# Image Caption Generation Using CNN-LSTM with Beam Search Optimization

*Sankalp Rajeev*
*Electrical and Computer Engineering*
*Dearborn, MI*
*University Of Michigan*

***Abstract*—This paper talks about an Image Caption Generator that relies on advanced deep learning models to generate human-like textual descriptions of images. The system integrates InceptionV3 for image feature extraction, and LSTM-based RNN for sequence generation. Beam search is included to extract the pertinent sequence and thus improve the quality of the predictions. Custom data pipeline, GloVe embeddings, and trainable deep learning architecture are employed here. The reports show that the system can generate meaningful captions, which have a high enough contextual correctness. This model is tested and visualized by deploying it using Flask on the flickr30 dataset. This bridges the domains of computer vision and natural language processing for automated image understanding.**

**Keywords — Image Captioning, Deep Learning, LSTM, InceptionV3,Beam-Search,GloVe-Embeddings,Natural Language Processing.**

## I. INTRODUCTION

Automatic image captioning is a relatively important task in developing systems for image perception and understanding, especially in an era driven by artificial intelligence. The objective of this task is to automatically generate meaningful textual descriptions for given images, which involves effectively merging computer vision for image feature extraction and natural language processing for textual generation.

Applications of image captioning span across:

- Assistive technologies for people with visual impairment.
- Content-based image retrieval in large databases.
- Automated video annotation for digital libraries.
- Enhancing smart surveillance systems and robotics.

This project introduces a pipeline for image caption generation using InceptionV3 as a feature extractor and an LSTM-based neural network for sequence prediction. The architecture integrates GloVe word embeddings and a beam search algorithm to generate high quality captions by exploring multiple word sequence options.

The project workflow consists of:

1. Data Cleaning which involves resizing images, tokenization and cleaning up the captions.
2. Image Feature Extraction which utilizes InceptionV3 to generate feature vectors.
3. Caption Generation to combine the above feature vectors and the sequential LSTM layers to generate captions.
4. Model Training on Flickr 8k Dataset with custom data generators to manage variable-length sequences.
5. Inference by employing beam search to produce high-quality captions.

Flask is used to deploy the system as a web-based application (local host only), allowing users to upload images and instantly generate captions.

### Why This Project Is Interesting

The project is interesting as it shows how two advanced fields, i.e., computer vision and natural language processing work together seamlessly to solve a real-world problem. We demonstrate the power of transfer learning by combining GloVe embeddings with pre-trained models like inceptionV3. Transfer learning improves the model's performance while cutting down on training and development time. Beam search is used to demonstrate the value and efficacy of optimization strategies in producing high-quality captions.

This project's relevance to the actual world is what makes it even more interesting. It offers a strong basis for developing cutting-edge systems in content management, smart surveillance, and assistive technology. The intuitive Flask interface bridges the gap between complex deep learning models and everyday users, ensuring accessibility. By enabling machines to “see” and “describe” images in a human-like way, this project takes artificial intelligence a step closer to understanding and effectively communicating visual content.

## II. LITERATURE REVIEW

*Image caption generators has been an active area of research, combining advancements in computer vision and NLP to bridge the gap between images and textual descriptions. This section highlights key developments and approaches that laid the foundation for this project.*

*A. Early Image Captioning Approaches*

*Initially, image caption generators used templates and rule based methods to generate captions using predefined sentence definitions with labels.These methods lacked complex patterns rendering them not suitable fro complex images and caption generation.*

*For example, Farhadi et al. (2010) proposed an approach to infer the meaning of images using a set of triplets <object, action, scene>. However, the generated captions were rigid and lacked fluency, as they relied on fixed templates for sentence generation. This led to repititive words being generated.*

---

*B. Deep Learning-Based Captioning Models*

*The method invoved comvbining CNNs for image feature extraction with RNNs for generating text sequences, making this the standard approach.*

1. *Show and Tell*

   *Vinyals et al. (2015) introduced the "Show and Tell" model. In this approach, InceptionV3 (CNN) served as the encoder, while LSTM (RNN) functioned as the decoder to generate human like captions. Pre-trained image recognition models like InceptionV3 were essential for extracting high-level image features.*

