Attention Please! Deep Dive into Image Captioning Model Efficacy

JELLYFAM: Kevin Kang, Mithun Ramesh, Ethan Polley, Siddharth Somasi.
TA name: Theo Fernandez. Brown University

Abstract

This project investigates the performance of transformer-based models in image captioning tasks, critically examining the limitations of generalized models like VGG19, ResNet, and InceptionX. We compare the performance of un-tuned models with a fine-tuned model on a specific dataset, and create a baseline model using TensorFlow for further comparison. Our twin objectives are to evaluate the state-of-the-art in image captioning and assess the adaptability of these models to specialized datasets, a crucial aspect for applications like medical imaging. This research aims to improve the granularity and specificity of AI image captioning.

1. Introduction

The success of seminal research paper "Attention is All You Need" has highlighted the power of self-attention mechanisms in language modeling tasks, and has spurred a wave of research into transformer-based models. As companies like OpenAI continue to push the boundaries of what's possible with generative text models, it's clear that the race is far from over.

Image captioning is a challenging task in computer vision that requires a deep understanding of both visual and linguistic content. Our group aims to investigate and compare the performance of different transformer-based models for image captioning.

While image extractors like VGG19, ResNet, and InceptionX have achieved impressive results on a wide range of image recognition tasks, there is a growing concern that their generalized nature may lead to blind spots in the AI. This lack of specificity poses a significant problem for image captioning, as it can limit the model's ability to accurately describe and contextualize images. Our group aim to investigate this issue by comparing efficacy of un-tuned base image captioning models relative to a fine-tuned an image captioning model on a specific dataset.

As a baseline, we will also develop our own captioning model using TensorFlow and compare performance against more sophisticated models. We hope to achieve two major goals. First, we aim to see the performance of leading image captioning models against one another to assess the efficacy of the current landscape. Further, we aim to see how efficiently these models can be fine tuned to work on more niche datasets. This is especially important because many real world applications rely on granularity within image subspaces – for example, a system that captioned medical images would need to be able to learn and parse through complex terminology to be effective.

2. Related Work

Our approach to implementing an image caption model from scratch was inspired by Google's "Attention is All You Need" paper, which introduced the transformer-based architecture and showed promising results for natural language processing tasks. We were drawn to the transformer-based approach because of its ability to process long sequences and capture global context, which is particularly useful for image captioning where both the visual and textual components need to be considered.

In addition, we drew inspiration from recent advancements in transformer models and other generative models, including the "Medical Image Captioning via Generative Pretrained Transformers" paper in the medical field. This paper demonstrated the potential of transformer-based models for generating captions for medical images, which is a challenging and important problem

3. Method

3.1. Self-Implemented Model

For our baseline self-constructed model, the methodology can divided into 3 parts: Preprocessing, Model Architecture and Model Training/Evaluation

3.1.1 Preprocessing

For all portions of this project, we used the Flickr8k dataset, consisting of 8,000 images with 5 corresponding captions contained in "captions.txt". Specifically, the exact Flickr8k dataset was sourced from Kaggle and was stored locally on our system so as to not burden the Github repository.

We also implemented the infrastructure to use the Flickr30Kdataset, a dataset consisting of 30,000 images with 5 correspond captions contained in a csv file along with COCO120K Dataset, a dataset consisting of 120,000 images with 5 corresponding captions contained within a JSON file. These datasets were considerably large and there were much larger performance issues with the Flickr30K and especially, the COCO120KDataset. In order to get around these performance issues, we handled a few optimizations. The main optimization that was for all three datasets, we used Python's multithreading library to extract images and image features and to clean/process captions. This allowed us to better utilize system resources and significantly enhance performance for the Flickr8K dataset.

The main processing of the images and captions was extracting the image features using pre-trained Convolutional Neural Network's and extracting the captions. The pre-trained Convolutional Neural Networks we used were VGG16, ResNet50, and InceptionV3. For the captions, we used an NLP library called Spacy to clean the captions by removing stop words, excessive whitespace, and lemmatization.

