Image Captioning using Vision Transformers

Pradnya Mundargi
University of Maryland
pradnya@umd.edu

Aditi Ramadwar
University of Maryland
adiram@umd.edu

Shelvin Pauly
University of Maryland
spauly@umd.edu

Abstract

Image captioning is a challenging task in the field of computer vision, where the goal is to automatically generate a natural language description of an image. This task can be thought of as converting a sequence of pixels into a sequence of words. It utilizes both Computer Vision(CV) as well as Natural Language Processing(NLP) expertise. The Vision Transformer (ViT) model has been shown to be effective for this task by using self-attention mechanisms to capture the global dependencies among image patches, without the need for spatial convolutions. This allows the model to scale to larger image sizes and achieve better performance on the image captioning task. By using the ViT model, it is possible to generate more detailed and accurate descriptions of images, which can be useful for a wide range of applications. For our final project we propose to develop a pipeline which requires an image as an input and generates the appropriate caption for it as the output using a Vision Transformer. Our model was trained as well as tested on the MS Coco dataset as well as Flicker. It was observed that the Vision transformer performed better than the baseline model on MSCOCO dataset. We have also attempted to use our model for a novel application discussed in section 9.

1. Introduction

Image captioning is a challenging problem in the field of computer vision, where the goal is to automatically generate a natural language description of an image. Deep learning models have been widely used for this task, as they are able to learn complex patterns from large amounts of data and generate high-quality image captions. There are several different types of deep learning models that have been applied to image captioning, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs). CNNs are able to extract high-level features from the raw image pixels, while RNNs are able to capture the sequential nature of natural language and generate captions one word at a time[1].

For a long period of time, a combination of CNNs and LSTM was considered state of the art for the longest time [2]. However, the limitations of LSTM training, efficiency and expression ability limit the impact of relevant models. Following the popularity of the Multi-head SelfAttention (MSA) mechanism and Transformer architecture in NLP [3], many researchers began to incorporate MSA into LSTM decoder[4] or adopt Transformer architecture as decoder[5]. Recent advances in deep learning have also led to the development of transformer-based models, which use self-attention mechanisms to capture global dependencies among image features. These models have shown promising results on the image captioning task and offer a promising direction for future research. However, currently most neural networks use a CNN as a feature extractor and just the decoder part of a transformer for the purpose of image captioning. This results in generation of generalized captions which may not be relevant to the input image.

Realizing this shortcoming, our model architecture uses a CNN, a vision transform encoder as well as the decoder. Using an encoder we positionally encode not only the words but also the image features extracted by the CNN. We believe this technique will promote the creation of more relevant captions for a given image.

We could observe that our approach outperformed the baseline model by adding more details of the image in the caption. The loss during training was much lower than that of the baseline model. We dive into the results even further in the coming sections.

2. Related Work

Image captioning algorithms can be divided primarily into two types: encoder-decoder based algorithms and the template method based algorithms. Initial research in the field on image captioning heavily relied on the template method. This method mainly consisted of extracting key feature data such as any significant objects and special attributes through different types of classifiers. Following this specific templates were used to convert the obtained feature information into description text. The authors in [6] extract key feature information that primarily includes the object in the image, the action of the object, and the scene in which the object is located. Finally, the extracted information's relationship is examined, and the final image description result is obtained. However, this method is not robust and cannot handle versatile inputs.

The image captioning model based on the encoder decoder structure outperformed the template method models. The encoder in this case was primarily a convolution neural network that extracted and vectorized features from the input image. The decoder, which could be a recurrent neural network, combined vectorized image features with semantic information to create a descriptive image caption. The introduction of techniques such as attention mechanisms allowed researchers [7] [8] to produce more relevant captions. The authors of [9] proposed two attention mechanisms, namely soft attention and hard attention. The former mechanism concentrates on all areas within the image, with each area having a different weight value. With a greater value, more

significant the region is for prediction. The latter mechanism is usually used to increase flexibility because it only focuses on one area which is randomly selected and its probability value is calculated. Recently, with the introduction of Vision Transformers [10], the image captioning research made a jump from using CNN + RNN networks to Vision Transformers to produce far better results[11]. For our model we have used a pre-trained BERT tokenizer[12]. Our approach is similar to the work proposed in DETR[19], this paper proposes the use of ViT for object detection by predicting bounding boxes of the object in the image. We derive our architecture from this paper and create modifications to serve the purpose of generating image captions.

