Group 13 - Image Captioning

Course: DEEP LEARNING I

Group Members:

- Pujan Bhatt
- Ryan Chan
- Sujay Macwan
- Rohit
- Shiva Kher

Date: 2024-06-06

TABLE OF CONTENTS

1.0 INTRODUCTION

Image captioning is the process of automatically generating textual descriptions for images using machine learning techniques. This sophisticated task lies at the intersection of computer vision and natural language processing (NLP), requiring models to understand and translate visual content into coherent and contextually appropriate language.

1.1 Importance and Applications

- **Enhancing Accessibility for Visually Impaired Users:**

  By providing descriptive captions for images, visually impaired users can better understand and interact with visual content on the web and in applications. This can be particularly beneficial for navigating websites, social media, and digital documents.

- **Improving Image Search Engines:**

  Image captioning can enhance the functionality of image search engines by enabling more accurate and relevant search results. By generating descriptive text, search engines can better index and retrieve images based on user queries.

- **Assisting in Automated Content Creation:**

  In the realm of social media and news articles, image captioning can aid in automating the creation of engaging and relevant content. This can save time for content creators and ensure consistency in descriptions across platforms.

1.2 Intersection of Computer Vision and NLP

Image captioning exemplifies the synergy between computer vision and natural language processing through the use of a dual-model approach:

- **Dual-Model Approach:**
  - **Convolutional Neural Networks (CNNs):** These are used for image feature extraction. CNNs excel at processing visual data and can identify and encode the important features and elements of an image into a fixed-length vector representation, often referred to as an embedding.
  - **Long Short-Term Memory Networks (LSTMs):** These are utilized for text generation. LSTMs are a type of recurrent neural network (RNN) well-suited for sequential data and can generate coherent sentences by maintaining context over longer sequences.

- **Process Workflow:**
    - **Image Feature Extraction with CNNs:** The process begins with a CNN, which analyzes the input image to produce a high-dimensional vector embedding. This embedding encapsulates the essential visual characteristics of the image, such as objects, actions, and scenes.
    - **Word Embeddings and LSTM Integration:** The vector embedding from the CNN is then combined with word embeddings, which are numerical representations of words in a continuous vector space. These combined embeddings are fed into an LSTM network.
    - **Generating Descriptive Text:** The LSTM processes the combined embeddings to generate a sequence of words that form a descriptive caption for the image. The LSTM's ability to maintain and utilize context over sequences enables it to produce more accurate and fluent descriptions.

1.3 Approach

1.3.1 Encoder-Decoder Framework

- **Input Image Encoding**: The input image is encoded into a vector representation using a Convolutional Neural Network (CNN). This vector captures the salient features of the image.
- **Decoding into Text Sequence**: The encoded vector is then decoded into a text sequence using a Recurrent Neural Network (RNN), typically a Long Short-Term Memory (LSTM) network, which generates a descriptive sentence word by word.

1.3.2 Use of CNN for Feature Extraction

- **Feature Extraction**: A pre-trained CNN (such as VGG16, ResNet, or Inception) is used to extract vector embeddings from the input image. The CNN processes the image through multiple convolutional layers, capturing different levels of abstraction.
- **Vector Embeddings**: The output from a certain layer (often the last fully connected layer before classification) is taken as the image embedding, which is a fixed-length vector representing the image.

1.3.3 Use of LSTM for Text Generation

- **Text Generation**: An LSTM network generates text by processing the image embeddings along with the word embeddings of previously generated words. This is done iteratively to produce the entire caption.
- **Concatenated Embeddings**: At each step, the LSTM takes as input the concatenated vector of the image embedding and the embedding of the previously generated word (or a start token for the first word).
- **Sequence Generation**: The LSTM processes this concatenated vector to produce a probability distribution over the vocabulary for the next word, from which the most likely word is selected.

