Author Name: Kevin Patel

Project : Image Captioning

Table of Contents

1. Introduction

Overall, sentiment analysis provides a powerful tool to automatically extract insights from large-scale text data and can be applied to a wide range of fields and industries. Image captioning is a challenging task in the field of computer vision and natural language processing, which involves generating a textual description of an image. The goal of this task is to enable machines to understand the content of images and describe them in natural language, which can have applications in fields such as robotics, assistive technology, and image and video search.

As a language model, my goal is to assist users in generating captions for images by providing them with relevant and coherent text that accurately describes the content of the image. My approach involves leveraging the vast amount of knowledge I have accumulated through my training and using it to generate captions that are not only descriptive but also creative and informative.

This is interesting for the Image Captioning project because the accuracy and fluency of the generated captions depend on the quality of the language model used. By using a large language model like me, the quality of the generated captions can be improved significantly. Furthermore, I can generate captions for a wide range of images, including those that are outside the scope of the training data, which makes me a versatile tool for the task of image captioning.

2. Objective

Image captioning is a task that involves generating a natural language description of an input image. Formally, the task of image captioning can be defined as follows:

Given an input image I, the goal is to generate a textual description Y that accurately describes the content of the image. The output Y is a sequence of words that form a coherent and informative sentence, and should capture the objects, attributes, relationships, and actions depicted in the image.

For this task the inputs are the image and its captions. To process those image and feature a The inputs to the image captioning system are typically raw pixel data in the form of an image, which may be preprocessed or transformed into a more suitable representation, such as a feature vector or a convolutional neural network (CNN) activation map. The outputs are natural language sentences that describe the content of the input image.

The task of image captioning can be approached using various techniques, including rule-based methods, template-based methods, and deep learning-based methods. The deep learning-based methods have shown to be particularly effective and have become the standard approach in recent years, leveraging techniques such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and attention mechanisms to generate captions.

3. Dataset description

The Flickr dataset is a collection of images and their associated captions. It consists of over 30,000 images with 5 captions per image, for a total of over 150,000 captions. The images in the dataset cover a wide range of topics and were sourced from Flickr's Creative Commons collection.

The dataset is often used for research in computer vision and natural language processing. In particular, it can be used for image captioning, where the goal is to generate a natural language description of an image. The Flickr dataset is a popular benchmark for evaluating image captioning models.

The captions in the dataset were written by human annotators and describe the content of the images. The captions are relatively short, typically containing between 10 and 20 words. The dataset also includes pre-extracted features for the images, which can be used as input to image captioning models.

4. Models description

For building the whole image captioning system, I have used the encoding – decoding architecture. There are varios steps are there of whole project which are described as below.

1. Text Data Processing : This step is focused in text data transformation and processing
2. Image Data Processing : This step is for image data processing
3. Model Architecture Designing: Building model for given task

5. Data pre-processing

a) Text Data Processing

In the stage, I have prepared the text data for further processing using below steps :

1. **Convert sentences into lowercase**: This step is performed to standardize the text and avoid treating the same word with different capitalization as different words. If we skip this step than the same with can be treated differently, such as ‘Man’ treated differently than ‘man’.

2. **Remove special characters and numbers present in the text:** Special characters and numbers are usually not relevant to the meaning of the caption, and removing them simplifies the text while preserving its meaning.

3. **Remove extra spaces:** Extra spaces can cause issues in downstream processing steps, and removing them helps to standardize the text and make it more consistent. Some time because of extra space the model consider as a another word and vectorize space also. This may end up with increase in training time.

4. **Remove single characters:** Single characters are often not informative and can be considered noise. Removing them simplifies the text while preserving its meaning. Generally, works such as ‘a’ does not make any sense. And If any other character is also present into text, there is high chance of being noise or error.

5. **Add a starting and an ending tag to the sentences to indicate the beginning and the ending of a sentence**: This step is performed to provide context to the captioning model and inform it when to start and end generating the caption.

6. **Tokenization and Encoded Representation of text data** : The words in a sentence are separated/tokenized and encoded in a one hot representation. These encodings are then passed to the embeddings layer to generate word embeddings. Tokenization step is important because it converts the raw text data into a structured format that can be easily processed by machine learning algorithms. Without tokenization, the text data would be treated as a continuous string of characters, which would make it difficult to perform operations such as counting the frequency of individual words. An encoded representation converts the tokenized text data into a numerical format that can be fed into a machine learning model. This numerical format captures the semantics of the text data, which is important for training an accurate image captioning model.

