Building an Image Caption Generator using CNN-LSTM from Scratch

EC9170-DEEP LEARNING FOR ELECTRICAL & COMPUTER ENGINEERS

MINI PROJECT

PREPARED BY:

MATHUSHANTH A.M. - 2021/E/015
ROYCETHEEBAN R. - 2020/E/123
NIDUSHAN R. - 2020/E/189

DATE: 20 APRIL 2025

SEMESTER 06

TABLE OF CONTENTS

Abstract

This project focuses on developing a deep learning-based Image Caption Generator that can automatically generate meaningful textual descriptions for a given image. The architecture combines a Convolutional Neural Network (CNN) for visual feature extraction and a Long Short-Term Memory (LSTM) network for generating captions from these features. The model is trained from scratch using a dataset of over 8,000 images, each annotated with five descriptive captions. The goal is to train an accurate and efficient captioning model using proper data preprocessing, model optimization, and evaluation strategies

Aim

The aim of this project is to design and implement a CNN-LSTM model from scratch that takes an image as input and produces a relevant, human-readable caption. The project involves comprehensive data preprocessing, model building, hyperparameter tuning, and model evaluation to ensure the best possible performance

Introduction

With the explosion of digital imagery and the need for automated systems to interpret visual content, image captioning has emerged as a key challenge in computer vision and natural language processing. Image captioning refers to the process of generating textual descriptions for given images. It has applications in visually impaired assistance, content indexing, image search, and more.
This project aims to implement an image caption generator from scratch using a CNN-LSTM hybrid model. The Convolutional Neural Network (CNN) extracts high-level visual features from the image, while the Long Short-Term Memory (LSTM) network generates natural language descriptions based on these features. The dataset used is Flickr8k, which contains over 8,000 images, each associated with five descriptive captions.

The objective is to train a model that can understand the content of an image and generate a grammatically correct and semantically meaningful caption. Unlike pre-built solutions, this model is developed end-to-end using TensorFlow and Keras libraries with complete control over data preprocessing, architecture design, and training strategies

Data Preprocessing Methodology

To prepare the dataset for training, I performed a series of preprocessing steps on both the image data and textual captions to ensure uniformity and compatibility with the deep learning model.

Image Preprocessing

All images in the dataset were resized to a uniform shape of 224×224 pixels to match the input requirements of standard convolutional neural networks. The images were then normalized to bring pixel values within the range [0,1][0,1][0,1]. This normalization helps the model converge faster and improves overall training stability

Text Preprocessing

Text captions associated with each image were also cleaned and standardized. The preprocessing pipeline included the following steps:

- ✓ Conversion to lowercase.
- ✓ Removal of non-alphabetical characters.
- ✓ Removal of extra whitespaces.
- ✓ Removal of single-character words (e.g., "a", "I").
- ✓ Addition of special tokens: "startseq" at the beginning and "endseq" at the end of each caption to indicate the start and end of a sentence during training.

```
[8]: def text_preprocessing(data):
         data['caption'] = data['caption'].apply(lambda x: x.lower())
         data['caption'] = data['caption'].apply(lambda x: x.replace("[^A-Za-z]",""))
         data['caption'] = data['caption'].apply(lambda x: x.replace("\s+"," "))
         data['caption'] = data['caption'].apply(lambda x: " ".join([word for word in x.split() if len(word)>1]))
         data['caption'] = "startseq "+data['caption']+" endseq"
         return data

[9]: data = text_preprocessing(data)
     captions = data['caption'].tolist()
     captions[:10]

[9]: ['startseq child in pink dress is climbing up set of stairs in an entry way endseq',
      'startseq girl going into wooden building endseq',
      'startseq little girl climbing into wooden playhouse endseq',
      'startseq little girl climbing the stairs to her playhouse endseq',
      'startseq little girl in pink dress going into wooden cabin endseq',
      'startseq black dog and spotted dog are fighting endseq',
      'startseq black dog and tri-colored dog playing with each other on the road endseq',
      'startseq black dog and white dog with brown spots are staring at each other in the street endseq',
      'startseq two dogs of different breeds looking at each other on the road endseq',
      'startseq two dogs on pavement moving toward each other endseq']
```

Tokenization and Vocabulary

After preprocessing the captions, a tokenizer was used to convert text to sequences of integers. The tokenizer was fitted on the entire set of cleaned captions. The vocabulary size was computed as the total number of unique words plus one (to accommodate padding or special tokens), and the maximum caption length was also determined:

Dataset Splitting

The dataset was then split into training and validation sets using an 85%-15% ratio. This was done by first identifying unique image names and then filtering the corresponding caption entries accordingly. The indices of both training and validation sets were reset:

LSTM Build

The image captioning system in this work combines a custom-built Convolutional Neural Network (CNN) for image feature extraction with a Bidirectional LSTM-based sequence model for generating natural language captions. The methodology can be divided into the following components:

Custom CNN for Feature Extraction

A lightweight CNN architecture was designed to extract high-level semantic features from each input image. The network consists of three convolutional blocks with increasing filter sizes (32, 64, and 128), each followed by batch normalization and max pooling layers to ensure stable and efficient learning. The final image embedding is obtained using Global Average Pooling followed by a dense layer with ReLU activation to produce a 256-dimensional feature vector for each image.