3. Dataset

The Microsoft Common Objects in Context (MS COCO) caption dataset is a large-scale dataset for image captioning, which means that it includes a large number of images and their corresponding captions. The dataset was created to help train and evaluate algorithms for automatically generating descriptions of images. It contains more than 120,000 images and over 5 million caption annotations, making it one of the largest image captioning datasets available. The captions in the dataset are written in natural language and provide detailed, human-like descriptions of the images. This makes the dataset particularly useful for training and evaluating algorithms that aim to generate human-like image captions.

We conducted an Exploratory data analysis (EDA) for MS COCO 2017 dataset.

Training dataset: 118,287 images
Validation dataset: 5,000 images
Number of captions: 123,287
There are no standard dimensions of images in the MSCOCO dataset.

Parts of images can affect the way the model looks at the image to generate captions. So, we checked for the number of objects in each image of the dataset.

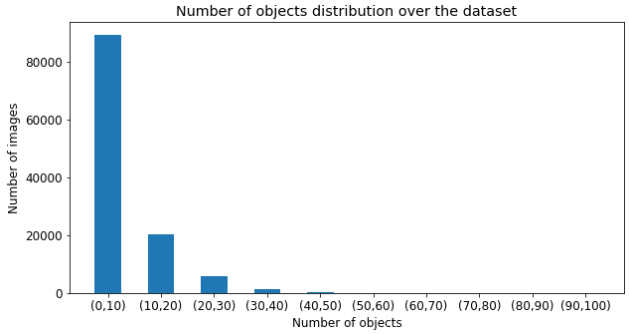

Figure 3.1: Statistics of objects per image

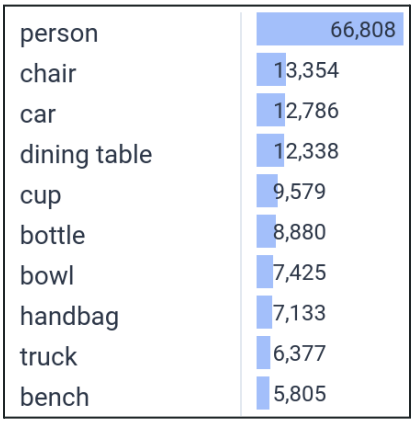

Figure 3.2 : The highest occurring objects in each caption[18]

The dataset consists of majorly people in it. There are other objects as well in the images such as bicycles, cars, motorcycles ,etc. Objects like vase, scissors, teddy bear, hair drier, toothbrush have the lowest distribution in the images of the dataset.

Image ID	Label Group
person	1
car	3
motorcycle	4
boat	9
bench	15
handbag	31
kite	38
cup	47
banana	52
chair	62

Figure 3.3: Table of Image IDs and Label Groups

Figure 3.4: A random image with its associated caption

Words associated with the random image generated from the dataset: person, bowl, carrot, sink, cat and knife.

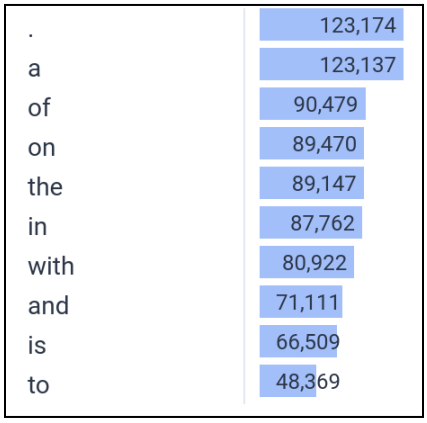

Figure 3.5 : The highest occurring words in each caption[18]

4. Our Approach

The overall architecture of our model is shown in Figure 4 which is inspired from the model mentioned in [19].