3.1.2 Model Architecture

The architecture of our model was based on an encoder-decoder structure, which is a common design in many state-of-the-art models for tasks such as image captioning. This design allows the model to encode the input image into a representation of features that can be extracted from other images and to use these features along with a sequential model (which we experimented with) to predict a final caption.

The encoder was a Convolutional Neural Network (CNN) that was used to extract features from the images. We experimented with several pre-trained CNN models, including VGG16, ResNet50, and InceptionV3. We primarily focused on using InceptionV3, due to efficiency purposes.

For our model, we embedded our captions before feeding them to the decoder. We experimented with several types of embeddings, including a self-trained embedding matrix, GLOVE embeddings, and BERT embeddings. We also we made sure to tokenize all our sentences. This process involved adding start and end tags to each caption to mark when the sentence begins and ends, and allows the model to learn sentence structure for sequence generation. The tokenizer also incorporated padding to ensure that each caption was 15 tokens long, providing a uniform input size for our model. Out-Of-Vocabulary (OOV) tokens were included to handle words that were not present in the vocabulary.

The decoder was then structured to output a sequence of words, one at a time. For this, we experimented with four different types of sequential decoders: Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), and a Transformer Block with Multi-Head Attention. After running through the decoder, the model outputted a TimeDistributed softmax over the vocabulary, aiming to generate a caption of 15 words.The Transformer Block ended up being our preferred architecture and what we found (as expected) to be the form of decoder that we chose to use.

For our actual model, the inputs are the image features and a sequence that consists of the caption minus the first word, prepended with a start token. The output label is the sentence shifted by one token to the right, such that the model is trained to predict the next word given the image features and the preceding words in the caption. This type of setup is known as a sequence-to-sequence model and allows the model to learn relationships between words when predicted a sequence.

3.1.3 Model Training and Evaluation

For model training, we used 80 percent of the data for training and the remaining 20 percent for testing. We trained our data for 10 epochs using a train by batch system. The final batch size for each training instance we chose involved 256 examples per batch. We evaluated each of our answers using the BLEU metric.

For model inference, the main system we used was Greedy Search. Greedy Search is a simpler method of decoding a sequence, where the model attempts to predict each word in the sequence one-by-one in multiple iterations through the model. At each step, after predicting the next word in the sequence, the model feeds back the past sequence with its last guess appended and repeats this process until either the maximum length of the sequence is met or an end token is generated. During greedy search, the first sequence the model receives is simply the start token, and it must append tokens afterwards.

For each caption, we padded the token to an input length of 15, but also experimented with repeating the last token the model predicted to reach length 15. While we weren't able to run it on larger portions of data, we also implemented Beam Search, which we believe would have improved performance. Beam Search maintains multiple possible sequences (known as "beams") at each step. It explores these sequences simultaneously and retains only the top-k most probable sequences at each step, where k is a predefined parameter. Beam Search allows the model to avoid local optima with greedy search and keep solutions where the relationships of further-apart words are stronger. However, Beam Search is way more computationally expensive

3.2. Image Captioning Models

To generate image captions, we utilized pre-trained models from Hugging Face. We specifically used three pre-trained models for our image captioning task: nlpconnect/vit-gpt2-image-captioning, microsoft/git-base, and Salesforce/blip-image-captioning-base.

Intuitively, all three of these models are multimodal models, processing both images and text. For the nlpconnect/vit-gpt2-image-captioning and Salesforce/blip-image-captioning-base models, we integrated the model using the Hugging Face pipeline function to build the entire model. However, for the microsoft/git-base model, we had to reconstruct the image feature extractor, tokenizer, and decoder using other Hugging Face libraries. Specifically, we used the Hugging Face classes AutoProcessor and AutoModelForCausalLM. After manually combining these components, we were able to generate captions for the input images.

