Image Captioning via Vision and Language Transformers

Luke Davidson - davidson.lu@northeastern.edu

Kishore Pagidi - pagidi.k@northeastern.edu

Nand Dave - dave.na@northeastern.edu

Abstract

With today's rise in artificial intelligence applications, the relationship between computer vision and natural language processes has been an area of great interest. Image captioning is a task that combines aspects of computer vision and natural language processing to automatically generate descriptive captions of input images. Image captioning techniques are used in a wide range of applications, including visual media, virtual assistants, and accessibility domains. In this project, we developed an end-to-end image captioning model using vision and language transformers capable of generating descriptive captions for a range of input images. We utilized transformer architecture-based computer vision and natural language processing methods to build and train a model capable of generating accurate captions in unobserved images. This model was constructed by implementing components such as image patch descriptors and encoders, decoders, multi-head attention layers, and transformers to decompose and find the relationship between images and training captions. We conducted experiments to evaluate the impact of various hyperparameters on the model's performance and employed a custom learning rate scheduler to adapt the learning rate during the training process. The results indicate that increasing the batch size, as well as the number of heads and layers, can lead to improved performance in terms of accuracy.

Introduction

The relationship between computer vision and natural language processing applications, two major areas of deep learning, is one that is greatly increasing

in popularity, research and implementation today due to its many potential use cases and benefits. Image captioning methods are a perfect combination of these two areas of deep learning, combining object detection and classification methods directly with sentence formation methods to create descriptive captions. As previously stated, one of the primary reasons for the increase in popularity of image captioning models is its many potential use cases and benefits. Automated image captioning methods are used in industries such as marketing, editing, and assistance services to complete tasks such as the automated captioning of social media posts, recommendation systems for image and video editing, and assisting visually impaired individuals with describing a visual of a live view or still image [1]. These use cases increase aspects of an individual's life, such as efficiency in their occupation or the general quality and simplicity of their everyday life. Image captioning methods and models have proven to be extremely efficient in these domains and are only becoming more accurate and robust [1].

In this study, we developed a transformer-based model to identify objects and corresponding object locations in images and match those objects and locations with descriptive captions. Transformer-based deep learning models have been widely used to solve natural language processing problems but have recently been more implemented in computer vision domains as well. A great strength of transformer-based deep learning models is their ability to encode feature location descriptions with detected features [1], whether that be the location of a word in a sentence or the relative location of an object in an image. This is an extremely important aspect when considering which model architecture to use to solve an image captioning problem, as overall scene understanding is a crucial aspect to generating an accurate, descriptive caption of an image. We chose a transformer-based model architecture primarily because of this benefit, implementing methods such as image patch extractors and descriptors, feature encoders and decoders, and multi-head attention layers, further explained below in the *Approach* section.

We used the Common Objects in Context (COCO) dataset [2] to train our transformer-based image captioning model. The COCO dataset was created for applications such as scene understanding and segmentation and aims to solve three core research problems seen in common image datasets: detecting non-iconic views (or non-canonical perspectives) of objects, contextual reasoning between objects, and the precise 2D localization of objects [3]. The dataset contains roughly 328,000 images of over 90 objects in their natural environment. Most importantly, the dataset also contains roughly 4-6 captions for each image, with over 2.5 million total captions describing key objects, object localizations and scene environments. Two examples of images and corresponding captions from the COCO dataset can be seen below in Figures 1 and 2.

the young boys are practicing tricks on their skate boards.
a boy doing a trick on a skateboard on a rail.
this is a skateboarder doing a dangerous trick.
a boy on a skateboard going down a handrail at a set of stairs outside.
a boy skateboards down a hand rail while two others watch.

Figure 1: Example image and captions from COCO dataset

a man in the park holding onto a green frisbee and the leash for a dog
a person in a ball cap and holding a frisbee with a dog.
the man is holding the leash of a black and white dog.
a person in a field with a frisbee and a dog
a man walking a dog while holding a frisbee.

Figure 2: Example image and captions from COCO dataset

Background

As previously stated, there has been a large increase in the popularity of image captioning and image captioning like models that relate computer vision and natural language processing applications in recent years due to today's advances in artificial intelligence. Transformer based architectures have been the most popular deep learning architecture to solve problems like this due to their ability to encode feature descriptors and efficiently work with large amounts of data.

The popular paper *"An Image is Worth 16x16 Words: Transformers for Image Recognition At Scale"* [4] displays these strengths through comparing an implementation of a vision transformer with typical high performance neural networks such as ResNet and EfficientNet. The vision transformer, which uses 16x16 pixel subsections of images to pass to the transformer architecture, outperforms the neural networks when pre-trained with at least 100 million training images in just a fraction of the computational power, while maintaining the key benefit of being able to encode feature locations due to the use of the pixel patch extractors. Another extremely important aspect that makes a transformer-based architecture efficient in computer vision and natural language processing domains is the use of attention. An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key [4]. The paper *"Attention is All You Need"* [5] highlights the benefits of attention by proposing a new simple network architecture based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Yiyu Wang, Jungang Xi, and Yingfei Sun also introduce an alternative to the popular R-CNN backbone architecture used in many image captioning models in their paper *"End-to-End Transformer Based*

Model for Image Captioning" [6]. Their approach replaces the R-CNN backbone encoder with a SwinTransformer to extract grid-level features. This method includes a refining encoder and mean pooling of grid features and addresses the limitations of pre-trained CNNs, object detectors in the encoder and LSTM in the decoder, using a Swin-Transformer encoder and MSA to determine the intra-relationship of image grid features. We have built off of these previous studies to implement our own variation of an end-to-end vision and caption transformer, further explained in the following *Approach* section.

Approach

As previously stated, we chose to develop a transformer-based network architecture to create our image captioning model. Our implementation comprises four main components: a data pipeline, a vision transformer, a combined encoder and a combined decoder.

The data pipeline is a crucial first step to ensure the images and captions downloaded from the COCO dataset are in the correct format to optimize our model. Our data pipeline consists of four main steps: downloading the raw data, grouping captions to their corresponding images, tokenizing the captions, and transforming the images. The raw dataset of ~328,000 images and ~2.5 million captions is downloaded from the web. Each image is then paired with its corresponding list of ~4-6 captions. Next, each caption is tokenized and split into its key words, phrases and components. This step is crucial to the network performance as the individual caption features and locations are utilized in the encoder. Finally, the images are preprocessed by being resized to a certain dimension optimized for the vision transformer and normalized.

Once the data is prepared, the first step of the image network is implementing the vision transformer, or ViT. The general architecture of the vision transformer can be seen below in Figure 3:

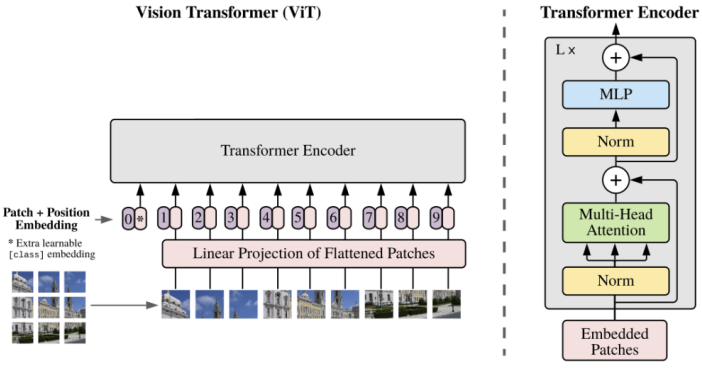

Figure 3: Vision Transformer architecture

The image is first split into 196 16x16 pixel patches. Each patch is flattened into

a tensor of size 1x768 (3x16x16) during the patch embedding process, and is concatenated with the positional embedding of the patch location in the image. These final embeddings are passed to the main ViT encoder to create a learned encoded representation of each patch embedding [7]. This ViT encoder contains an architecture of attention layers, normalization layers and MLP layers, all of which we experimented with and are further described in the following *Results* section. The final output of the ViT is passed to the main transformer encoder.

The second, and equally as important, aspect of the encoder is the caption encoder. This part of the encoder is carried out primarily using multi-head attention operation, shown in Figure 4:

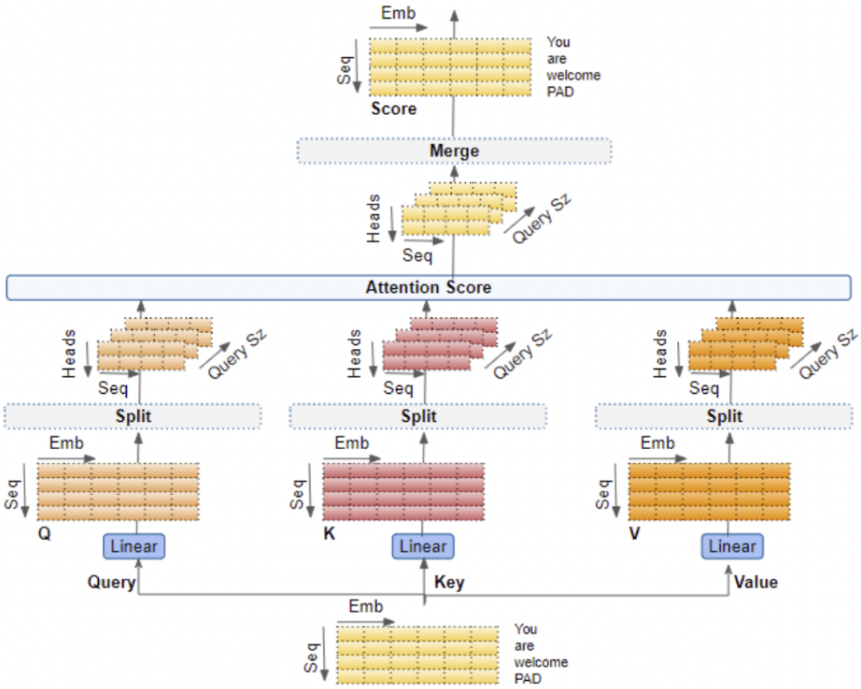

Figure 4: Graphical representation of Multi-Head Attention

Each token of the caption is treated as an individual aspect of the caption and is concatenated with a positional encoding based on where it is in the sentence, similar to the ViT. These encodings are then fed the learnable parameters of the multi-head attention method, represented by Query, Key and Value in Figure 4, which then produces an encoded representation for each word in the input sequence, that now incorporates the attention scores for each word as well [7].

Once the encoded caption and image representations are calculated through the attention layers, they are passed to the decoder block, shown highlighted in red in Figure 5:

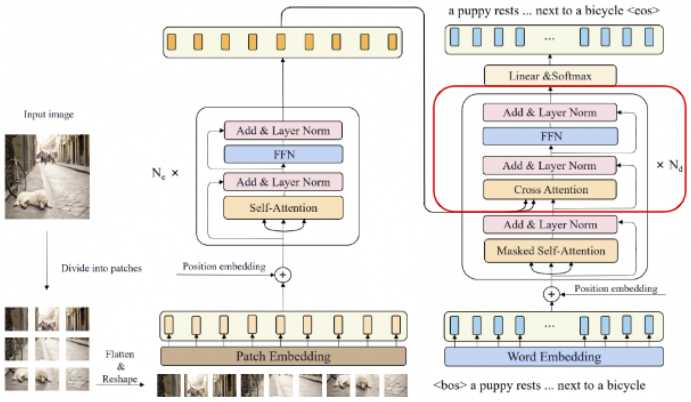

Figure 5: Entire Image Captioning Transformer Architecture

The most important method of the decoder is the Cross Attention layer of the network. This layer receives a representation of both the input image and target caption and produces a final representation with the attention scores for each generated caption token that also encapsulates the influence of the input image attention. This representation is then passed through another set of attention, normalization and MLP layers, and finally through a linear and softmax layer to create a final output.

Experiments and Results

In this study, we evaluated the performance of our model by varying several hyperparameters, including the number of heads, the number of layers, and the batch size. We explored different configurations, ranging from 1 to 6 heads and layers, as well as batch sizes from 8 to 256. Due to memory constraints and the limitations of our computing resources, we were unable to increase the batch size beyond 256, as this led to out-of-memory (OOM) errors.

We conducted our experiments using multiple GPUs, including the NVIDIA P100, K40, and V100, with varying memory sizes ranging from 1 GB to 100 GB. By increasing the available memory, we were able to accommodate larger batch sizes, which in turn impacted the overall training process. On average, each epoch took approximately 50 minutes to complete. Among the GPUs utilized, the V100 demonstrated the most significant performance improvement.

Considering the time constraints of our study, we limited the experiments to a single epoch for each configuration and compared the results accordingly. This approach allowed us to efficiently assess the impact of different hyperparameters on the model's performance without incurring excessive computational costs. Further analysis could potentially involve additional epochs or alternative configurations to refine our understanding of the model's behavior under different conditions.

num_heads	num_layers	batch_size	training time	accuracy
2	6	64	3267.45	0.2781
6	6	256	2905.82	0.3187
4	2	256	2905.00	0.3058
6	8	256	3269.32	0.3179

Table 1: A table showing the number of heads, layers, model dimensions, and batch size and accuracy.

The results of our experiments are summarized in Table 1. From the table, we can observe that the highest accuracy of 0.3187 and the lowest training time of 2905.82 seconds are achieved in the second row, with 6 heads, 6 layers, and a batch size of 256. This suggests that increasing the number of heads and layers, while keeping the batch size at the maximum allowable value, leads to better performance in terms of both accuracy and training time.

In our experiments, we found that increasing the batch size contributed to a significant improvement in accuracy. Consequently, we conducted further iterations using the highest possible batch size (256) to maximize performance. As the number of layers increased from 2 to 6, the accuracy improved, but a further increase to 8 layers did not yield a substantial improvement. This could imply that 6 layers might be the optimal choice for this specific problem. Due to time constraints and computational limitations, we were unable to run multiple epochs for each configuration. To overcome this limitation and still evaluate the performance of our model effectively, we resorted to monitoring the batch accuracy during training. Batch accuracy is a measure of the model's performance on a single mini-batch of data, providing a snapshot of how well the model is learning from the current batch.

As we progressed through the training process, we observed that the batch accuracy was steadily increasing, as expected. This indicates that our model was successfully learning and adapting its parameters to minimize the loss on the training data. Although batch accuracy is not as comprehensive as evaluating the model's performance over multiple epochs, it still provides a useful approximation of the model's ability to learn and generalize from the given dataset.

It is worth noting that the differences in training time are relatively small when comparing configurations with the same batch size. This implies that the choice of the number of heads and layers may have a limited impact on the training time, while their influence on the model's accuracy is more pronounced.

We employed a custom learning rate scheduler to adapt the learning rate during the training process. The custom scheduler plays a crucial role in the model's learning process by dynamically adjusting the learning rate based on the current training step and a specified warm-up period.

The impact of the number of layers and heads on training time and accuracy

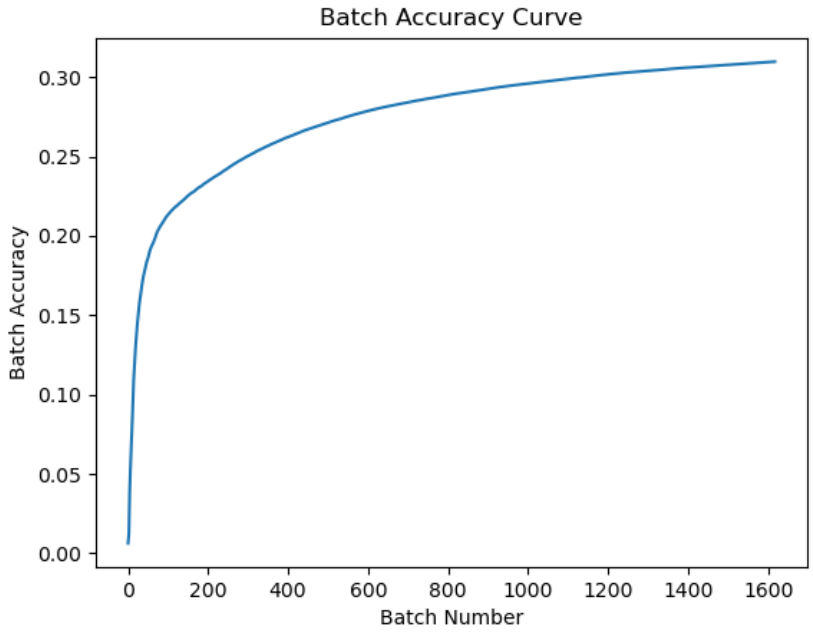

Figure 6: Training accuracy per batch for 1 epoch

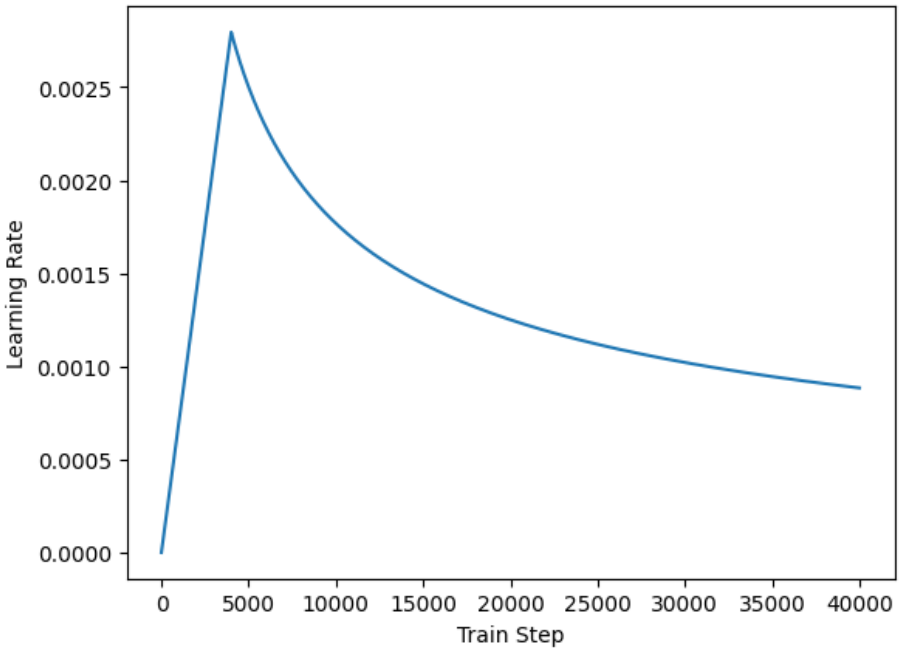

Figure 7: Custom Schelduer

can be attributed to various factors. Let's discuss these factors separately for each hyperparameter:

Number of Layers: The number of layers in a neural network directly affects its depth, which in turn influences its representational capacity. A deeper network can learn more complex and hierarchical features from the input data. As the number of layers increases, the model becomes more capable of capturing intricate patterns and relationships, leading to improved accuracy.

However, adding more layers also increases the number of parameters and computations, which can affect training time. In some cases, the increase in training time may be negligible, as observed in the provided table, while in other cases, it may be more substantial, depending on the model architecture and other factors. Additionally, deeper networks are more prone to overfitting, especially when training on smaller datasets, and may require more sophisticated regularization techniques or larger datasets to generalize well.

Number of Heads: In the context of attention mechanisms, such as those used in the Transformer architecture, the number of heads impacts the model's ability to capture different aspects of the input data simultaneously. Each head can focus on different parts or features of the input, allowing the model to learn a richer and more diverse representation of the data. Consequently, increasing the number of heads can lead to better performance and higher accuracy.

On the other hand, increasing the number of heads also increases the number of parameters and computations involved, which can affect the training time. In some cases, the increase in training time may be minimal, while in others, it may be more significant, depending on the model architecture, the size of the input data, and other factors. The optimal number of heads might depend on the specific problem and dataset, and it may require experimentation to determine the best balance between accuracy and training time. Our results suggest that increasing the batch size, as well as the number of heads and layers, can lead to improved performance in terms of accuracy. However, further analysis and experimentation with different hyperparameter settings may be necessary to fully understand the trade-offs between accuracy, training time, and computational resources.

Conclusion

Deep learning applications that combine aspects of computer vision and natural language processing methods, such as image captioning systems, will continue to increase in efficiency, accuracy and use in the near future due to today's increase in artificial intelligence studies and use cases. Many individuals and organizations rely on efficiency and accuracy of these models to complete tasks and increase factors such as job performance and overall quality of life. Transformer-based deep learning architectures have proven to be the leading methods for applications that combine these two industries. In this study, we have shown that a transformer-based deep learning model containing aspects such as a vision

A red and white double decker bus is parked in the street

A man doing a skateboard trick on a skateboard.

A man jumping over a skate board on a ramp.

A car sitting on the ground in front of a building.

A man in white shirt holding a bat and a ball

A man is sitting on a bed with a laptop on the bed

transformer, language encoder and decoder blocks can be extremely effective at tasks combining aspects of computer vision and natural language processing. Furthermore, altering the architecture of attention, normalization, and MLP layers within the transformer architecture can greatly affect the overall output.

References

[1] https://www.ripublication.com/ijaer18/ijaerv13n9_102.pdf

[2] COCO Dataset. [Online]. Available: https://cocodataset.org/#home

[3] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3128-3137. [Online]. Available: https://arxiv.org/pdf/1412.2306.pdf

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in International Conference on Learning Representations, 2021. [Online]. Available: https://arxiv.org/pdf/2010.11929.pdf

[5]A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All You Need," in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5998-6008. [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf

[6] Yiyu Wang1, Jungang Xu2, Yingfei Sun1 "End-to-End Transformer Based Model for Image Captioning" 2022 Available: https://arxiv.org/pdf/2203.15350.pdf

[7] T. Niranjan, "Transformers Explained Visually - Part 3: Multi-Head Attention Deep Dive," Towards Data Science, 2020. [Online]. Available: https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853#:~:text=Multiple%20Attention%20Heads,independently%20through%20a%20separate%20Head

Contributions

Kishore Pagidi

- Developed data pipeline, put together transformer architecture, bug fixes, training, and evaluation.
- Contributed to report.

Luke Davidson

- Developed main decoder architecture.
- Contributed to report and presentation slides.

Nand Dave

- Developed main encoder architecture, error logging, and managed the project.
- Contributed to report.

All members contributed equally