We utilize the common encoder-decoder framework, where the encoder is made up of a ResNet backbone and stacks of N refining encoder blocks, and the decoder is made up of stacks of N decoder blocks. The encoder's job is to refine the grid features by recording their intra-relationships as they are extracted from the input image. By identifying the relationships between word and image grid characteristics, the decoder creates the captions word-by-word using the revised image grid features.

Bert Tokenizer

A BERT tokenizer is a type of tokenizer that is specifically designed to be used with BERT, a popular natural language processing model. A BERT tokenizer splits text into tokens, which are the basic units that BERT uses to process language. In BERT, each word in the text is typically represented by a single token, but some words may be represented by multiple tokens if they are part of a multi-word token or if they have been split into subwords. BERT tokenizers are typically trained on a large dataset of text, which helps them to learn the rules for tokenization and to produce tokens that are well-suited for use with BERT. They are often used in natural language processing tasks to preprocess text data before it is fed into a BERT model.

Patch and Positional Embeddings

Positional embeddings are a way of incorporating information about the position of an input within a sequence of data into a neural network. This is particularly useful in natural language processing tasks, where the order of words in a sentence can provide important information about the meaning of the sentence. Positional embeddings are typically added to the input data at the beginning of the neural network, and are used to help the network learn the relationship between the position of words in the input and their meaning. This can help the network to better understand the context of the input and improve its performance on natural language processing tasks.

Patch embeddings are a way of representing visual information in the vision transformer, a type of neural network architecture for image recognition. In the vision transformer, an image is divided into a grid of patches, and each patch is represented by a fixed-length vector, known as a patch embedding. These patch embeddings are then used as input to the transformer, which uses self-attention mechanisms to learn the relationships between the patches and generate a global representation of the image. This global representation is then used to make predictions about the content of the image. The use of patch embeddings allows the vision transformer to process images in a hierarchical manner, which can improve its performance on challenging image recognition tasks.

Multi Headed Self Attention Mechanism

Self-attention is a mechanism used in certain types of artificial neural networks, specifically in the context of natural language processing (NLP) tasks. It allows the network to automatically learn relationships between different words or elements of the input, rather than relying on pre-specified rules or manually-engineered features.

In a self-attention mechanism, the output of each element in the input sequence is calculated by taking a weighted sum of all the other elements in the sequence. This weighting is based on the similarity between the elements, which is determined by a compatibility function that compares the elements. This allows the network to focus on certain elements of the input while ignoring others, depending on their relevance to the task at hand.

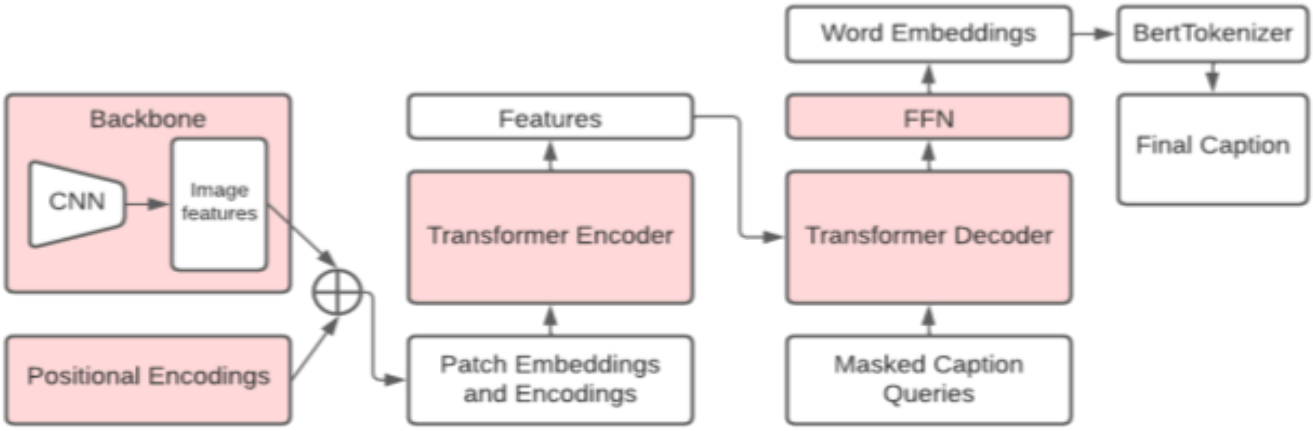

Figure 4: Overall Model Architecture

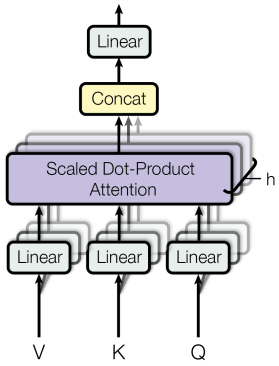

Figure:4.1.1 Multi-Head Attention consists of several attention layers running in parallel [3]