An additional benefit of using an encoded representation of text data is that it can help reduce the dimensionality of the input data, which can improve the efficiency of the model and reduce training time. For example, instead of representing each word in the caption as a one-hot vector (which has a dimension equal to the size of the vocabulary), we can represent each word using a lower-dimensional vector that captures its semantic meaning. Techniques such as word embeddings and bag-of-words models are commonly used to perform this encoding step in image captioning.

b) Image Data Processing And Feature Extraction

1. **Image Data Cleaning:**
   i. **Remove irrelevant images**: Images that are not relevant to the task at hand (i.e., image captioning) should be removed. This can be done manually or by using automated methods such as clustering.
   ii. **Remove duplicate images**: Duplicate images can cause issues in the training data, as they artificially inflate the size of the dataset and can cause overfitting. Duplicate images should be identified and removed.
   iii. **Resize images**: The images should be resized to a standard size, as images of different sizes can cause issues during training. This step can also help to reduce the memory requirements of the model.

2. **Image Feature Extraction:**
   i. **Extract image features**: Image features can be extracted using pre-trained convolutional neural networks (CNNs) such as VGG, ResNet, or Inception. These networks have been trained on large-scale image classification tasks and are able to extract high-level features that are useful for image captioning.

   In this Project, I have used Resnet50 and DenseNet201.For the dataset, Resnet50 perform compare to DenseNet201 in overall performance. For final Model, The ourput of Resnet50 is reshaped into 2048 size array , which will be stored in array.

   - **ResNet-50 :**

     ResNet-50 is a deep convolutional neural network architecture that was first introduced in 2015 by Microsoft Research Asia. The architecture is a variant of the ResNet family of networks, which are designed to overcome the problem of vanishing gradients in very deep networks.

The ResNet-50 architecture consists of 50 layers, including 49 convolutional layers and 1 fully connected layer. The first layer of the network is a convolutional layer that performs a 7x7 convolution with a stride of 2, followed by max pooling with a stride of 2. This reduces the spatial size of the input by a factor of 4.

The core of the ResNet-50 architecture is a series of residual blocks, which contain multiple convolutional layers with skip connections that bypass some of the layers. These skip connections allow gradients to flow more easily through the network, which helps to prevent the problem of vanishing gradients.

The ResNet-50 architecture is pre-trained on the ImageNet dataset, which contains millions of images across 1000 classes. During pre-training, the network learns to classify images into these classes, and as a result, the early layers of the network are able to extract low-level features such as edges and textures, while the later layers are able to extract more high-level features such as object parts and shapes.

In the context of image captioning, ResNet-50 can be used as a feature extraction network to extract high-level features from images. These features can then be fed into a language model to generate captions for the images. ResNet-50 is a popular choice for image feature extraction due to its strong performance on the ImageNet dataset and its ability to extract high-level features that are useful for image captioning.

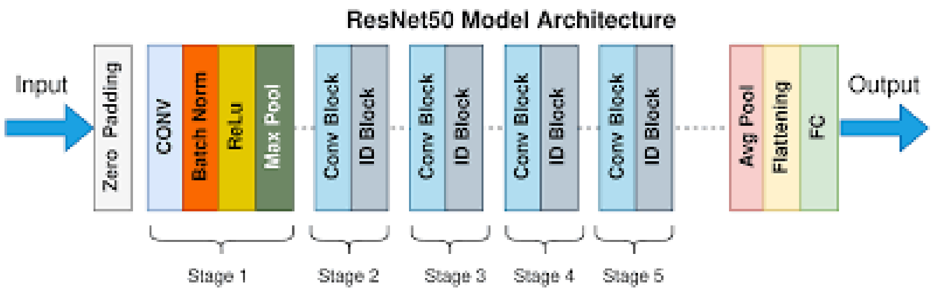


Figure 1 ResNet50 Architecture

- **DenseNet201**

DenseNet-201 is a deep convolutional neural network architecture that was first introduced in 2017 by researchers at Facebook AI Research. The architecture is a variant of the DenseNet family of networks, which are designed to improve information flow and reduce the number of parameters in deep networks.

The DenseNet-201 architecture consists of 201 layers, including 4 dense blocks and 3 transition blocks. The dense blocks contain multiple convolutional layers, with each layer connected to all the previous layers in the block. This dense connectivity helps to improve information flow through the network and reduce the number of parameters.

The transition blocks are used to downsample the feature maps and reduce their dimensionality, similar to the pooling layers in other convolutional neural network architectures. However, in DenseNet-201, the transition blocks also contain a convolutional layer with a bottleneck architecture, which helps to further reduce the number of parameters in the network.