1.3.4 Detailed Workflow

- **Image Encoding (CNN)**
  - Input: Image.
  - Process: Pass the image through a CNN to extract feature maps.
  - Output: Vector embedding representing the image.

- **Text Decoding (LSTM)**
  - Input: Image embedding and word embeddings.
  - Initial Step: The image embedding is used as the initial state of the LSTM.
  - Recurrent Steps: For each time step, the LSTM takes the embedding of the previously generated word (starting with a special start token) and generates the next word in the sequence.
  - Output: Sequence of words forming the caption.

2.0 METHODOLOGY

The methodology section outlines the structured approach we followed to develop an image captioning model. This project leverages advanced deep learning techniques to generate descriptive captions for images. Our methodology encompasses the selection and preprocessing of data, the design and architecture of the model, the training process, and the evaluation of the model's performance. Below, we provide a general overview of each step involved in our methodology.

2.1 Data collection and preprocessing

Image captioning involves generating textual descriptions for images using machine learning techniques, combining the fields of computer vision and natural language processing (NLP).

2.1.1 Dataset

The dataset is a crucial component of our image captioning project. It consists of two primary types of data:

Text-based data:

- This includes images paired with descriptive captions. For example, a caption might describe an image as "A child in a pink dress is climbing up a set of stairs."
- Such captions help the model learn to accurately describe similar images by providing contextual information.

Example Table:

Image	Caption
1000268201_693b08cb0e.jpg	A child in a pink dress is climbing up a set of stairs.
1000268201_693b08cb0e.jpg	A girl going into a wooden building.
1000268201_693b08cb0e.jpg	A little girl climbing into a wooden playhouse.
1000268201_693b08cb0e.jpg	A little girl climbing the stairs to her playhouse.
1000268201_693b08cb0e.jpg	A little girl in a pink dress going into a wooden house.

Image-based data:

- This consists of images with concise descriptions such as "Man in black shirt is playing guitar" or "Construction worker in orange safety vest, working on road."
- These descriptions enrich the dataset, providing diverse scenarios for the model to learn from.

Example Images with Captions:

- "Man in black shirt is playing guitar."
- "Construction worker in orange safety vest, working on road."

2.1.2 Data preprocessing

Data preprocessing is an essential step to prepare the dataset for training the image captioning model. The preprocessing steps undertaken include:

Reading images and captions:

- Images and their corresponding captions are loaded from the dataset.

Text Cleaning Steps:

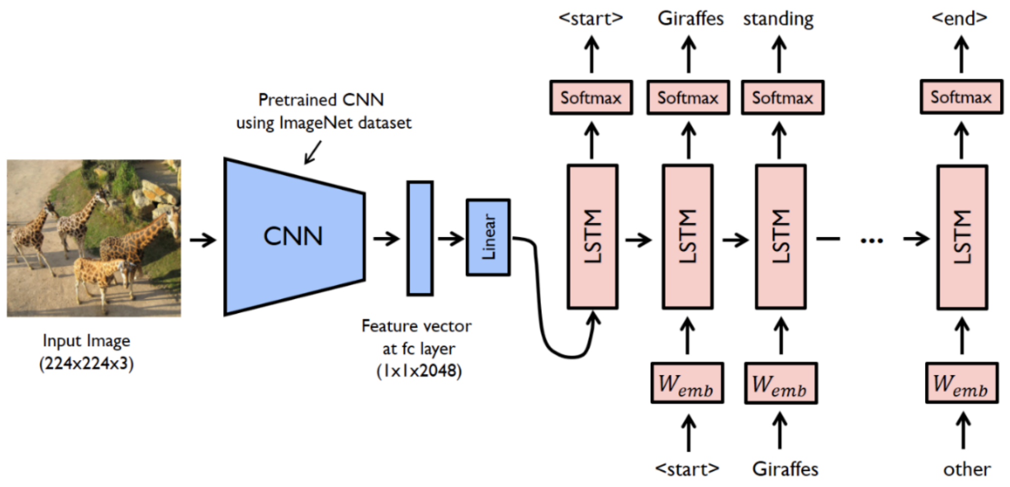
- **Convert Text to Lowercase:** Ensures consistency across all captions.
- **Remove Special Characters and Extra Spaces:** Eliminates noise and irrelevant data.
- **Remove Single Characters:** Removes characters that do not contribute meaningfully to the captions.