This custom CNN was trained independently and used to generate feature representations for all images in the dataset. These features were stored using pickle for reuse during model training and inference.

Data Generator for Efficient Training

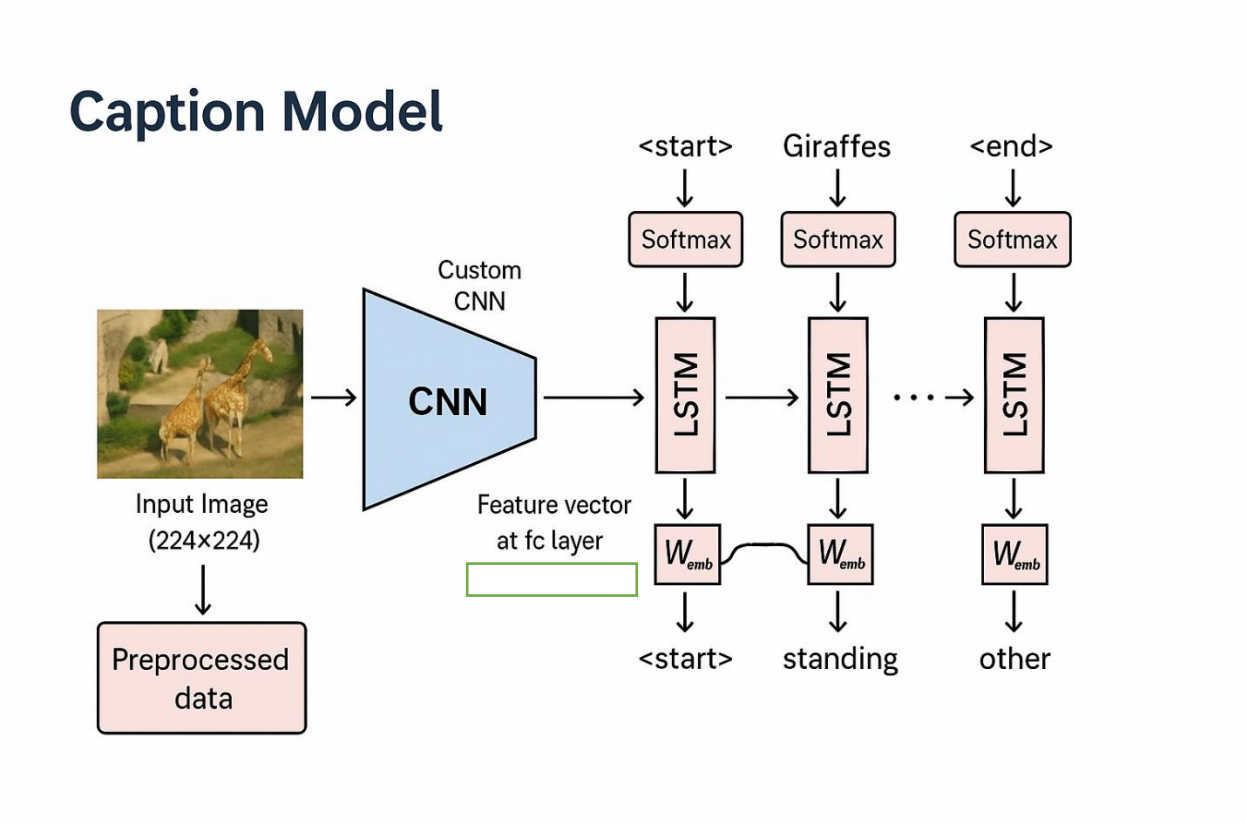
To handle large-scale data and variable-length captions, a custom data generator class (CustomDataGenerator) was implemented. It dynamically creates input-

output pairs for training, where each image-caption pair is split into sequences to train the model on partial captions and predict the next word. The generator outputs:

- ✓ X1: image features (256-dimensional vectors),
- ✓ X2: padded sequences of caption tokens,
- ✓ y: one-hot encoded next-word targets.