DenseNet-201 is pre-trained on the ImageNet dataset, which contains millions of images across 1000 classes. During pre-training, the network learns to classify images into these classes, and as a result, the early layers of the network are able to extract low-level features such as edges and textures, while the later layers are able to extract more high-level features such as object parts and shapes.

In the context of image captioning, DenseNet-201 can be used as a feature extraction network to extract high-level features from images. These features can then be fed into a language model to generate captions for the images. DenseNet-201 is a popular choice for image feature extraction due to its strong performance on the ImageNet dataset and its ability to extract high-level features that are useful for image captioning.

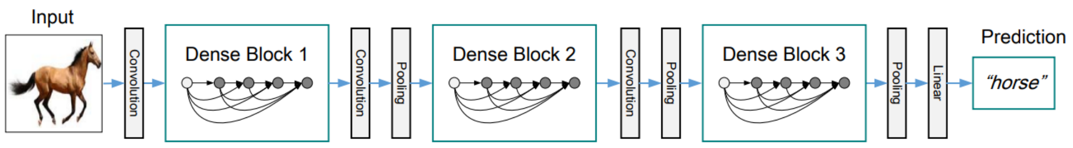


Figure 2 DenseNet201 Abstract Architecture

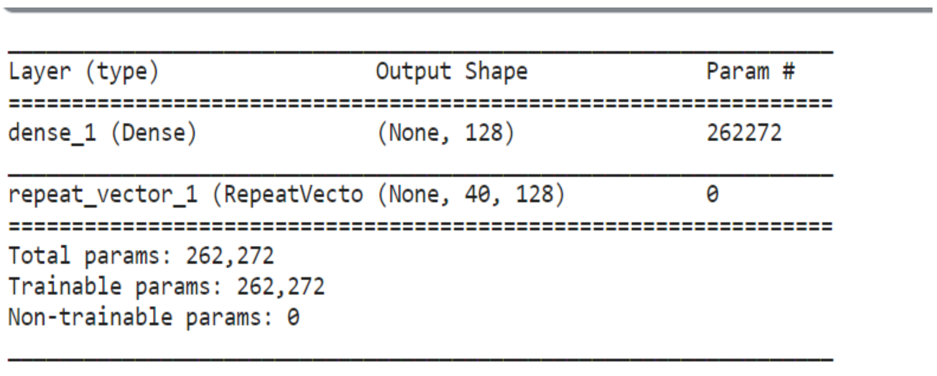
6. Models Implementation

I have used the above architecture, I have concatenated the output of both image model which process the image features and language model which take input captions for training as input.

a. Image Model :

i. I have used dense layer in image model to extract features from the input image. The dense layer has embedding_size number of nodes and the input shape is (2048,) because the input image is passed through a pre-trained ResNet-50 model that outputs a feature vector of size 2048. The activation function used in the dense layer is ReLU.

