an increase of 24.95% for the BLEU score (Unigram) and 5.39% for the BLEU score (Bigram).

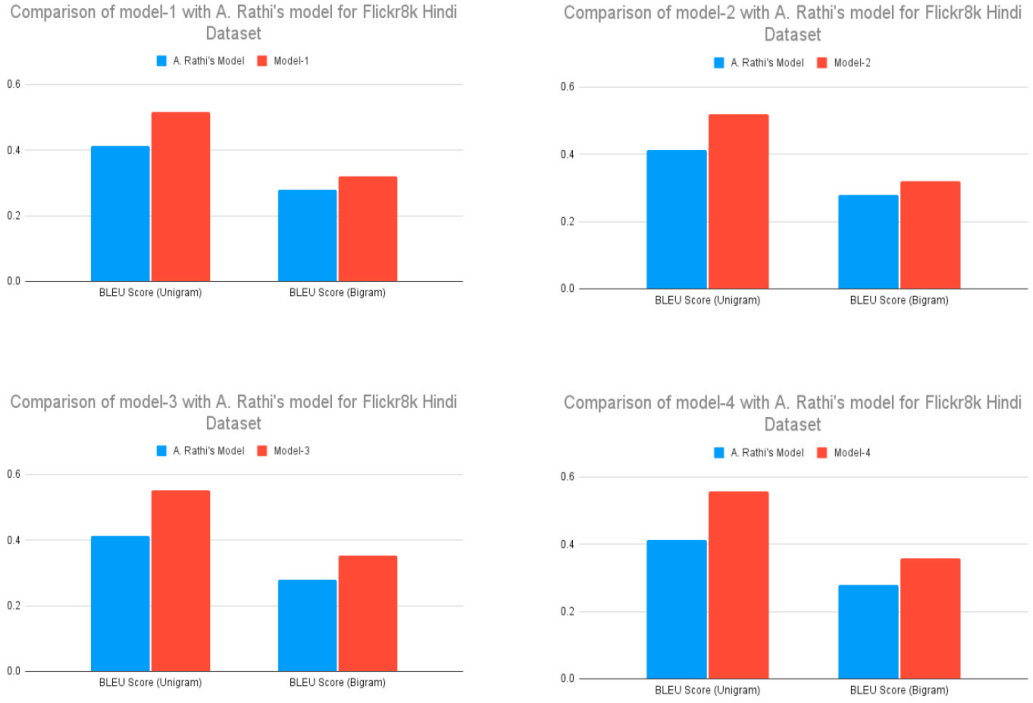


Figure 9. Comparison of proposed models with A. Rathi's model for Flickr8k Hindi Dataset

- In comparison to the existing work, it is observed that our model-2 has shown an increase of 25.77% for the BLEU score (Unigram) and 14.74% for the BLEU score (Bigram).
- In comparison to the existing work, it is observed that our model-3 has shown an increase of 33.58% for the BLEU score (Unigram) and 26.47% for the BLEU score (Bigram).
- In comparison to the existing work, it is observed that our model-4 has shown an increase of 34.64% for the BLEU score (Unigram) and 29.13% for the BLEU score (Bigram).

Thus, from the above comparative analysis, it can be concluded that for the given Flickr8k Hindi Dataset, the model having a higher number of dense and LSTM layers has shown an improved BLEU score in comparison to the traditional CNN-LSTM based encoder-decoder model.

6. Conclusion and Future Scope

This paper proposed an Encoder-Decoder model on the Flickr8k Hindi dataset that uses a pre-trained CNN (VGG16) for feature extraction and uses LSTM for the language modeling. To prove the effectiveness of the Encoder-Decoder framework for the Hindi captioning dataset, the BLEU score was computed on various image captioning models. These models were optimized by tuning the hyperparameters and changing the hidden layers in the existing framework. It is clear from the experimental results and analysis that the multilayered CNN-LSTM neural network model has shown significant improvement in terms of the BLEU score in contrast to the traditional CNN-LSTM neural network model. The findings of this experiment can serve as a benchmark for future research in the field of deep learning.