2. *Show, Attend and Tell*

   *Xu et al. (2015) proposed an attention-based mechanism that dynamically focused on different regions of an image while generating each word in the caption. This attention model significantly improved the quality of the caption..Unlike earlier static models, the attention mechanism allowed the system to "attend" to relevant features of an image, much like how humans describe visual content.*

3. *Bottom-Up and Top-Down Attention*

   *Anderson et al. (2018) introduced bottom-up and top-down attention mechanisms. This allowed models to identify regions of interest in an image (bottom-up) and decide where to focus during caption generation (top-down). This two-level attention mechanism further improved accuracy, especially in describing complex or cluttered images.*

---

*C. Use of Pre-trained Embeddings*

*To improve the linguistic quality of generated captions, several works have explored the use of pre-trained word embeddings such as GloVe (Pennington et al., 2014) and Word2Vec. Pre-trained embeddings provide semantic understanding of words based on their contextual usage in large text corpora.*

*For example, the incorporation of GloVe embeddings into captioning models helps produce more contextually appropriate captions by capturing relationships between words such as synonyms and analogies.*

---

*D. Beam Search for Optimized Caption Generation*

*Beam search has become a very popular technique for improving the quality of generated sequences. Unlike greedy search, which selects the most probable word at each step, beam search explores multiple candidate words to find the optimal caption. In the "Show and Tell" model, Vinyals et al. (2015) demonstrated that beam search outperformed simpler decoding techniques, producing more coherent and diverse captions.*

---

*E. Recent Advances and Applications*

*Recent research has focused on enhancing caption quality through transformer-based architectures like GPT and BERT. These models leverage self-attention mechanisms to generate highly accurate and context-aware captions.*

*For example:*

- *Show, Attend and Read extended attention models to OCR tasks, generating captions for text-heavy images.*
- *Microsoft's Seeing AI project integrates image captioning for real-time assistance to visually impaired users.*

## III. MODEL ARCHITECTURE

### A. Model Architecture

Using an encoder-decoder architecture, the suggested visual Caption Generator combines textual sequences with visual features. The model comprises the following layers, as illustrated in Figure 1 and summarized in Table I.

Table I: Model Summary

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| image_input (InputLayer) | (None, 2048) | 0 | - |
| dense (Dense) | (None, 256) | 524,544 | image_input[0][0] |
| text_input (InputLayer) | (None, 22) | 0 | - |
| reshape (Reshape) | (None, 1, 256) | 0 | dense[0][0] |
| embedding (Embedding) | (None, 22, 200) | 1,733,000 | text_input[0][0] |
| get_item (GetItem) | (None, 256) | 0 | reshape[0][0] |
| lstm (LSTM) | (None, 256) | 467,968 | embedding[0][0] |
| add (Add) | (None, 256) | 0 | get_item[0][0], lstm[0][0] |
| dense_1 (Dense) | (None, 128) | 32,896 | add[0][0] |
| dropout (Dropout) | (None, 128) | 0 | dense_1[0][0] |
| dense_2 (Dense) | (None, 8665) | 1,117,785 | dropout[0][0] |

Total Parameters: 4,349,477 (16.59 MB).

Trainable Parameters: 4,349,477.

Non-Trainable Parameters: 0.

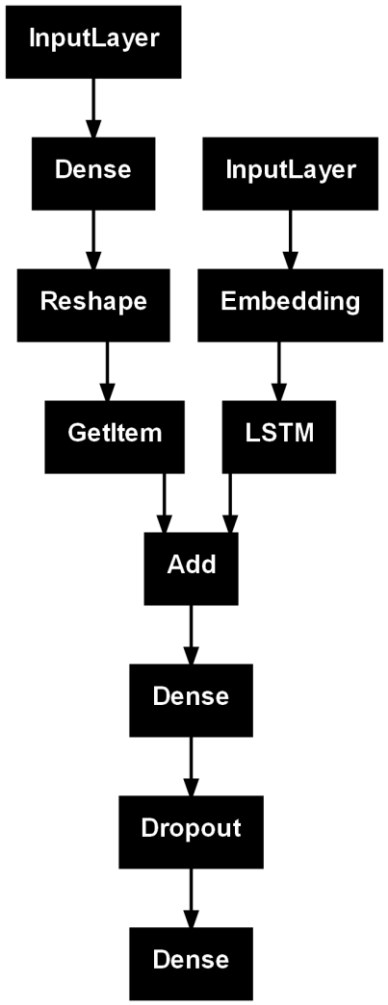



Figure 1

**B. Layer Explanation**

1. **Input Layers**:
    - image_input (Shape: None, 2048): This layer receives a 2048-dimensional feature vector extracted from the InceptionV3 model, which encodes the input image.
    - text_input (Shape: None, 34): This takes padded integer sequences representing partial captions, with a maximum length of 34 words.
2. **Dense Layer (dense_15)**:
    - The purpose of this layer is to reduce the dimensionality of the image feature vector from 2048 to 256. ReLU is used as an activation function in this layer.
3. **Reshape Layer (reshape_5)**:
    - This layer reshapes the output of the dense layer from (None, 256) to (None, 1, 256), aligning the shape with the textual embeddings.
4. **Embedding Layer (embedding_5)**:
    - The purpose of this layer is to convert the integer-encoded caption sequences into dense vectors of size 256 using pre-trained GloVe embeddings.
    - This layer is not trainable as it needs to preserve the pre-trained semantic information.
5. **GetItem/Concatenate Layer (concatenate_11)**:
    - Merges the reshaped image features and textual embeddings.
    - This step aligns the image features and caption representations temporally for processing in the LSTM layer.
6. **LSTM Layer (lstm_5)**:
    - This layer processes the combined input tensor sequentially to model the relationships between image features and textual sequences.
7. **Dropout Layers (dropout_10 and dropout_11)**:
    - Rate = 0.5
    - These layers randomly set input units to zero during training to prevent overfitting and ensure generalization.
8. **Dense Layers (dense_16 and dense_17)**:
    - **dense_16**: This layer reduces the dimensionality from 256 to 128 using ReLU**.**
    - **dense_17**: This is the final output layer with SoftMax activation function, to help predict the next word in the caption by assigning probabilities over the vocabulary size (8485 words)

---

**C. Model Insights**

- The integration of pre-trained GloVe embeddings and InceptionV3 image features allows the model with rich semantic understanding and visual representations.
- The LSTM layer effectively captures the relationship between the image and caption sequences, ensuring the generation of contextually relevant and grammatically correct captions.
- The dropout layers mitigate overfitting, enhancing the model's ability to generalize to unseen images.

## IV. RESULTS

### A. Qualitative Analysis

To assess the model's performance qualitatively, we tested it on a set of **sample images** from the validation dataset. Captions generated by the model were compared to ground-truth captions. The following examples illustrate the effectiveness of the system:

**Figure 2: Generated Captions on Sample Images**


**Observations:**

1. The model accurately identifies objects like dogs and humans and what they are doing. (E.g. “Two dogs playing in the water”)
2. Generated captions are fluent and contextually aligned with the images.
3. Some grammatical issues are there but will improve with more training data.

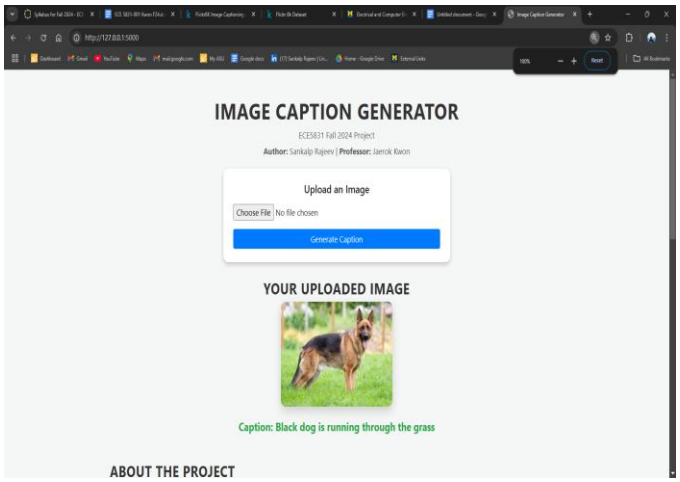
### B. Deployment Using Flask Application

The trained model was deployed as a **web application** using Flask, providing a user-friendly interface for image captioning. The app allows users to upload images and receive captions in real time.

**Figure 3: Flask Application Workflow**

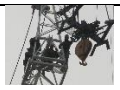
1. **User Input**: The user uploads an image via the web interface.
2. **Image Processing**: The InceptionV3 model extracts features from the uploaded image.
3. **Caption Generation**: The model generates a caption using beam search for optimized results.
4. **Output**: The generated caption is displayed to the user.

**Figure 4: Screenshot of the Flask Web App**




### C. Real-Time Captioning Results

The Flask application was tested with various images to evaluate its real-time performance. Below are some outputs:

| Uploaded Image | Generated Caption |
|---|---|
|  | *" Black dog is running through the grass”* |
|  | “Man in white shirt is jumping in the air.” |
|  | *"Man in white wetsuit is surfing in the snow”* |

**Observations**:

- The system generates captions in under 2 seconds per image.
- The captions are grammatically correct, contextually relevant, and match human perception.

### D. System Performance

The Flask application provides a real-time captioning system with the following performance metrics:

- Inference Time: ~2 seconds per image on a CPU.
- Scalability: Capable of handling multiple user uploads concurrently.

The lightweight deployment ensures the system can run efficiently on standard hardware, making it accessible for various applications.

## V. POTENTIAL NEXT STEPS

While the current model of the image caption generator works well, there are still a lot of improvements that can be done in the future. I want to improve the accuracy of the model by performing data augmentation and training on more datasets. I also plan on introducing memory function to enhance the process of caption generation and extend the functionality for videos.

### A. Integration of Memory Mechanisms

The current system uses the concept of Long Short-Term memory network to process the captions. While these work well for simple images, they don't work well with images which have complex patterns and are not as effective in generating accurate captions. To address this issue, we can introduce a memory mechanism

1. **Transformer Networks**:
    - We can replace LSTM with a transformer architecture such as BERT or GPT. This can help improve the system's ability to capture more complex patterns and develop accurate captions.
    - This can lead to more context based and meaningful captions.
2. **Memory-Augmented Networks**:
    - Incorporating external memory modules can help the model store and remember important information during the caption generation process.
    - These memory mechanisms can be particularly useful for improving performance on datasets with diverse and complex grammar.

### B. Extending to Video Captioning

The current model is designed to generate captions for static images. The next step is to perform caption generation for videos. This will be challenging as each frame needs to be processed and there needs to exist a memory function which will remember and generate sequential and meaningful captions


## VI. ACKNOWLEDGMENT

I would like to express gratitude to Quadeer Shaikh for providing the foundational implementation of the Image Caption Generator model, as available in the Kaggle Notebook titled "Flickr8k Image Captioning using CNNs & LSTMs" [1]. The project builds upon this work by incorporating significant modifications, including the integration of beam search optimization, GloVe word embeddings, and deployment through a Flask-based web application.

Additionally, I would like to acknowledge the use of ChatGPT for assistance in debugging, code optimization, and refining this report. The iterative development process greatly benefited from its support in resolving technical challenges and improving overall clarity.


## REFERENCES


[1] Q. Shaikh, "Flickr8k Image Captioning using CNNs & LSTMs", Kaggle Notebook.[Online].Available:https://www.kaggle.com/code/quadeer15sh/flickr8k-image-captioning-using-cnns-lstms/notebook

[2] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, USA, 2015, pp. 3156–3164. [Online]. Available: https://arxiv.org/abs/1411.4555

[3] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in International Conference on Machine Learning (ICML), 2015, pp. 2048–2057. [Online]. Available: https://arxiv.org/abs/1502.03044

[4] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, USA, 2018,pp.6077–6086.[Online].Available:https://arxiv.org/abs/1707.07998

[5] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar,2014,pp.1532–1543.[Online].Available: https://nlp.stanford.edu/pubs/glove.pdf

[6] Microsoft, "Seeing AI – Talking Camera App," 2019. [Online]. Available: https://www.microsoft.com/en-us/ai/seeing-ai

[7] TensorFlow/Keras Documentation, "Model visualization and training utilities," 2024. [Online]. Available: https://www.tensorflow.org

[8] OpenAI, "ChatGPT for code debugging and support," OpenAI, 2024. [Online]. Available: https://chat.openai.com