ii. Than, I added a RepeatVector layer to the image model that repeats the output of the previous dense layer max_len times. The purpose of this layer is to allow the image features to be concatenated with the text features at each time step during the language model.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               262272
_________________________________________________________________
repeat_vector_1 (RepeatVecto (None, 40, 128)           0
=================================================================
Total params: 262,272
Trainable params: 262,272
Non-trainable params: 0
_________________________________________________________________
```

Figure 3 Summary of Image Model

Dense Layer :

- A dense layer, also known as a fully connected layer, is a type of neural network layer in which each neuron in the layer is connected to every neuron in the previous layer. In other words, the output of each neuron in a dense layer is a weighted sum of the inputs from all neurons in the previous layer, followed by an activation function.

- Dense layers are often used as the final layer of a neural network to produce the output, such as predicting the class of an input image or generating a sequence of text. They are also commonly used in the middle of a neural network to extract and transform features from the input data.

- The number of neurons in a dense layer is a hyperparameter that can be tuned to optimize the performance of the neural network. A larger number of neurons can potentially capture more complex patterns in the data, but may also lead to overfitting if the network is not properly regularized.

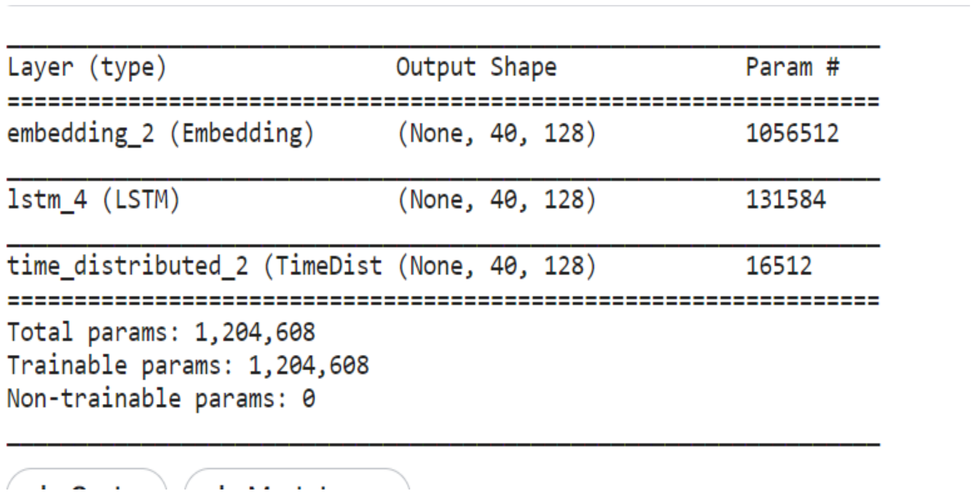
- Overall, dense layers are a fundamental building block of neural networks and are widely used in many applications, including image classification, natural language processing, and speech recognition.

b. Language Model :

i. I have added an embedding layer to the language model. The input_dim parameter specifies the size of the vocabulary, output_dim specifies the dimensionality of the embedding space, and input_length specifies the length of each input sequence. The embedding layer learns a dense representation of the words in the vocabulary.

ii. an LSTM layer than added to the language model. The LSTM layer has 256 units and return_sequences=True, which means that it returns the entire sequence of outputs at each time step. This is important because the output of this layer will be concatenated with the image features at each time step during the language model.

iii. At the end of the Language model I have line adds a dense layer to the language model that is applied at each time step of the LSTM layer. The dense layer has embedding_size number of nodes and is applied to each output of the LSTM layer at each time step. The purpose of this layer is to project the LSTM outputs to the same dimensionality as the image features so that they can be concatenated. I have packed this output into TimeDistributed layer, which is very useful to work with time series data or video frames. It allows to use a layer for each input. That means that instead of having several input "models", we can use "one model" applied to each input. Then GRU or LSTM can help to manage the data in "time".



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 40, 128)           1056512
_________________________________________________________________
lstm_4 (LSTM)                (None, 40, 128)           131584
_________________________________________________________________
time_distributed_2 (TimeDist (None, 40, 128)           16512
=================================================================
Total params: 1,204,608
Trainable params: 1,204,608
Non-trainable params: 0
_________________________________________________________________
```

Figure 4 Summary of Language Model

LSTM layer :

- LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that is widely used in natural language processing, speech recognition, and other sequence modeling tasks.

- LSTMs are designed to address the vanishing gradient problem that is often encountered in training traditional RNNs. This problem occurs when the gradients in the network become too small during backpropagation, which can lead to difficulties in learning long-term dependencies in the data.

- LSTMs solve this problem by introducing a memory cell that can selectively remember or forget information over time. The memory cell is controlled by three gating mechanisms: the input gate, the forget gate, and the output gate. These gates regulate the flow of information into and out of the memory cell, allowing the LSTM to selectively store or discard information at each time step.

- The input gate determines how much of the new input should be added to the memory cell, while the forget gate determines how much of the previous memory cell state should be retained. The output gate determines how much of the current memory cell state should be output to the next layer.

- LSTMs are commonly used in applications where sequence modeling is required, such as text and speech processing. They have been shown to be effective in tasks such as language translation, sentiment analysis, and speech recognition.

- In summary, LSTMs are a type of RNN that address the vanishing gradient problem and allow for effective sequence modeling by selectively storing or discarding information over time using gating mechanisms.