In a transformer, the self-attention mechanism is augmented with the use of multiple "heads". Each head is a separate self-attention mechanism, and the output of the self-attention layer is the concatenation of the outputs of all the heads. Using multiple heads allows the transformer to attend to different parts of the input sequence simultaneously, using different combinations of elements to calculate the output for each head. This allows the model to capture different types of relationships between the elements in the input sequence, and can improve the performance of the model on certain tasks.

The use of multiple heads also has the benefit of making the transformer more parallelizable, which can improve the speed and efficiency of model training.

4.1 Model Architecture

A CNN backbone to extract a compact feature representation, an encoder-decoder transformer, and a basic feed forward network (FFN) that determines the final detection prediction are its three main parts, which we will go over in more detail below.

Backbone:

A traditional CNN backbone creates a lower-resolution activation map from the initial image (which has three color channels).

ResNet is a type of convolutional neural network (CNN) that is specifically designed to be able to identify features in images and classify them. It does this by using a series of convolutional layers, which are layers of neural networks that are designed to process data from images. These convolutional layers are interleaved with other layers known as "residual" layers, which allow the network to more easily learn and identify complex features in the images. This makes ResNet particularly effective at tasks like image classification and object detection.

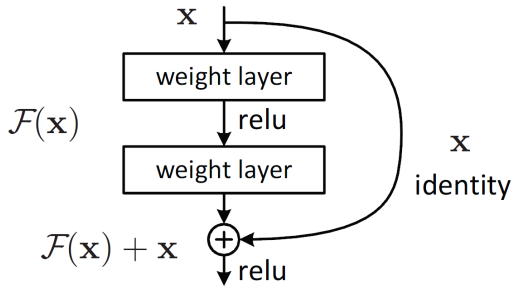

Figure 4.1.1: Residual Block[16]

To get a 2D representation of the input image that will be useful for the transformer, features will be extracted from the image by using a backbone. The backbone chosen is a pre-trained feature extractor ResNet34 [16].

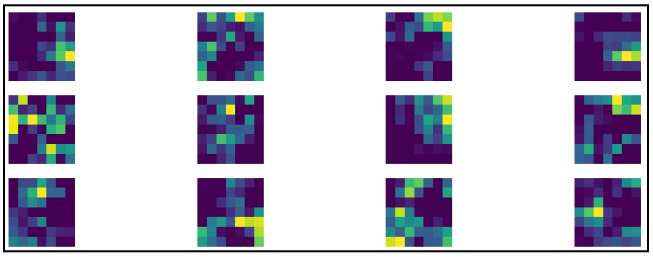

Figure 4.1.2: Reshaped features from ResNet34

The dimension of features coming out of the backbone is 512x1.

Transformer Encoder:

A 1x1 convolution first creates a new feature map by condensing the high-level activation map's channel dimension to a smaller dimension. We reduce the spatial dimensions of the new feature map to one dimension because the encoder requires a sequence as input. A feed forward network and a multi-head self-attention module make up the conventional architecture of each encoder layer (FFN). The fixed positional encodings are an addition to the permutation-invariant transformer architecture and are added to the input of each attention layer [13,14].

The transformer encoder consists of a series of self-attention layers, which allow the model to attend to different parts of the input sequence and weigh their importance in relation to the task at hand. This allows the model to capture complex relationships between the words in the input sequence and use them to make predictions or generate text.