The work provided here paves the way for further research in this domain. This is simply a first-cut solution, and there are many ways to further enhance it. In the future, this research can be extended by implementing a fine-tuned customized VGG16 model [9] and using some large datasets. Also, integrating object recognition with a convolutional model can be implemented to make the model more robust. The model's efficiency can also be improved by adding an attention module to the existing encoder-decoder

architecture. Thus, despite the fact that image captioning has seen tremendous progress in recent years, there is always scope for improvement in the future.

References

[1] Al-Malla, M.A., Jafar, A., Ghneim, N. (2022) "Image captioning model using attention and object features to mimic human image understanding" Journal of Big Data 9.1 : 1-16.
[2] Farhadi, A., Hejrati, M., Sadeghi, M.A., Young, P., Rashtchian, C., Hockenmaier, J., Forsyth, D. (2010) "Every picture tells a story: Generating sentences from images" European conference on computer vision. Springer, Berlin, Heidelberg.
[3] Gu, J., Wang, G., Cai, J., Chen, T. (2017) "An empirical study of language cnn for image captioning" Proceedings of the IEEE International Conference on Computer Vision.
[4] Kaur, J., Josan, G.S. (2020) "English to Hindi Multi Modal Image Caption Translation" Journal of Scientific Research 64.2.
[5] Kiros, R., Salakhutdinov,R., Zemel, R.S. (2014) "Unifying visual-semantic embeddings with multimodal neural language models" arXiv preprint arXiv:1411.2539.
[6] Kulkarni, G., Premraj, V., Ordonez, V., Dhar, S., Li, S., Choi, Y., Berg, A.C., Berg, T.L. (2013) "Babytalk: Understanding and generating simple image descriptions" IEEE transactions on pattern analysis and machine intelligence 35.12 : 2891-2903.
[7] Mishra, S.K., Dhir, R., Saha, S., Bhattacharyya, P., Singh, A.K. (2021) "Image captioning in Hindi language using transformer networks" Computers and Electrical Engineering 92 (2021): 107114.
[8] Papineni, K., Roukos, S., Ward, T., Zhu, W. (2002) "Bleu: a method for automatic evaluation of machine translation" Proceedings of the 40th annual meeting of the Association for Computational Linguistics.
[9] Rastogi, P., Khanna, K., Singh, V. (2022) "LeuFeatx: Deep learning–based feature extractor for the diagnosis of acute leukemia from microscopic images of peripheral blood smear" Computers in Biology and Medicine 142 (2022): 105236.
[10] Rastogi, P., Khanna, K., Singh, V. (2022) "Gland segmentation in colorectal cancer histopathological images using U-net inspired convolutional network" Neural Computing and Applications 34.7 : 5383-5395.
[11] Rathi, A. (2020) "Deep learning apporach for image captioning in hindi language" 2020 International Conference on Computer, Electrical and Communication Engineering (ICCECE), IEEE.
[12] Srinivasan, L., Sreekanthan, D., Amutha, A.L. (2018) "Image captioning–a deep learning approach" Int. J. Appl. Eng. Res 13.9 : 7239-7242.
[13] Tanti, M., Gatt, A., Camilleri, K.P. (2017) "What is the role of recurrent neural networks (rnns) in an image caption generator?." arXiv preprint arXiv:1708.02043.
[14] Tanti, M., Gatt, A., Camilleri, K.P. (2018) "Where to put the image in an image caption generator" Natural Language Engineering 24.3 : 467-489.
[15] Vinyals, O., Toshev, A., Bengio, S., Erhan, D. (2015) "Show and tell: A neural image caption generator" Proceedings of the IEEE conference on computer vision and pattern recognition.
[16] Xiao, X., Wang, L., Ding, K., Xiang, S., Pan, C. (2019) "Deep hierarchical encoder–decoder network for image captioning" IEEE Transactions on Multimedia 21.11 : 2942-2956.
[17] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y. (2015) "Show, attend and tell: Neural image caption generation with visual attention" International conference on machine learning. PMLR.