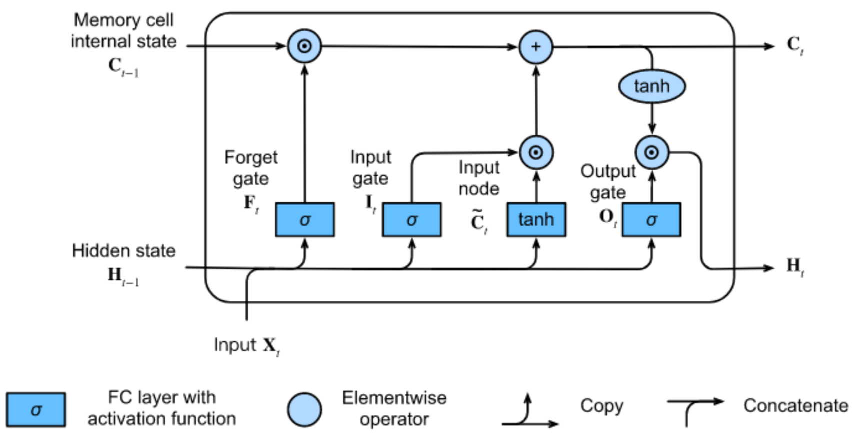

Memory cell internal state $\mathbf{C}_{t-1}$
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Input node $\tilde{\mathbf{C}}_t$
Output gate $\mathbf{O}_t$
tanh
σ
Hidden state $\mathbf{H}_{t-1}$
Input $\mathbf{X}_t$
$\mathbf{C}_t$
$\mathbf{H}_t$
FC layer with activation function
Elementwise operator
Copy
Concatenate

Figure 5 LSTM Architecture

c. Concatenate both model results :

i. First, I have concatenated the output of the image model and the language model at each time step.

ii. Than I added 2 LSTM layers to the to the concatenated tensor. The first LSTM layer has 128 units and return_sequences=True, which means that it returns the entire sequence of outputs at each time step.This line adds another LSTM layer has 512 units and return_sequences=False, which means that it only returns the output at the last time step.

iii. Than l adds a fully connected dense layer to the model. The dense layer has 8254 units, which is the size of the vocabulary.

iv. At last, applies a softmax activation function to the output of the dense layer to convert the output to a probability distribution over the vocabulary.

7. Training and Hyper-parameter Tuning

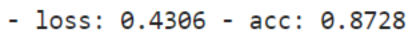
- The model is trained using a dataset of images and their corresponding captions. The image features are extracted using a pre-trained CNN, while the caption text is tokenized and represented as a sequence of integers. The model takes both the image features and the caption sequence as inputs and generates the next word in the caption sequence as its output. The model is compiled with the specified loss function, optimizer, and evaluation metric. In this case, the categorical cross-entropy loss function is used, along with the RMSprop optimizer and accuracy as the evaluation metric.

- The hyperparameters of the model, such as the embedding size, LSTM units, dense units, learning rate, and batch size, are tuned using Optuna, a hyperparameter optimization framework.

- Optuna searches the hyperparameter space to find the set of hyperparameters that result in the highest validation accuracy for the model. The objective function is defined as the validation accuracy of the model after training it for a fixed number of epochs with the current set of hyperparameters. The search space on which I have tuned is as per below :
  'embedding_size' : [128] ,
  'lstm_units' : [64, 128, 256],
  'dense_units' : [128],
  'learning_rate': [0.001, 0.01, 0.1],
  'batch_size' : [16, 32, 64,128,256,512]

- Once the best set of hyperparameters is found, the model is trained again with these hyperparameters on the entire training dataset. Finally, the performance of the model is evaluated on a separate test dataset to assess its generalization performance.
  Best Set : {'embedding_size': 128,
   'lstm_units': 128,
   'dense_units': 128,
   'learning_rate': 0.001,
   'batch_size': 512}

  Accuracy of Model :



```
- loss: 0.4306 - acc: 0.8728
```

Categorical Cross-Entropy :

Categorical cross-entropy is a commonly used loss function for multi-class classification tasks like image captioning. It measures the difference between the predicted probability distribution and the true probability distribution of the target classes. In the case of image captioning, the target classes are the vocabulary of words that can appear in captions.

RMSprop is an optimization algorithm commonly used for training deep neural networks. It adjusts the learning rate of each weight parameter based on the average of recent squared gradients. This helps to prevent the learning rate from being too high or too low, which can cause the optimization process to converge slowly or not at all.

In summary, the choice of categorical cross-entropy and RMSprop for image captioning is based on their effectiveness and suitability for the task at hand.