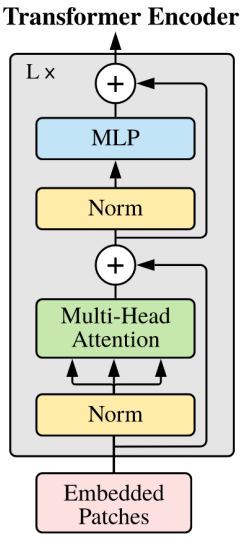

Figure 4.1.3: Transformer Encoder[17]

Transformer Decoder:

The decoder uses multi-headed self- and encoder-decoder attention techniques to convert the embeddings in accordance with the transformer's standard architecture. Vaswani et al. [15] utilizes an autoregressive model that predicts the output sequence one element at a time, in contrast to our model, which decodes the output sequence concurrently at each decoder layer. Since the decoder is also permutation-invariant, different input embeddings will result in various outputs. These input embeddings, which we refer to as caption queries, are learned positional encodings that we add to the input of each attention layer in a manner similar to how the encoder does it.

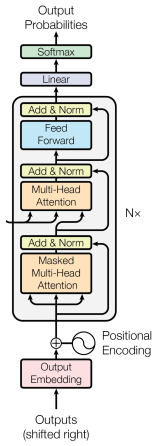

Figure 4.1.4: Transformer Decoder

The output of the decoder will be a list of token embeddings that represent the predicted caption. These tokens will be decoded to extract captions in its word format.

Feed Forward Neural Networks (FFNs):

A three-layer perceptron with a hidden dimension and ReLU activation function, as well as a layer of linear projection, compute the final prediction. The FFN makes predictions about the caption's word embeddings based on the input image. These embeddings can then be decoded using the BertTokenizers to extract a sentence as the caption.

5. Experiments and Training

As mentioned above, we initialize the CNN part of the network with pretrained ResNet34 weights. We then set up the work tokenization using the pre-trained bert tokenizer and set the vocabulary size to 30,522. The number of encoder layers is set to 6 layers, similar to the number of decoder layers. The number of attention layers in each multi-head attention block is set to 8. The number of parameters of this model is 37,317,562.

The network was written and trained in the Pytorch framework on a RTX 3060 GPU with 6Gb of memory with a batch size of 32. The loss function used was the cross entropy loss.

The optimizer used was the Adam Optimizer with weight decay of 0.0001, which is a regularization technique that helps to prevent overfitting by adding a penalty term to the loss function. This penalty term encourages the network to use smaller weights, which can improve the generalization ability of the network and help it to perform better on unseen data. When using Adam with weight decay, the optimizer updates the weights of the network based on both the gradients of the loss function and the weight decay penalty, which can improve the performance of the network on a wide range of tasks. The network was trained for 10 epochs with a learning rate of 0.0001 with a learning drop of 20. Multiple experiments were conducted during training to get the hyper

parameters right

The current model took about 1 hour to train per epoch given the limitation in compute and memory. Multiple combinations of the model architecture were made in order to generate correct captions.

Data augmentation and pre-processing of the input image during training is done by applying random rotation, horizontal flips, normalization, color jitter for varying brightness, contrast and saturation.

Training loss was observed to be as shown in figure below.

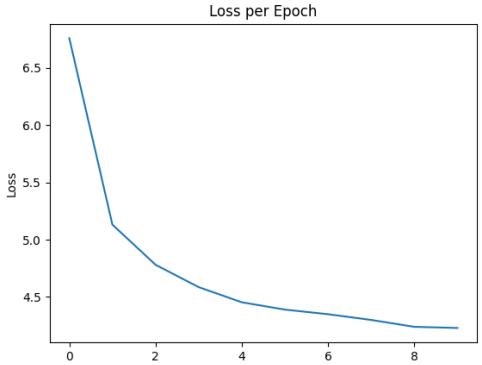

Figure 5.1 : Loss per epoch our model

We also compare our model results with the baseline results.[20] We implemented the building blocks of [20] to create our baseline model. Our baseline model is derived from the classical transformer where the encoder is a CNN and the decoder is a classical transformer decoder. The architecture of the baseline model is shown in Figure 5.2.

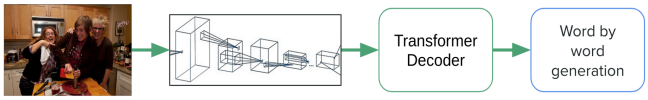

Figure 5.2 : Baseline Model Architecture

This architecture is currently widely used for image captioning which gives decent results hence we chose this model as the baseline model to improve upon the encoder to provide better representation of the input to the decoder.

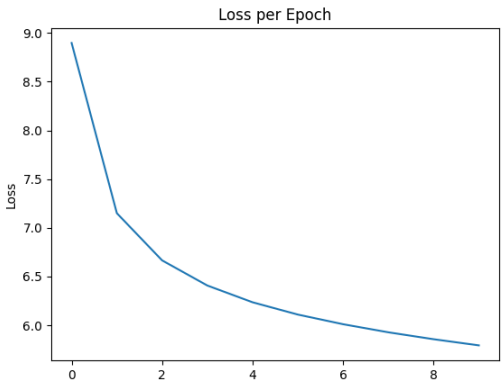

Figure:5.3 Loss per epoch of baseline model

By comparing the loss per epoch during training between our model and the baseline model, we can observe that there is a drastic decrease in the loss in our model as compared to the baseline model. This can be due to the lack of attention and tending to the input sequence.

Since the model is trained on the MSCOCO dataset, we performed an experiment to test the model on images outside of the training domain. Hence we extracted images from the Flickr dataset in order to carry out this experiment.

6 Results

6.1 Comparison between Baseline model and our model

Ground Truth: a bicycle is mounted on the back of a motorcycle
Baseline Output: a bike is sitting on the side of the street
Our model Output: a man standing next to a motorcycle in a parking lot

Ground Truth: a park and walkway lined with benches and bushes
Baseline Output: a clock is on a <UNK> in a park
Our model Output: a park bench sitting next to a tree in a park

The comparison between the baseline model and our model can be seen where due to a better understanding of the input image through self attention and positional embedding, the model is able to focus on certain features of the image and come up with a coherent caption which explains the image accurately. It can be seen that due to the absence of attention, the baseline model is unable to recognize a few features and is labeling them as unknown with the <UNK> token.

6.2 Test on Flickr Dataset

Flickr dataset consisting of 8,000 photographs with five distinct captions for each image that clearly describe the important people, places, and things.

A group of young men playing soccer on a field.

A group of people standing around a horse.

A man riding a skateboard down a street.

It can be seen that after training the model for 10 epochs on the COCO dataset, if the model is tested on a different dataset, it still does fairly well. Although the caption could be generated closer to the ground truth by training the model for more epochs, provided higher compute and memory is available.

7. Limitation

Currently, our model is extremely computationally heavy owing to the large network. Furthermore, the model trained on the MSCOCO dataset does not perform particularly well when tested on the Flicker dataset. We attribute this to our model being trained only for 10 epochs due to the limited resources. To cope with this limitation, the training dataset can also be diversified by combining datasets together.

8. Conclusion

When tested on the challenging COCO dataset, our model achieves better results to our baseline classical transformer. Furthermore, owing to the inclusion of the vision transformer encoder, the performance is significantly better as the model generates more accurate and detailed captions. Our design presents its own set of challenges, the most notable of which is its computational cost. A future path of improvement could be with respect to our novel application as discussed in the next section. At this moment, our application provides a similarity score which indicates a degree of correctness of the user's input. Moving forward we would like our application to provide suggestions and directions of improvement.

9. Going above and beyond

We decided to use our trained model for a novel application aimed at people for whom English is a second language. Our novel application would show the user an image and would request the user to describe what they see in one simple sentence. The baseline reference for that image would be generated by our model when the same image is passed through it at inference time. The similarity between the user input and generated baseline is calculated using Bilingual Evaluation Understudy (BLEU). BLEU is a metric used to evaluate the quality of a machine-generated translation. It compares the machine-generated translation to a set of reference translations and calculates a score based on how closely the machine-generated translation matches the reference translations. The score is a number between 0 and 1, with a higher score indicating a better translation. The BLEU score is commonly used in natural language processing (NLP) research and has been found to be a reliable metric for evaluating machine translation quality.

Since simply displaying the similarity score would not make much sense to an average user, we set range brackets in order to provide feedback. A score below 0.3 is considered as a bad score and the user would be asked if they would like another attempt. A score between 0.3 to 0.7 is considered a good attempt with room for improvement and in this case as well the user will be provided with an option to try again. A score above 0.7 is considered as a fantastic match and the user will be provided with the positive feedback. The demo video of the application can be found here[].

Image displayed for the user

```
Please describe what you see in one simple sentence: People in a room
--------
Similarity score: 0.043443485862611285
--------
Not the best attempt. Do you want to try again?
----------
Baseline sentence: A group of people standing around a table with a cake
```

User's attempt 1

```
Please describe what you see in one simple sentence: A group of people standing together
--------
Similarity score: 0.3621651737558986
--------
Good attempt. Do you want to try again?
----------
Baseline sentence: A group of people standing around a table with a cake
```

User's attempt 2

```
Please describe what you see in one simple sentence: A group of people standing around the table with food
--------
Similarity score: 0.7238699344287677
--------
Fantastic!
----------
Baseline sentence: A group of people standing around a table with a cake
```

User's attempt 3

References

[1] Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015. Show and tell: A neural image caption generator. In Proceedings of the CVPR, 3156–3164.

[2] Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. Neural Computation, 9(8): 1735–1780.

[3] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In Proceedings of the NIPS, 5998– 6008

[4] Huang, L.; Wang, W.; Chen, J.; and Wei, X. 2019. Attention on Attention for Image Captioning. In Proceedings of the ICCV, 4633–4642.

[5] Cornia, M.; Stefanini, M.; Baraldi, L.; and Cucchiara, R. 2020. Meshed-Memory Transformer for Image Captioning. In Proceedings of the CVPR, 10575–10584.

[6] 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. In: European conference on computer vision, Berlin, pp. 15-29

[7] Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Liu, W., & Chua, T. S. (2017). Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5659- 5667

[8] You, Q., Jin, H., Wang, Z., Fang, C., & Luo, J. (2016). Image captioning with semantic attention. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4651-4659.

[9] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In: International conference on machine learning, pp. 2048-2057.

[10] Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

[11] He, S., Liao, W., Tavakoli, H.R., Yang, M., Rosenhahn, B. and Pugeault, N., 2020. Image captioning through image transformer. In Proceedings of the Asian Conference on Computer Vision.

[12] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[13]. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.:Image transformer. In: ICML (2018)
[14] Bello, I., Zoph, B., Vaswani, A., Shlens, J., Le, Q.V.: Attention augmented convolutional networks. In: ICCV (2019)

[15] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)

[16] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[17] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. and Uszkoreit, J., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[18] EDA for MSCOCO

[19] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. and Zagoruyko, S., 2020, August. End-to-end object detection with transformers. In *European conference on computer vision* (pp. 213-229). Springer, Cham.

[20] Baseline model code

[21] Demo code for novel application for language learning. Demo Video

Contribution

Aditi:
1. Our model implementation:
Implemented the new model using the DETR paper by changing the input and output as well as the task. Developed code for masking the captions and target in the transformer architecture to replace the object detection algorithm used in the paper.

2. Model training experiments:
Due to large number of model parameters, carried out multiple experiments to change the batch size, learning rate, weight decay and number of epochs

3. Architecture experiments:
Multiple resenet architectures were used starting from resnet101 to resnet32. The number of multi-headed attention layers were varied to extract the best results. The number of encoder and decoder layers were varied as well.

Pradnya:
1. Baseline model implementation
Implemented Assignment 3: Image captioning using transformers to obtain results for the baseline model
2. Baseline training experiments
Tuned the model hyperparameters to create a model with the best possible output.
3. Created interface for novel application
Wrote a script to display an image, get the caption generated when that image runs through the inference for our trained model and prompt the user for an input. A similarity score was calculated between both the sentences and this score runs through a interpretability function to provide the user a feedback

Shelvin:
1. Literature review
Read related survey papers for literature review and trained various models as experiments

2. EDA for MSCOCO
Exploratory data analysis MSCOCO dataset

3. Generated results using COCO and flickr