Adding Start and End Tags to Captions:

- Start and End Tags: 'startseq' is added at the beginning and 'endseq' at the end of each caption. These markers are crucial as they help the LSTM model recognize the boundaries of each caption.
- Importance of Tags: Without these tags, the model would face challenges such as unclear sentence boundaries, training inefficiency, and issues during inference. The tags help the model start and stop the text generation process accurately, ensuring the production of coherent and contextually appropriate captions.

2.2 Model Architecture

Our model architecture for the image captioning project was a combination of a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) network. This dual-model approach leveraged the strengths of both CNNs for image feature extraction and LSTMs for text generation.

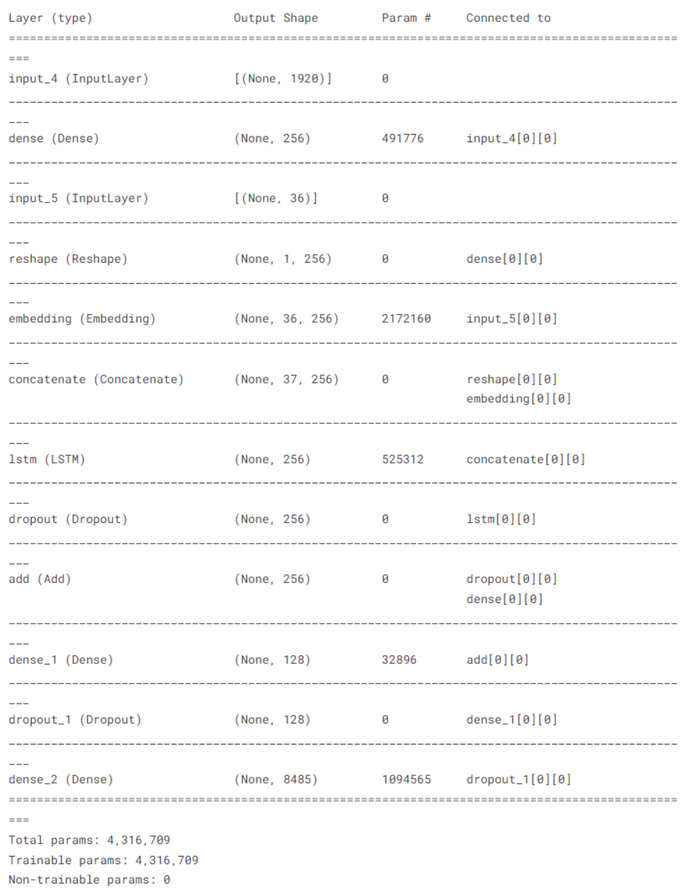


2.2.1 CNN for Feature Extraction

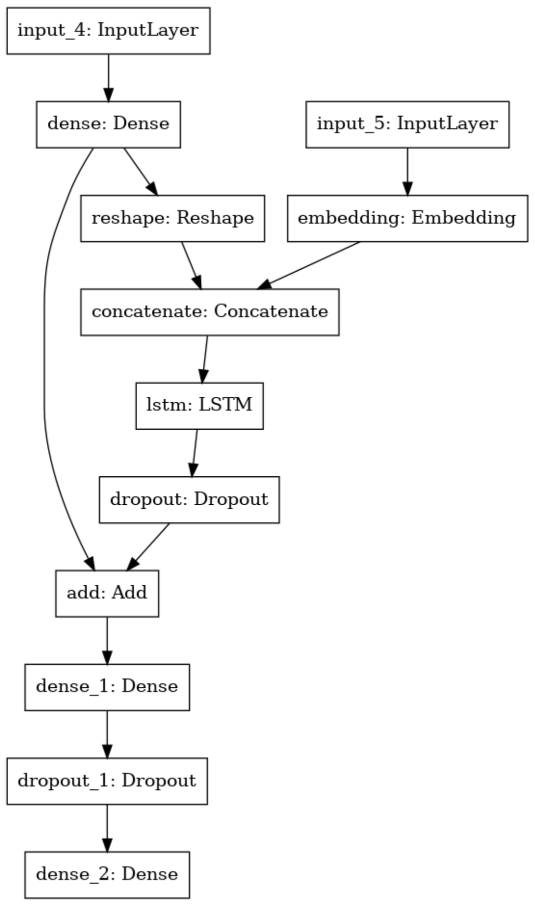
We used the DenseNet201, a pre-trained CNN model, for extracting features from images. DenseNet201 is a deep neural network with 201 layers. It includes dense blocks where each layer receives inputs from all previous layers, promoting efficient feature propagation and reuse. This architecture was advantageous because it mitigated the vanishing gradient problem and enhanced feature propagation. By using a pre-trained model, we saved computational resources and time, as the model had already learned to identify a wide variety of features from a large dataset (ImageNet).

2.2.2 LSTM for Caption Generation

The LSTM network was designed to handle sequential data, such as sentences, making it suitable for generating image captions. LSTMs are a type of Recurrent Neural Network (RNN) that can remember long-term dependencies in sequences, which is essential for generating coherent and contextually relevant sentences.



```
Layer (type)                  Output Shape        Param #    Connected to
==================================================================================================
===
input_4 (InputLayer)          [(None, 1920)]      0
--------------------------------------------------------------------------------------------------
---
dense (Dense)                 (None, 256)         491776     input_4[0][0]
--------------------------------------------------------------------------------------------------
---
input_5 (InputLayer)          [(None, 36)]        0
--------------------------------------------------------------------------------------------------
---
reshape (Reshape)             (None, 1, 256)      0          dense[0][0]
--------------------------------------------------------------------------------------------------
---
embedding (Embedding)         (None, 36, 256)     2172160    input_5[0][0]
--------------------------------------------------------------------------------------------------
---
concatenate (Concatenate)     (None, 37, 256)     0          reshape[0][0]
                                                             embedding[0][0]
--------------------------------------------------------------------------------------------------
---
lstm (LSTM)                   (None, 256)         525312     concatenate[0][0]
--------------------------------------------------------------------------------------------------
---
dropout (Dropout)             (None, 256)         0          lstm[0][0]
--------------------------------------------------------------------------------------------------
---
add (Add)                     (None, 256)         0          dropout[0][0]
                                                             dense[0][0]
--------------------------------------------------------------------------------------------------
---
dense_1 (Dense)               (None, 128)         32896      add[0][0]
--------------------------------------------------------------------------------------------------
---
dropout_1 (Dropout)           (None, 128)         0          dense_1[0][0]
--------------------------------------------------------------------------------------------------
---
dense_2 (Dense)               (None, 8485)        1094565    dropout_1[0][0]
==================================================================================================
===
Total params: 4,316,709
Trainable params: 4,316,709
Non-trainable params: 0
```



Detailed Layer Structure:

- **Embedding Layer**: The captions were first passed through an embedding layer, converting words into dense vectors that captured their semantic relationships.

- **LSTM Layers**: Our model included several LSTM layers. Each LSTM layer had a specific number of units, enabling the model to remember long-term dependencies in the sequences.

- **Dropout Layers**: To prevent overfitting, we incorporated dropout layers. These layers randomly turned off some neurons during training, which helped the model generalize better.

- **Fully Connected (Dense) Layers**: The output from the LSTM was passed through fully connected layers. The final dense layer used a softmax activation function to predict the next word in the sequence.

In summary, this architecture utilized DenseNet201 for image feature extraction and LSTM networks for effective sequence prediction. The use of embedding, dropout, and dense layers ensured our model could generate accurate and meaningful image captions.

2.3 **Model** Training

Training the model involved several crucial steps to ensure it learned effectively and generalized well to new data.

2.3.1 Dataset Split

We began by splitting our dataset into training and validation sets, with an 80:20 ratio. This helped in evaluating the model's performance and ensuring it did not overfit to the training data.

2.3.2 Loss Function

We used categorical cross-entropy as our loss function. This was well-suited for multi-class classification problems like ours, where each word in the caption was treated as a class. The loss function measured the difference between the predicted word probabilities and the actual words in the captions, guiding the optimization process.

2.3.3 Optimizer

The Adam optimizer was used for training. Adam is popular because it combines the advantages of two other extensions of Stochastic Gradient Descent (SGD): AdaGrad and RMSProp. It adjusts the learning rate for each parameter, providing an optimization algorithm that can handle sparse gradients on noisy problems.

2.3.4 Callbacks

To enhance the training process, we used several callbacks:

- **EarlyStopping**: This stopped training when the model's performance on the validation set stopped improving, helping to prevent overfitting.
- **ReduceLROnPlateau**: This reduced the learning rate when a metric had stopped improving, fine-tuning the learning process.

2.3.5 Training Process

We trained our model for 50 epochs with a batch size of 64. The choice of batch size was influenced by hardware and time constraints, balancing the need for efficient computation and effective learning. The batch size of 64 allowed the model to update weights frequently while still maintaining enough data diversity in each batch to learn effectively. The training was paused early on the 13th epoch with no significant improvement past that point.

During training, the model processed batches of images and corresponding captions. Inside each neuron, inputs were transformed through a series of weights and biases, followed by an activation function that introduced non-linearity. The gradients of the loss function with respect to these weights were calculated, and the optimizer adjusted the weights to minimize the loss, iteratively improving the model's performance.

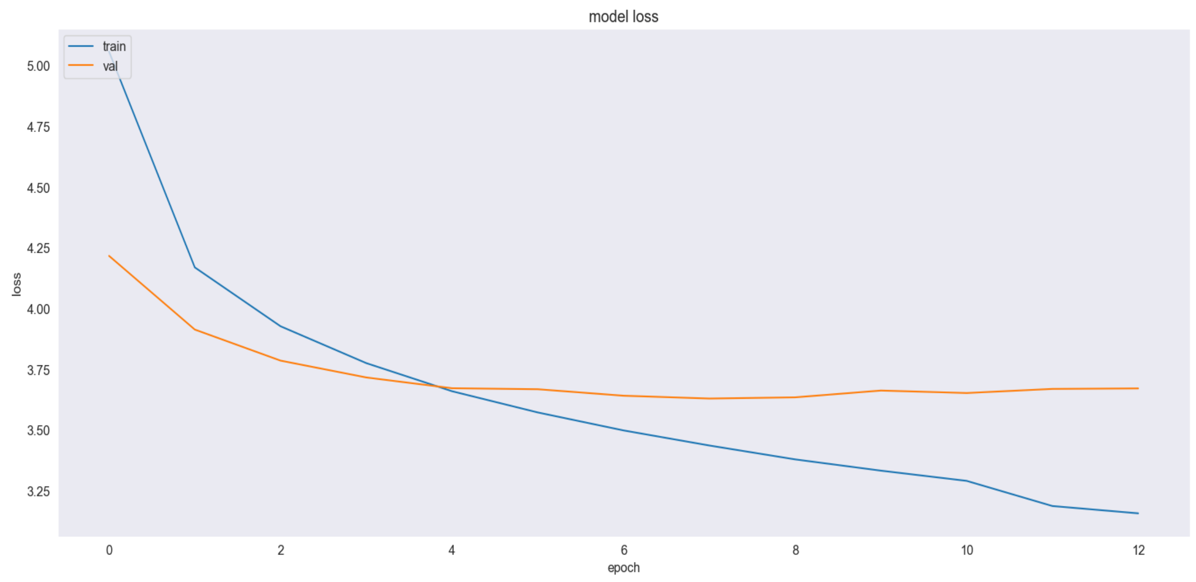
In conclusion, our training methodology, incorporating effective dataset splitting, choice of loss function and optimizer, and strategic use of callbacks, ensured that the model learned efficiently and generalized well to new data, providing accurate and meaningful image captions.

3.0 INFERENCE AND RESULTS

During inference, the trained model generates captions for new images, showcasing its ability to generalize and perform well on unseen data. Initially, a pre-trained Convolutional Neural Network (CNN) extracts a 2048-dimensional feature vector from the input image, encapsulating its visual content and context. These features are then fed into a Long Short-Term Memory (LSTM) network, which begins the caption generation process with a start token (<start>). The LSTM predicts the first word based on the image features and start token, iteratively generating subsequent words by feeding each predicted word back into the LSTM along with the image features, until an end token (<end>) is reached. This iterative process, guided by the Softmax layer's probability outputs, ensures contextual relevance and coherence in the generated captions. This approach demonstrates the model's practical application and potential to effectively describe a wide range of new images in real-world scenarios.

3.1 Learning Curve:

- The learning curve plots the loss over epochs for both the training and validation sets. The training loss steadily decreases, indicating that the model is learning effectively from the training data. However, the validation loss begins to plateau and eventually increases, suggesting potential overfitting. This indicates that while the model performs well on the training data, it struggles to generalize to new, unseen data.

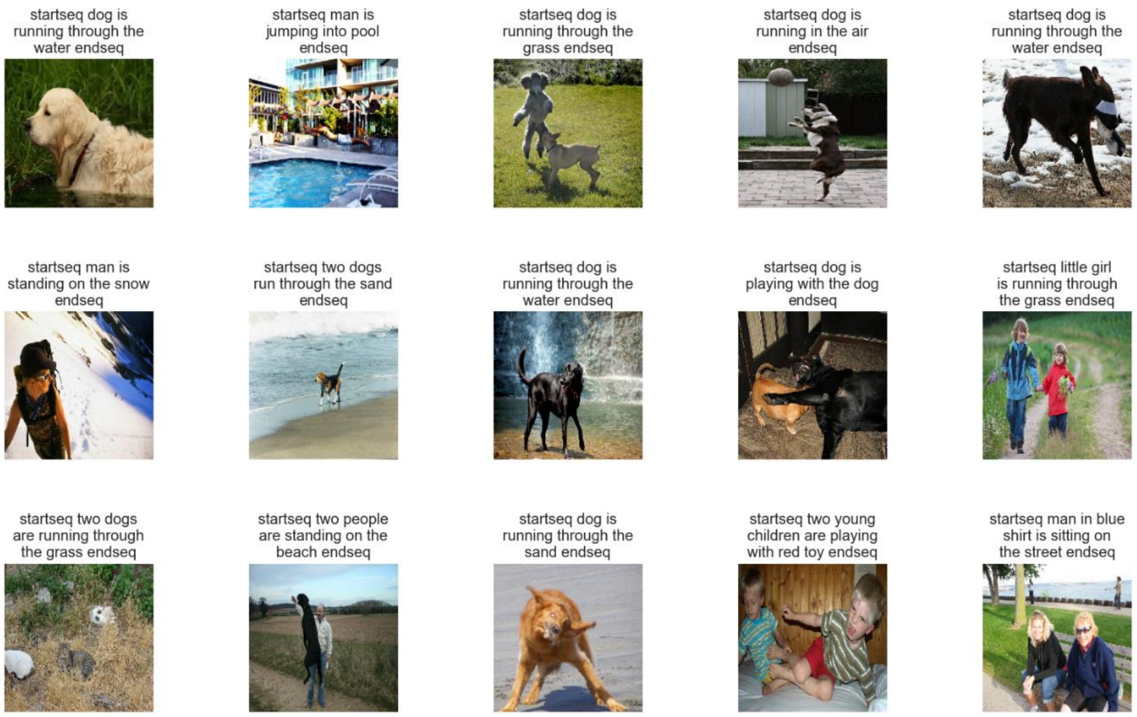


3.2 Model Performance:

The observed overfitting implies that the model may have learned specific details from the training data that do not generalize well to other images. The limited amount of training data is likely a contributing factor. A more extensive and diverse dataset would help the model learn more generalized features, improving its performance on unseen images.

3.3 Assessment of Generated Captions:

Instead of relying solely on quantitative metrics like BLEU scores, we conducted a qualitative assessment by visually examining the relevance of the generated captions to the images. While many captions were accurate and relevant, we noted some discrepancies that highlighted areas where the model struggled. These errors often involved generating less specific or contextually irrelevant words. This qualitative assessment provided valuable insights into the model's strengths and weaknesses, guiding further improvements.


startseq dog is running through the water endseq
startseq man is jumping into pool endseq
startseq dog is running through the grass endseq
startseq dog is running in the air endseq
startseq dog is running through the water endseq
startseq man is standing on the snow endseq
startseq two dogs run through the sand endseq
startseq dog is running through the water endseq
startseq dog is playing with the dog endseq
startseq little girl is running through the grass endseq
startseq two dogs are running through the grass endseq
startseq two people are standing on the beach endseq
startseq dog is running through the sand endseq
startseq two young children are playing with red toy endseq
startseq man in blue shirt is sitting on the street endseq

3.4 Addressing Overfitting:

To address the overfitting issue, we can consider training the model on a larger dataset, such as Flickr40k. More data can help the model learn more robust and generalizable features. Incorporating attention mechanisms is another potential improvement. Attention models allow the LSTM to focus on specific parts of the image while generating each word, leading to more accurate and contextually appropriate captions. Additionally, experimenting with different model architectures or regularization techniques can help mitigate overfitting and enhance model performance.

4.0 CONCLUSIONS

Overall, while our current model shows promise, there are clear paths for further development and enhancement. By leveraging larger datasets and advanced techniques, we can improve the model's ability to generate high-quality, relevant captions for a wide range of images. Continued research and development will enable the creation of more robust and accurate image captioning models, enhancing their practical applications in various fields such as accessibility, content creation, and automated image understanding.

5.0 REFERENCES

Brownlee, J. (2017, October 3). A Gentle Introduction to Long Short-Term Memory Networks by the Experts. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

Chollet, F. (2017). Deep Learning with Python. Manning Publications.

Densenet201. densenet201 - Torchvision main documentation. Retrieved from https://pytorch.org/vision/main/models/generated/torchvision.models.densenet201.html

Karpathy, A. & Fei-Fei, L. (2015). Deep Visual-Semantic Alignments for Generating Image Descriptions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(4), 664-676. Retrieved from https://cs.stanford.edu/people/karpathy/deepimagesent/

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv. Retrieved from https://arxiv.org/abs/1412.6980

RNN - PyTorch 2.3 documentation. Retrieved from https://pytorch.org/docs/stable/generated/torch.nn.RNN.html

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929-1958. Retrieved from http://jmlr.org/papers/v15/srivastava14a.html

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 (CVPR), 3156-3164. Retrieved from https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Vinyals_Show_and_Tell_2015_CVPR_paper.pdf

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., & Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Proceedings of the 32nd International Conference on Machine Learning, 2048-2057. Retrieved from http://proceedings.mlr.press/v37/xuc15.pdf

Saeed G., Image Captioning | Transformer | Flickr30k https://www.kaggle.com/code/saeedghamshadzai/image-captioning-transformers-flickr30k