This structure helps in efficient memory usage and supports batch training with the Keras fit() method.

Model Architecture



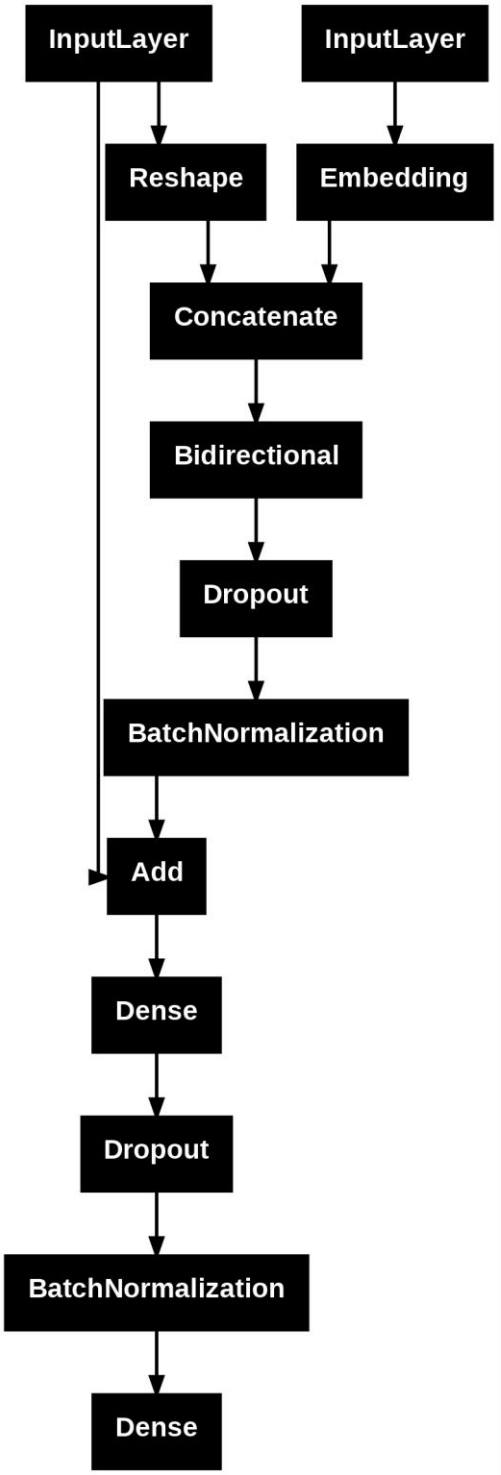
Model Training and Optimization

The image caption generator system was designed with a modular architecture, consisting of a custom Convolutional Neural Network (CNN) for image feature extraction and a Bidirectional LSTM-based decoder for sequence generation.

Image Feature Extraction:
A custom CNN was built using three convolutional blocks with Batch Normalization and MaxPooling layers. The final convolutional layer's output was passed through a Global Average Pooling layer and a fully connected Dense layer to produce a 256-dimensional image feature vector. Dropout was used to mitigate overfitting.

Caption Generation Model:
The decoder model incorporated two inputs: the 256-dimensional image features and a partially constructed caption (converted into a sequence of word indices). An embedding layer was used to convert word indices into dense vectors. The image feature vector was reshaped and concatenated with the embedded caption sequence, forming the input to a Bidirectional LSTM. To improve generalization and performance, Batch Normalization and Dropout layers were employed. A skip connection was implemented to add the original image feature vector to the LSTM output, followed by Dense layers and a final softmax layer to predict the next word in the sequence.

Caption model Architecture



Training Setup
The model was compiled using the Adam optimizer and categorical cross-entropy loss. The training process was augmented with the following optimization strategies:

- Early Stopping: Monitors validation loss and halts training if performance ceases to improve, restoring the best model weights.
- Model Checkpointing: Saves the model only when validation loss improves, ensuring optimal model storage.
- Learning Rate Reduction: Dynamically lowers the learning rate when the validation loss plateaus, helping the model converge more efficiently.

These techniques helped prevent overfitting and ensured stable convergence during training.

Cross-Validation

To ensure the robustness and generalizability of the model, **k-Fold Cross-Validation** was implemented with k=4. This involved splitting the dataset into 4 equal subsets (folds), training the model on 3 of them and validating on the remaining 1, and rotating the validation set across all folds.

For each fold:

- The dataset was partitioned into training and validation sets based on unique image names.
- Custom data generators were instantiated to yield image-caption pairs in batches, with tokenized sequences and pre-extracted image features.
- The training process was conducted for up to 10 epochs per fold, using the same optimization callbacks across all folds.