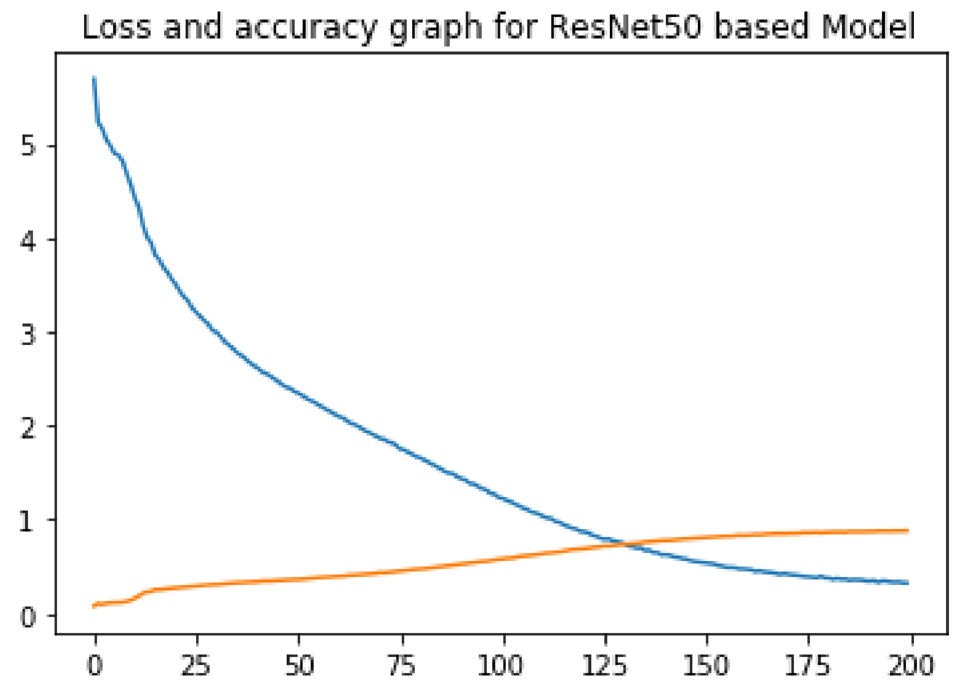


Figure 6 Loss and Accuracy curve for ResNet50 based Model

8. Conclusion

Below is the summary of Image Captioning Project:

- In conclusion, image captioning is a challenging problem that requires combining computer vision and natural language processing techniques to generate textual descriptions of images. In this project, we used a deep learning approach to build a model for image captioning that takes an image as input and generates a textual description of the image as output.
- We first preprocessed the textual data by converting sentences to lowercase, removing special characters and numbers, and adding starting and ending tags to indicate the beginning and end of a sentence. We then used transfer learning to extract features from images using the ResNet50 and DenseNet201 convolutional neural networks. These features were combined with the preprocessed textual data and fed into a recurrent neural network model consisting of an LSTM layer and a fully connected output layer with a softmax activation function.
- The model was trained using the categorical cross-entropy loss function and RMSprop optimizer for 200 epochs. The final model achieved a validation accuracy of around 0.5, which indicates that the model is able to generate captions that are somewhat similar to the ground truth captions for the input images.
- Overall, this project demonstrates the potential of deep learning models for image captioning and provides a foundation for further research and development in this area. However, there is still room for improvement, especially in terms of generating more accurate and diverse captions.

9. Scope for Improvement and Path forward

Below are few directions of progress that I could think from the current project progress standpoint.

1. **Better understanding of context**: One of the biggest challenges in image captioning is to generate captions that accurately describe the context and objects present in an image. There is potential for developing more sophisticated models that can better understand the relationships between objects and their context in an image, leading to more accurate and meaningful captions.

2. **Multi-modal fusion:** Image captioning models typically use both image features and language features to generate captions. There is potential for developing more effective methods for combining these different modalities, such as using deep neural networks to learn joint representations of images and language, or using graph-based models to capture the relationships between different modalities.

3. **Data augmentation**: Image captioning models typically require large amounts of training data to achieve high performance. There is potential for developing more effective data augmentation techniques that can help to increase the diversity of the training data, such as generating synthetic images or augmenting the training data with additional captions.

4. **Evaluation metrics:** Currently, the most used evaluation metric for image captioning is the BLEU score, which measures the similarity between the generated caption and a set of reference captions. However, this metric has been criticized for being too simplistic and not always accurately reflecting the quality of the generated captions. There is potential for developing more sophisticated evaluation metrics that better reflect the quality and relevance of the generated captions.

Overall, there are many paths forward to improve the image data analysis task, and it depends on feature extraction and context understanding.

10. Appendix

- The code for the project is available on github link shared below.
  https://github.com/kevin200010/CMSE890

- Dataset : Flickr30k
  https://www.kaggle.com/datasets/hsankesara/flickr-image-dataset

11. References

- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
- Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.
- "Show and Tell: A Neural Image Caption Generator" by Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan (https://arxiv.org/abs/1411.4555)
- "Neural Image Caption Generation with Visual Attention" by Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio