Advanced Machine Learning Semester 1b Project

Using Transformers for Image Captioning With COCO

Ben Blankenborg (s4895606)
Raghav Chawla (s4241657)
Stef Timmermans (s5862159)
Hylke Westerdijk (s5423937)

University of Groningen
Faculty of Science and Engineering
Advanced Machine Learning
Professor H. Jaeger

February 4, 2025

Contents

Abstract

This paper explores the use of transformer-based models for image captioning, leveraging convolutional feature extraction and attention mechanisms to generate accurate textual descriptions. The model follows an encoder-decoder architecture, where MobileNetV3 extracts image features, and a transformer decoder processes tokenized captions. The COCO dataset is used for training, ensuring a diverse range of visual contexts. We employ Hyperband-based hyperparameter tuning to optimize key parameters such as dropout rate, number of layers, attention heads, and learning rate. Evaluation is conducted using standard NLP metrics, including BLEU, CIDEr, ROUGE-L, and METEOR, to assess caption quality. The model shows promising results, but challenges such as computational limitations and dataset preprocessing constraints impact performance. Our findings highlight the effectiveness of transformer-based captioning models and suggest future improvements, including more efficient feature extraction and enhanced decoding strategies.

1 Introduction

This paper introduces a transformer-based image captioning model that utilizes a convolutional neural network and a tokenizer for encoding visual features and processing textual representations before decoding. This encoder-decoder architecture allows for effective feature extraction and translation of images into meaningful natural language descriptions. Our approach is trained on the Common Objects in Context (COCO) dataset, a benchmark dataset containing images with diverse scenes and annotated captions [1]. By incorporating tokenization strategies, we can enforce a general sentence structure of the captions to improve efficiency and fluency.

Our contributions throughout this project are threefold:

- The preprocessing of COCO data for the image captioning model, shaping images to a consistent `(224, 224, 3)` and using `MobileNetV3Small` to shape the raw images into extracted feature vectors.
- The implementation of self-attention and cross-attention mechanisms for caption generation using a transformer model with the preprocessed train and validation datasets.
- The analysis of the results of our model gained through this competition to assess our generated captions validated via various metrics and different decoding strategies. Also, visualizations are provided to better understand the results of the model.

The report itself discusses our strategies and findings throughout the implementation of this three-part data pipeline. The remainder of the report is structured as follows:

- Section 2 provides light background knowledge regarding image captioning.
- Section 3 breaks down the Common Objects in Context dataset and how it is incorporated into our project.
- Section 4 lists the strategies employed throughout the model building process and describes the model's architecture.
- Section 5 goes into detail about the evaluation metrics used to evaluate our model.
- Section 6 shows the model's results.
- Section 7 ends the report with a discussion of difficulties faced during the implementation phase, a conclusion and a reflection.

2 Background

The advent of machine learning has revolutionized numerous domains, with image captioning standing as a prime example of the fusion of computer vision and natural language processing. Image captioning involves generating a textual description of an image by capturing its objects, actions, and contextual relationships. Image captioning has garnered significant attention for accessibility reasons; captioning can be very helpful for the visually impaired who may not be able to recognize a photo's entire information [2]. It also has relevance in other fields such as human-machine interaction, biomedicine, prescription management, education, quality control, and traffic analysis [3].

Early forms of image captioning relied on different technologies compared to the strategies employed today. Until 2014, the state-of-the-art method to perform captioning was leveraging more traditional machine learning techniques for segmentation recovery. In 2014, popularity shifted to encoder/decoder models, which were more effective with the type of data used for captioning. More recent techniques relied on convolutional neural networks (CNNs) for image feature extraction and recurrent neural networks (RNNs) for sequence generation [3]. The rise of transformers via its landmark paper, provides a more powerful alternative, addressing prior limitations with their ability to model complex dependencies through self-attention mechanisms [4]. This brought focus back to the attention-based mechanisms that had some popularity in the mid-2010s [3].

Transformers are a form of neural network architecture that changes an input sequence to an output sequence through learning context and tracking relationships. Transformers were a breakthrough in AI tasks through its self-attention mechanism; where the model looks at

different parts of the input sequence all at once to determine which pieces are the most important for context [5]. This differs from other AI methodologies such as GPT, where the focus is often on predicting "the next word" rather than considering a more nuanced context.

3 Dataset

The project utilizes the Common Objects in Context (abbreviated as COCO), supplied by the COCO Consortium under a Creative Commons license [1]. As the name implies, this dataset contains images of common objects, such as stop signs, people, or animals, in their natural context (for a stop sign, this would be next to a road). There are in total 91 entry-level categories of objects, meaning that every category is described the way a human would most likely describe it ('cat' instead of 'mammal' or 'Siamese cat'). These are all the 'thing' categories, where the image has clear boundaries that distinguish individual objects from their surroundings, as opposed to the 'stuff' categories, where the boundaries are much less clear, such as the sky or a road [6].

There are iconic and non-iconic images 2. Iconic images aim to display a single object or scene over anything else. If it is an object-iconic image, 2a, there will rarely be any other things in the frame, the object will take up most of the image space, and it will be centered in a canonical perspective [7]. If it is a scene-iconic image, 2b, there will rarely be any people in the frame, and the scene will have been captured from a canonical viewpoint [8]. The earlier mentioned non-iconic images from 2c do not deliberately center any object or scene. They ultimately lead to better generalizing of a model than iconic images [9]. The creators of the COCO dataset had the explicit goal of a majority of non-iconic images. They therefore retrieved their images from Flickr[10], a image-sharing website that anyone partake in, filled with content useful for image related machine learning, as most of the images include the natural context of the objects they depict [11]. Each image in the COCO dataset comes with 5 different, correct annotations [12]. These captions were collected using Amazon's Mechanical Turk (AMT) [13], and then evaluated using several metrics to find a top five.

We had to use the 2014 MSCOCO dataset, as it was the last dataset utilized for an image captioning competition in 2015. The dataset is divided into two main components: "Images" and "Annotations." The images cover a diverse range of subjects in various locations and performing different actions, and they are further split into training, validation, and test sets. Similarly, the annotations include ground truth captions for the training and validation sets, provided by the COCO Consortium, while the test set includes annotations without ground truth captions. The complete list of dataset downloads used in this project is as follows:

- 2014 Train images [83K/13GB]
- 2014 Val images [41K/6GB]
- 2014 Test images [41K/6GB]
- 2014 Train/Val annotations [241MB]
- 2014 Testing Image info [1MB]

[1]

4 Methods and Experiments

4.1 Data Preprocessing

In order to build our image captioning model, the first step was to preprocess our dataset. The dataset consists of images for training, validation and testing, and captions for training and validation. For both the training and validation sets, each image has 5 ground truth captions. The data preprocessing was completed in was done in three steps: 1. Set up, 2. Preparing the dataset, 3. Saving the dataset.

4.1.1 Set up

At the start of our notebook we ensure the necessary modules are installed, seed both our random number generators (from numpy and tensorflow) to ensure replicability, and set the variable `IMAGE_SIZE` to (224, 224, 3). We then load in the train and test captions using the coco API [1]. We have created a `utils.dataset` module with functions to set up our dataset, and we apply them to

the training, validation, and test sets. We use `setup_trainval_sets` for the training set, which creates a TensorFlow `Dataset` object in which image paths are paired with their captions. As there are five captions per image, this results in a dataset where each image path occurs five times, each time paired with a different caption. For the test set, which has no captions, we use `setup_test_set` to pair image paths with their image id's, also in a TensorFlow `Dataset`.

We also set up the model that we will use for feature extraction. We chose to use the pre-trained `MobileNetV3-Small` from `tf.keras.applications`. This model is very useful to us due to its relatively low computational cost compared to earlier models, achieved through platform-aware Neural Architecture Search [14], and the use of a h-swish activation instead of the swish activation, improving performance without adding delays [15], while still employing the depth-wise seperable convulsions from MobileNetV1 and the block with inverted residuals and linear bottlenecks from Version 2 that both already brought down computational costs [15, 16, 17]. We ensure to exclude both the classification layer and the option of training from this model, as that is not what we want to use it for. Lastly, we set up our tokenizer (from `tensorflow.keras.layers`), with a vocabulary size of 10000. We train the tokenizer on the captions of the training dataset, and, using the tokenizer, create a mapping from words to indices and indices to words.

4.1.2 Preparing the dataset

In order to prepare the training and validation datasets, we use the `load_image` function defined in our `utils.dataset` module. This function first loads the image as a JPEG file from the image path, converts it to a tensor and then resizes the tensor to our desired `IMAGE_SIZE`. We also use our tokenizer to tokenize the caption associated with the image. We load the images and tokenize the captions for batches in the dataset. As this is done in batches of 256, we need to ensure that each image tensor is still associated with the right tokenized caption through repeating the tensor. To this end we apply the `split_captions_on_images` function, to flatten the captions and then repeat the images along the caption dimension. Afterwards the results are unbatched to maintain the image tensor and caption pairs, and then shuffled and batched again so that the caption tokens can be shifted to create input and label tokens by shifting the tokenized captions: input tokens exclude the last token, and label tokens exclude the first token. Finally we ensure the final output of `image, input_tokens, label_tokens` contains only tensors.

4.1.3 Saving the dataset

Part of preprocessing is also saving the dataset to enable future use with a lower computational cost. To this end we first take the original dataset created in step 1, comprised of image path and caption string pairs. To batches of this dataset (size 256) we then apply the `load_image` function to get image tensors from the image path, after which we extract the feature map from this image tensor using `MobileNetV3`. This results in a TensorFlow dataset with pairs of feature maps (`tf.float32`) and captions (`tf.string`). After this, we tokenize the captions using our tokenizer, and again shift the tokens to create input and label tokens. Unbatching to retain individual samples and shuffling to ensure randommness, and as a last step we save this dataset in ten shards. Shards are subsets of the dataset, and partitioning the database like this makes for better performance and response times [18].

We saved both our training and our validation set this way.

4.2 Transformer Architecture

With our preprocessed data, the encoder-decoder transformer architecture was fully built and trained. The sequence generated by the model was processed through self-attention, while the image was handled via cross-attention. TensorFlow's image captioning transformer model, which we adapted for our project, consisted of three main steps: the input, the decoder, and the output [19]. The structure of the transformer architecture can be understood by breaking down each layer:

4.2.1 Sequence Embedding

`SeqEmbedding` is responsible for converting input text sequences into dense vector representations, a process known as embedding. It does this by looking up embedding vectors for each token and for each position in the sequence, then adding them together. This combination allows the model to capture both the identity of words and their positional context within the sequence. In this implementation, the position embeddings are learned rather than fixed, which simplifies the code but may limit generalization to longer sequences. Additionally, the `mask_zero=True` parameter is used to initialize the Keras masks for the model, ensuring that padding tokens are appropriately ignored during processing. [19]

4.2.2 Causal Self-Attention

`CausalSelfAttention` applies self-attention mechanisms to the input sequence, allowing each position to consider all previous positions up to the current one. This is important for autoregressive decoding, where the model generates text one token at a time and must base each prediction on the previous context. The implementation uses a multi-head attention mechanism with a causal mask to prevent the model from accessing future tokens during training. The output of the attention mechanism is then added to the original input (residual connection) and normalized, allowing for effective gradient flow and stable training. [20]

4.2.3 Cross-Attention

`CrossAttention` allows the model to combine the two halves of the architecture: the encoder and the decoder. The layer combines information from the generated text sequence and the encoded image features. By calculating attention scores between these modalities, the model can focus on the most relevant parts of the image when generating each word in the caption. The implementation involves a multi-head attention mechanism where the query comes from the text sequence, and the key and value come from the image features. The resulting attention scores can also be analyzed to understand which parts of the image influence specific words in the generated caption. [20]

4.2.4 FeedForward

After the attention mechanisms, the `FeedForward` layer applies a series of fully connected neural network layers to each position in the sequence individually. Typically, this involves a two-layer dense network with a ReLU activation function in between, allowing the model to process and transform input data effectively. A dropout layer is included to prevent overfitting by randomly setting a fraction of the input units to zero during training. As with the attention layers, a residual connection is added, and the result is normalized. [19]

4.2.5 Decoder Layer

`DecoderLayer` combines the `CausalSelfAttention`, `CrossAttention`, and `FeedForward` sub-layers into a single unit. The transformer contains multiple of these single unit layers. Each layer processes the input, focuses on the most relevant parts of the image, and refines the data to generate a caption. Stacking layers helps the model recognize patterns, improving its ability to create accurate and meaningful descriptions. [19]

4.2.6 Token Output

In the final step of the decoding process, the `TokenOutput` layer maps the continuous representations produced by the decoder to discrete tokens in the vocabulary. This is done using a dense layer followed by a `softmax` activation function, which outputs a probability distribution over all possible tokens. During inference, the model selects the token with the highest probability at each step to construct the output caption. This layer is very important for translating the model's internal processes into output in the form of text. [19]

4.2.7 Captioner

The `Captioner` combines the entire caption generation process. It takes an image and a partial caption as input and iteratively predicts

the next word until the complete caption is generated. The Captioner organizes the interactions between the encoder (which processes the image to extract features) and the decoder (which generates the actual text caption), ensuring that the model effectively learns to produce accurate and descriptive textual representations of images. [19]

4.2.8 Masked Loss

The masked loss function is the first of two parameters adapted from the TensorFlow image transformer documentation.[19] In our notebook, it first converts input labels to integers and computes the sparse softmax cross-entropy of the labels and their corresponding predictions. Computing loss with masking is important in context to padding; when data is padded the padding will inherently contribute to the loss calculation, preventing the model from making the best choice. Masking ensures that only meaningful data impacts loss and stronger model stability [21].

4.2.9 Masked Accuracy

The masked accuracy function is the second of two parameters adapted from the TensorFlow image transformer documentation [19]. As accuracy is also important in the context of a transformer model, the accuracy in comparing the labels to their predictions also requires careful tuning around any potential padding. In accounting for the padding through masks, padding does not impact model accuracy (and would often inflate it, as padding can be consistent in some ways) [21].

4.3 Hyperparameter Tuning

Initially, we intended to implement cross-validation in our hyperparameter tuning process. However, we quickly realized this was not feasible because the train and validation sets were pre-split. As a result, we opted to use `KerasTuner`, a library from the `Keras` package designed to simplify hyperparameter tuning for TensorFlow and Keras models. Among the various optimization algorithms offered by KerasTuner, we specifically focus on `Hyperband`. Hyperband is unique in the sense that it provides a more efficient approach by dynamically distributing computational resources to the most promising hyperparameter configurations while removing underperforming ones early in the process. [22] The model is trained for 16 epochs with 100 steps per epoch (covering 1600 of 1618 training samples) and validated with 48 steps per epoch (covering 768 of 792 validation samples) to prevent indexing errors. We also implemented `EarlyStopping` with patience = 3, to assist in preventing overfitting. Refer to the Appendix (Table 1) for the entire hyperparameter search space. The optimal hyperparameters were found as:

- Dropout Rate: 0.2
- Number of Layers: 2
- Number of Heads: 2
- Optimizer: Adam
- Learning Rate: 0.001

4.4 Evaluation

Model testing and evaluation were conducted using three temperature parameter modes. Greedy Decoding (temperature = 0) selects the most probable token at each step. Random Sampling (temperature = 1) samples tokens based on predicted probabilities, while Uniform Random Sampling (temperature $>$ 1) reduces the model's influence, approaching uniform distribution [19]. Caption generation was implemented using the `simple_gen` method, incorporating temperature-based variation. The initial intention was to perform image captioning on the test set across a number of temperature values, however due to time limitations, only temperature = 0.5 was tested, a balance between Greedy Decoding and Random Sampling. To assess model performance, captions for both the test and validation sets were generated and saved as JSON files in image_id - caption pairs. These files were then zipped and uploaded here to retrieve the model's score, though they were not submitted to the leaderboard.

5 Metrics

5.1 CIDEr (Consensus-based Image Description Evaluation)

CIDEr (Consensus-based Image Description Evaluation) is an automatic evaluation metric designed specifically for image captioning. It measures how well a generated image caption aligns with human consensus by comparing it to multiple reference captions. [23]

CIDEr calculates n-gram overlap between a generated caption and reference captions. CIDEr applies TF-IDF (Term Frequency-Inverse Document Frequency) weighting, giving higher scores to rare and important words while down-weighting common words. This ensures that informative words (e.g., "giraffe," "airplane") contribute more to the score than generic words (e.g., "a," "is," "on"). The similarity is then measured using cosine similarity between TF-IDF vectors of the generated caption and reference captions. The CIDEr score for n-grams of length n is computed as:

$$CIDEr_n(c_i, S_i) = \frac{1}{m} \sum_j \frac{g_n(c_i) \cdot g_n(s_{ij})}{\|g_n(c_i)\| \|g_n(s_{ij})\|} \tag{1}$$

where:

- c_i is the generated caption.
- S_i is the set of reference captions.
- $g_n(c_i)$ and $g_n(s_{ij})$ are the TF-IDF weighted n-gram vectors.
- m is the number of reference captions.

The CIDEr score combines the scores from n-grams of varying lengths as follows:

$$CIDEr(c_i, S_i) = \sum_{n=1}^{4} w_n CIDEr_n(c_i, S_i) \tag{2}$$

where w_n represents the weight assigned to each n-gram length, and is typically set to $w_n = \frac{1}{N}$. [23]

5.2 BLEU (Bilingual Evaluation Understudy)

BLEU (Bilingual Evaluation Understudy) is an automatic evaluation metric originally designed for machine translation but widely used in other natural language generation tasks, including image captioning. It measures how closely a generated text (such as an image caption) aligns with one or more reference texts. [24]

BLEU evaluates generated text using modified n-gram precision, which counts how many n-grams (sequences of words) in the generated text appear in the reference text. It considers unigram precision (single words) to measure word adequacy, while higher-order n-gram precisions (bigrams, trigrams, and four-grams) capture fluency and coherence. To prevent overly short captions from scoring high, BLEU applies a brevity penalty (BP). The final BLEU score is calculated as a geometric mean of the n-gram precisions, adjusted by the brevity penalty:

$$BLEU = BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3}$$

where:

- p_n is the modified n-gram precision for different n-gram sizes,
- w_n is the weight assigned to each n-gram precision (typically uniform),
- BP is the brevity penalty, defined as:

$$BP = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c \leq r \end{cases} \tag{4}$$

where:

- c is the length of the generated caption,
- r is the closest reference length.

[24]

5.3 ROUGE_L (Recall-Oriented Understudy for Gisting Evaluation - Longest Common Subsequence variant)

ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation - Longest Common Subsequence) is an automatic evaluation metric used to measure the quality of generated text, for example in image captioning. It measures the similarity between generated and reference

texts by computing the Longest Common Subsequence (LCS), capturing sentence structure while allowing some flexibility in word order. [25]

ROUGE-L is based on the Longest Common Subsequence (LCS), which measures the longest sequence of words that appears in both the generated and reference texts without needing to be in a row but maintaining order. This makes ROUGE-L quite flexible as it captures sentence structure while allowing for reordering. It works as follows:

1. Given two sequences X with length m and Y with length n, the LCS(X, Y) is the longest subsequence that appears in both, while maintaining word order.

2.
   - Recall: Measures how much of the reference text is captured by the generated text.

   $$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{5}$$

   - Precision: Measures how much of the candidate text overlaps with the reference.

   $$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{6}$$

3. The final ROUGE-L score is an F-measure combining recall and precision also known as the LCS-Based F-Measure:

   $$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{7}$$

[25]

5.4 METEOR (Metric for Evaluation of Translation with Explicit ORdering)

METEOR (Metric for Evaluation of Translation with Explicit ORdering) is an automatic evaluation metric for assessing the quality of generated text, for example in machine translation. It was designed to improve upon BLEU's limitations, achieving higher correlation with reference texts. Unlike BLEU, which relies solely on n-gram precision, METEOR includes recall, stemming, synonym matching, and word order penalties to produce a more precise evaluation. [26]

METEOR evaluates text by performing word-to-word matching between the generated text and reference text. It goes beyond exact matches by considering morphological variants (stemming), synonyms (WordNet), and paraphrases. After identifying all possible matches, METEOR computes the precision, recall, and alignment fragmentation to assess fluency. The final METEOR score is computed as:

$$Score = Fmean * (1 - Penalty) \tag{8}$$

where:

- $Fmean$ is a harmonic mean of precision and recall, favoring recall

  $$Fmean = \frac{10PR}{R + 9P} \tag{9}$$

- $Penalty$ accounts for word order fragmentation:

  $$Penalty = 0.5*(\frac{\text{number of chunks}}{\text{number of matched unigrams}}) \tag{10}$$

where fewer chunks indicate better fluency. METEOR computes scores for each reference text separately and selects the highest score as the final output. [26]

6 Results

The results for our transformer model were collected only for temperature = 0.5 due to aforementioned reasons. The evaluation for the test set, includes two results c5 (5 reference captions and c40 (40 reference captions), both will be discussed and compared. The results are evaluated using the metrics mentioned in the previous section. Refer to the Appendix (Table 2), for the evaluation metric scores and Figure 1 for a closer analysis of the two.

7 Discussion and Reflection

In this chapter, we delve into the discussion of the results obtained from our model trained on the MSCOCO image caption dataset and discuss the project and additional avenues for future research.

The results demonstrate a significant improvement in evaluation scores when increasing the number of reference captions from five (c5) to forty (c40), highlighting the impact of richer reference sets in image captioning evaluation. The BLEU scores show a clear upward trend across all n-gram levels, with BLEU-4 improving from 0.168 to 0.358, indicating that the generated captions are not only capturing individual words but also forming more coherent sentences when compared to reference texts. This suggests that when evaluating image captions with a small set of references, the model might be unfairly penalized due to natural linguistic variability, whereas a larger reference set provides a more comprehensive measure of caption accuracy.

The CIDEr score, which is specifically designed for image captioning, shows only a small increase from 0.539 to 0.569. This suggests that while additional references provide a better estimate of human consensus, the information captured by the generated captions remains relatively stable. In contrast, ROUGE-L improves significantly from 0.434 to 0.556, reflecting a greater alignment in sentence structure between generated captions and references. This indicates that with more reference captions, the generated descriptions are not just matching words but also following more natural linguistic patterns.

Additionally, METEOR improves from 0.188 to 0.252, indicating that a larger reference set helps capture synonymy and paraphrasing better. Since METEOR accounts for stemming and word-order flexibility, this increase suggests that the generated captions are more aligned with human references when diverse reference texts are available for comparison. The relatively smaller gains in CIDEr and METEOR compared to BLEU highlight an important aspect of automatic image caption evaluation: while BLEU primarily rewards exact matches, CIDEr and METEOR attempt to capture meaning, making them more robust to paraphrasing and diverse descriptions.

The quality of the generated captions varies significantly, as shown in the Appendix (Figure 3). Some captions are highly accurate and descriptive, while others are entirely incorrect. Additionally, some captions are partially correct but lack details or contain inaccuracies. This variability is likely due to the use of `MobileNetV3` for feature extraction, which produces smaller feature maps compared to architectures like `ResNet50` (as we were intially intending), limiting the amount of information available to the model. Furthermore, the model may also be overfitting to the training set, affecting its generalization performance.

7.1 Lack of Computational Resources

One of the biggest challenges we faced was the limited computational resources, which significantly constrained our ability to experiment with larger and more complex models. Initially, we planned to use ResNet50V2, as it showed strong performance in previous image captioning research. However, we quickly realized that the computational demands of training such a model were beyond our available resources. As a result, we had to scale down our approach, choosing for a more lighter architecture that allowed us to build a functional model within our time constraints. Unfortunately, we discovered the University of Groningen's Hábrók high-performance computing cluster too late, which could have significantly expanded our computational capabilities.

7.2 Adapting Around Documentation

While much of the project involved following TensorFlow's documentation and making adjustments for our chosen dataset, the process was still a valuable learning experience. It provided insight into how various components of machine learning integrate to create pipelines capable of producing meaningful results.

Although the final implementation was not as extensive as we initially envisioned, our goal was to develop a functional model within the constraints of a half-semester course. Early exploration of image captioning techniques led us to Hugging Face's image captioning documentation. Similar to TensorFlow's instructions, this page relied on several libraries and spec-

ified parameters, such as epochs and learning rates [2]. However, we found the depth of these instructions somewhat limited for our project's needs. Additionally, the example dataset referenced in Hugging Face's documentation had been removed due to a DMCA request, which rendered this source to become more of a general point of reference than one directly applicable to our source code.

7.3 Limitations of Preprocessing for Transformers

Once the scope of our project had become more finalized, we explored ways to make the transformer perform more effectively, even with stages of the pipeline being handled locally on systems with limited GPU compute. Middle-end Nvidia and Apple GPUs were used interchangeably during development to run the notebooks. We believed it would be more satisfying to have a transformer functioning purely locally, though we acknowledge that utilizing the University of Groningen's Hábrók high-performance computing cluster would likely have resulted in a more accurate outcome.

To address this trade-off, we investigated the use of Principal Component Analysis (PCA) during preprocessing to enhance the distinctions between images. PCA involves performing a series of orthogonal projections on variables in the direction of the most variance [27]. If we had successfully configured a way to make the transformer process the PCA data alongside the original dataset, we might have been able to isolate more specific differences between the images. However, PCA's primary purpose of dimensionality reduction is not particularly relevant in the context of caption generation, as the original captions' context remains a crucial input. This context also needs to be considered in the transformer's logic for defining the general grammar of the captions. While there may be a way to incorporate PCA for "smarter" learning in the transformer, we did not explore this further after achieving a functional transformer.

7.4 Comparison to Benchmarks

When comparing our results to the MSCOCO leaderboard, our model performs significantly worse than the top entries for c5. This difference is likely due to the fact that leading models are developed by research institutions or large corporations with access to extensive computational resources, allowing them to train more complex and expensive architectures. A similar or even larger performance gap is observed for c40, which is expected given that a greater number of reference captions provides a better basis for evaluation, increasing differences in model capability. [28]

7.5 Reflection

Overall, our group found this project to be an enjoyable and rewarding experience, despite the many challenges we encountered along the way. Our initial plan was to implement TensorFlow's image captioning transformer tutorial and then improve and extend it based on insights through research. However, we quickly realized the complexity of the code posed significant challenges, and adapting the tutorial to a different dataset consumed most of our available programming time. Nevertheless, we are proud of what we achieved. While the model's captions may not always be perfect, successfully building and training it was an accomplishment in itself. Additionally, at least one team member intends to continue working on this model independently to further refine its performance.

References

[1] COCO Consortium. *COCO: Common Objects in Context Dataset*. 2024. URL: https://cocodataset.org/#download.

[2] Hugging Face. *Image captioning*. 2025. URL: https://huggingface.co/docs/transformers/tasks/image_captioning.

[3] Abdelkrim Saouabe et al. "Evolution of Image Captioning Models: An Overview". In: *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)*. 2023, pp. 1–5. DOI: 10.1109/WINCOM59760.2023.10322923.

[4] Ashish Vaswani et al. "Attention is all you need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964. URL: https://dl.acm.org/doi/10.5555/3295222.3295349.

[5] Amazon Web Services. *What are Transformers in Artificial Intelligence?* 2025. URL: https://aws.amazon.com/what-is/transformers-in-artificial-intelligence/.

[6] Geremy Heitz and Daphne Koller. "Learning spatial context: Using stuff to find things". In: *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I 10*. Springer. 2008, pp. 30–43.

[7] Tamara L Berg and Alexander C Berg. "Finding iconic images". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. 2009, pp. 1–8.

[8] Xiaowei Li et al. "Modeling and recognition of landmark image collections using iconic scene graphs". In: *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I 10*. Springer. 2008, pp. 427–440.

[9] Antonio Torralba and Alexei A Efros. "Unbiased look at dataset bias". In: *CVPR 2011*. IEEE. 2011, pp. 1521–1528.

[10] Flickr. *Homepage*. 2025. URL: https://flickr.com.

[11] Lyndon Kennedy et al. "How flickr helps us make sense of the world: context and content in community-contributed media collections". In: *Proceedings of the 15th ACM international conference on Multimedia*. 2007, pp. 631–640.

[12] Xinlei Chen et al. "Microsoft coco captions: Data collection and evaluation server". In: *arXiv preprint arXiv:1504.00325* (2015).

[13] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. "Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data?" In: *Perspectives on psychological science* 6.1 (2011), pp. 3–5.

[14] Hadjer Benmeziane et al. "A Comprehensive Survey on Hardware-Aware Neural Architecture Search". In: *ArXiv* abs/2101.09336 (2021). URL: https://api.semanticscholar.org/CorpusID:231699126.

[15] Andrew Howard et al. "Searching for mobilenetv3". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1314–1324.

[16] Andrew G Howard. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[17] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[18] Siamak Solat. "Sharding distributed databases: A critical review". In: *arXiv preprint arXiv:2404.04384* (2024).

[19] TensorFlow. *Image captioning with visual attention*. 2024. URL: https://www.tensorflow.org/text/tutorials/image_captioning.

[20] TensorFlow. *Image captioning with visual attention*. 2024. URL: https://www.tensorflow.org/text/tutorials/transformer.

[21] TensorFlow. *Understanding masking padding*. 2024. URL: https://www.tensorflow.org/guide/keras/understanding_masking_and_padding?utm_source=chatgpt.com.

[22] TensorFlow. *Introduction to the Keras Tuner*. 2024. URL: https://www.tensorflow.org/tutorials/keras/keras_tuner.

[23] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. *CIDEr: Consensus-based Image Description Evaluation*. 2015. arXiv: 1411.5726 [cs.CV]. URL: https://arxiv.org/abs/1411.5726.

[24] Kishore Papineni et al. "BLEU: a Method for Automatic Evaluation of Machine Translation". In: (Oct. 2002). DOI: 10.3115/1073083.1073135.

[25] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of summaries". In: Jan. 2004, p. 10.

[26] Alon Lavie and Abhaya Agarwal. "METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments". In: (July 2007), pp. 228–231.

[27] GeeksforGeeks. *Principal Component Analysis (PCA)*. Sept. 2024. URL: https://www.geeksforgeeks.org/principal-component-analysis-pca/.

[28] Tsung-Yi Lin et al. *Microsoft COCO Captions Leaderboard*. Accessed: February 4, 2025. 2024. URL: https://cocodataset.org/#captions-leaderboard.

8 Appendix

Hyperparameter	Search Space	Description
Dropout Rate	0.1 to 0.5 (step = 0.1)	Regularization to prevent overfitting
Number of Layers	1 to 5 (step = 1)	Number of layers in the Captioner model
Number of Heads	1 to 5 (step = 1)	Number of attention heads in the model
Optimizer	Adam, SGD, RMSprop	Optimizer used for training
Learning Rate	{0.00001, 0.0001, 0.001, 0.01}	Learning rate selection for each optimizer

Table 1: Hyperparameter search space for model tuning.

Metric	c5 Score	c40 Score	Improvement
CIDEr	0.539	**0.569**	+0.030
BLEU-4	0.168	**0.358**	+0.190
BLEU-3	0.261	**0.487**	+0.226
BLEU-2	0.404	**0.637**	+0.233
BLEU-1	0.599	**0.786**	+0.187
ROUGE-L	0.434	**0.556**	+0.122
METEOR	0.188	**0.252**	+0.064

Table 2: Performance comparison of evaluation metrics for c5 and c40.

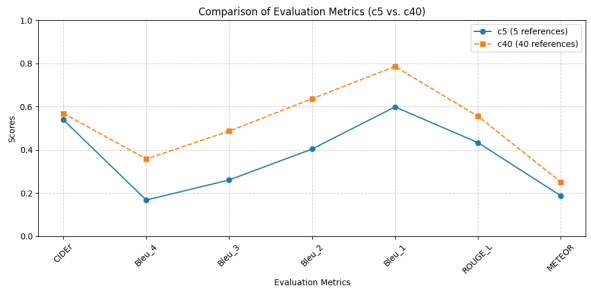

Figure 1: Comparison of evaluation metrics (c5 vs c40)

8.1 LLM Usage

Our group used LLMs, such as OpenAI's ChatGPT and Google's Gemini, only as a writing aid. Their role was limited to refining our own written sentences and enhancing clarity, with inspiration drawn from their suggestions. In the end, the final content remained our own written content.

(a) Object iconic image. The object is centered and large.

(b) Scene iconic image. The scene is centered, and there are no people.

(c) A non-iconic image. There are a lot of different objects visible, including people.

Figure 2: Three images, one object-iconic (2a), one scene-iconic (2b), and one non-iconic (2c). All images taken from the COCO 2014 train dataset.

a man on skis rides down a snow covered slope

a bathroom with a toilet sink and a mirror

a woman with a pink umbrella is standing next to a dog

Figure 3: Three images with generated captions (temperature = 0.5), all images taken from the COCO 2014 test dataset.