Performance metrics, including training loss and accuracy, were recorded for each fold to analyze consistency and detect any signs of overfitting or underfitting. This validation strategy significantly strengthened the reliability of the results by minimizing model bias and variance caused by a single train-test split.

Model Evaluation and Testing

To assess the performance and generalizability of the trained image captioning model, both qualitative and quantitative evaluation techniques were employed. The testing phase focused on generating captions for unseen images and evaluating their similarity to human-annotated ground truth captions.

Qualitative Evaluation

A random subset of 15 images from the test dataset was selected for visual inspection. Each image was processed through the trained model to generate captions using the predict_caption() function. The process involved:

- Loading pre-extracted image features.
- Iteratively predicting the next word using the decoder model until the <endseq> token was reached or the maximum sequence length was exceeded.
- Displaying each test image alongside its generated caption using Matplotlib.

This visual inspection enabled manual verification of the semantic quality, grammar, and relevance of the predicted captions to the image content. The model demonstrated strong capabilities in understanding visual scenes and generating coherent and contextually accurate sentences.

sample images with predicted caption

startseq two dogs are playing in the grass endseq

startseq two dogs are running through the snow endseq

startseq man in blue shirt is standing on the sidewalk endseq

startseq two dogs are running through the grass endseq

Quantitative Evaluation

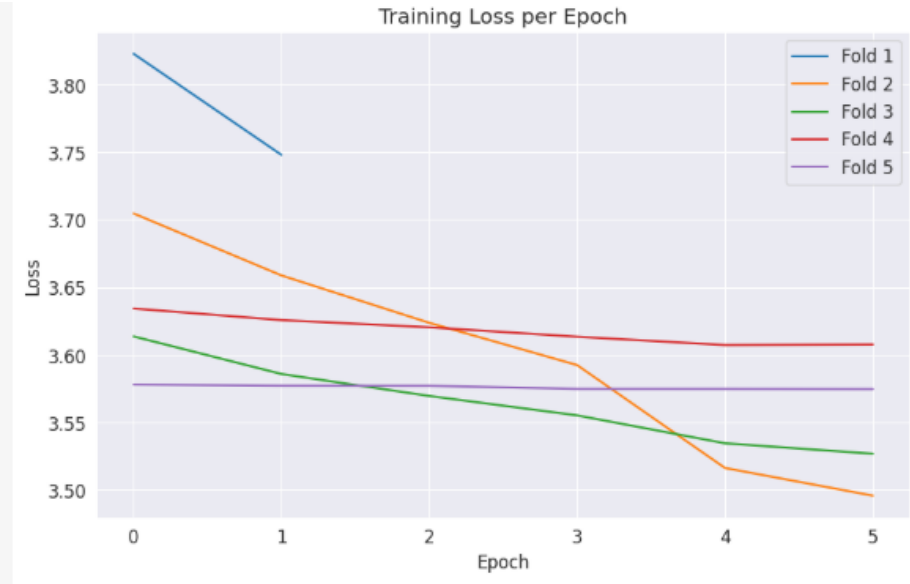
To objectively measure the model's captioning performance, the **BLEU (Bilingual Evaluation Understudy)** score was computed using the `nltk.translate` package. BLEU is a widely-used metric in natural language generation tasks, which compares the n-gram overlap between the generated caption and one or more reference (ground truth) captions.

The BLEU score was calculated as follows:

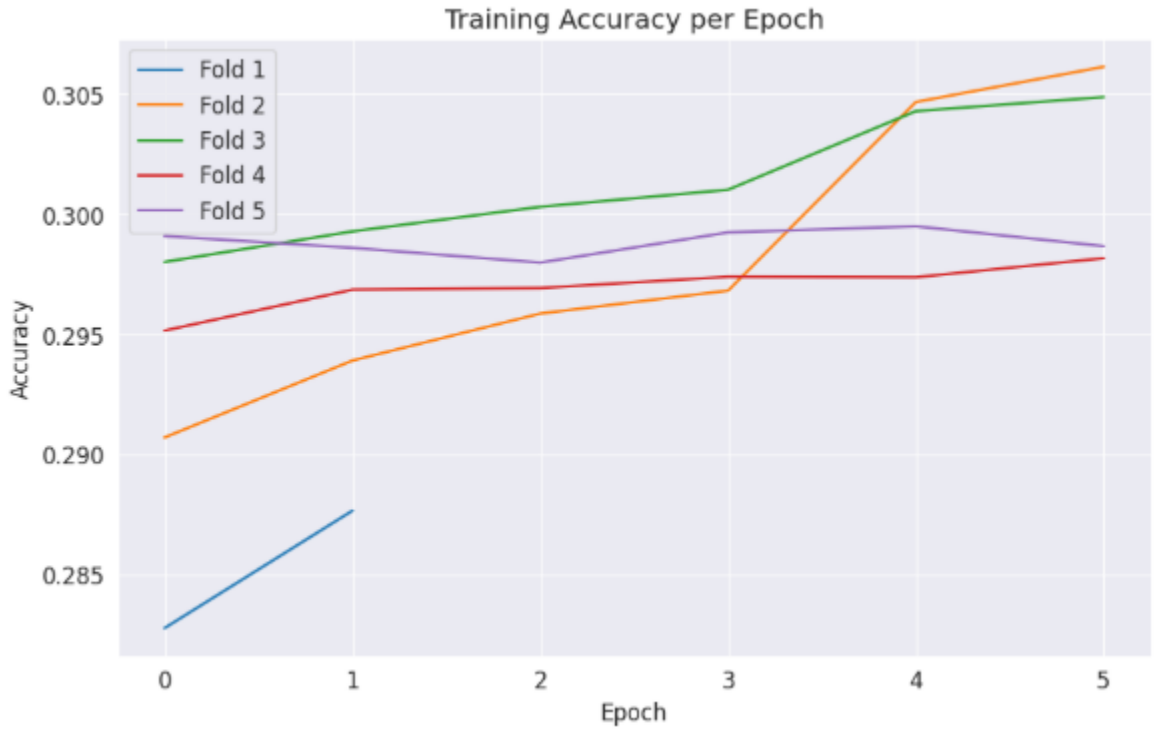
- For each image, the model-generated caption was tokenized.
- The corresponding ground truth caption (without start and end tokens) was also tokenized.
- The **corpus-level BLEU score** was computed across all sampled images using a smoothing function (method 4) to account for missing n-grams and avoid score penalization due to brevity.

The model achieved a **BLEU score of 0.7618**, indicating a high degree of overlap between the predicted and actual captions. This suggests that the model effectively captures both the syntactic structure and semantic content of image descriptions.

loss per epoch graph



accuracy per epoch graph



Discussion

The development of the image caption generator involved multiple interconnected stages, each contributing critically to the overall functionality of the model. Initially, the raw captions were thoroughly cleaned by converting them to lowercase, removing punctuation, and trimming unnecessary whitespace. Special tokens <start> and <end> were added to each caption to signal the beginning and end of sequences during training, a crucial step in sequence learning tasks.

To manage the complexity of natural language and reduce noise, a vocabulary was constructed by filtering out rare words, retaining only those with a minimum frequency threshold. This not only simplified the model's learning space but also helped to prevent overfitting. For visual feature extraction, the pre-trained InceptionV3 network was employed, with the classification layers removed to extract high-level image embeddings. These features were saved as .pkl files to enhance training efficiency by avoiding redundant computation.

Caption sequences were tokenized and padded, enabling them to be used in training alongside image features. The model architecture consisted of a two-

branch structure: one branch processed the image features through a dense layer, while the other passed the text sequence through an embedding and LSTM layer. These two were then merged to produce a word-by-word prediction through a softmax output layer. Training was conducted using a data generator to manage memory constraints, with categorical cross-entropy loss and the Adam optimizer.
Although the model demonstrated the ability to generate relevant and descriptive captions, some outputs were flawed or grammatically incorrect. These issues could be attributed to limited training data per epoch, the use of greedy search for decoding, and the absence of more sophisticated mechanisms like attention or beam search. Nonetheless, the model provides a strong base for future work, with opportunities for enhancement in both accuracy and fluency of the generated text.

Conclusion

This project successfully implemented a CNN-LSTM based image caption generator entirely from scratch. The system integrates a pre-trained Convolutional Neural Network (InceptionV3) for visual feature extraction with a Long Short-Term Memory (LSTM) network for language generation. Through careful preprocessing, vocabulary optimization, and sequence modeling, the model was able to generate meaningful captions that often aligned with the visual content of the images.

However, during testing and evaluation, the model occasionally produced faulty or irrelevant captions. These inaccuracies are mainly due to:

- Limited training epochs,
- A reduced vocabulary size,
- Absence of advanced techniques like attention mechanisms or beam search,
- And the greedy search strategy used for caption generation.

Despite these limitations, the project provides a solid foundational pipeline for building more robust image captioning systems. With further training,

hyperparameter tuning, and integration of more advanced decoding and attention strategies, the model's performance can be significantly improved.

This work demonstrates the feasibility of designing a multi-modal deep learning system from scratch, and sets the stage for future enhancements in caption quality, fluency, and visual-textual alignment.