We selected these models based on their performance in previous image captioning tasks and their availability on the Hugging Face platform. These models — sourced from the Salesforce and Microsoft Research departments — have been pre-trained on large datasets and have achieved state-of-the-art performance on image captioning tasks.

3.2.1 Testing Model Efficacy

To determine the efficacy of the pre-trained models on image captioning, we utilized the BLEU (Bilingual Evaluation Understudy) metric. BLEU measures the overlap between a generated caption and the actual caption based on n-grams, which are contiguous sequences of words.

In regards to evaluating model performance, the Flickr8k dataset, after pre-processing, was randomly shuffled with 10 percent (n=809) of the dataset selected as the validation set. Note that for each image in the validation set, we selected the first caption as the ground truth or actual caption. Being specific, each pre-trained model (nlpconnect/vit-gpt2-image-captioning, microsoft/git-base, and Salesforce/blip-image-captioning-base) generated captions for each image in the validation set, which then was compared vis-a-vis the actual captions. The BLEU score using the corpus_bleu function from the nltk.translate.bleu_score library.

3.2.2 Fine-Tunning Model

To investigate the effectiveness of fine-tuning a pre-existing model for image captioning, we used the lambdalabs/pokemon-blip-captions dataset to fine-tune the pre-existing Hugging Face microsoft/git-base model. While the pokemon-blip-caption dataset may seem arbitrary, we operated under the assumption that any niche dataset would likewise fit in, hence the pokemon dataset serving as a proxy for this user case.

Model	BLEU
ViT-GPT2	0.312
BLIP	0.193
GIT	0.349

Table 1. BLEU performance on 809 images from Flickr8k – predicted caption against actual caption.

The lambdalabs/pokemon-blip-captions dataset consists of 833 images of Pokemon characters with corresponding captions. Due to limitations with GPU size on Google Colab and difficulties with accessing the provided Google Cloud Platform GPUs, we reduced the training set size to 100 images. We randomly shuffled the entire dataset to avoid bias in the evaluation.

To fine-tune the pre-existing Hugging Face microsoft/git-base model, we first downloaded the weights from the Hugging Face model hub. We then used the AutoProcessor and AutoModelForCasualLM classes from the transformers library to rebuild the model and processor.

We trained the model using the lambdalabs/pokemon-blip-captions training set and fine-tuned it for 10 epochs. During training, we used the AdamW optimizer with a learning rate of 5e-5 and a batch size of 8.

To evaluate the performance of the fine-tuned model, we used the BLEU metric. We randomly shuffled the entire lambdalabs/pokemon-blip-captions dataset and used a non-tuned model and fine-tuned model to generate captions for each image. We then compared the generated captions with the actual captions and calculated the BLEU score using the same libraries previously mentioned.

4. Results

The performance of the foundational, off-the-shelf captioning models on a validation set (n=809) from the Flickr8k dataset was mostly consistent, with ViT-GPT2 and GIT outperforming BLIP.

We created a web server where users could upload images from their computer and run them on both the basic foundational models as well as the fine-tuned foundational models. Referring to Figure 1, the example photo we took from Google Images shows the model performs well on photos even not taken from its training or testing datasets.

Finetuning ViT-GPT2 on the Pokemon dataset was fruitful. There was around a 25 percent jump in BLEU score between the untuned model and the fine-tuned model. Figure 2 highlights this disparity with a few examples.

The basic model we developed ourselves performed considerably worse. The best BLEU score we were able to

Figure 1. Web server with ViT-GPT2 captioning an uploaded image.

achieve on the Flickr8k testing set was 0.05, and the outputted captions sometimes matched, but were often unintelligible/or a completely wrong caption. Our ideal model architecture ended up being the following: a BERTEmbedding embedding layer that fed into a DenseLayer and was then concatenated with the InceptionV3 decoder. Afterwards, these were fed into 4 Transformer blocks made up of MultiHeadAttention, LayerNormalization and Dropout layers. For our loss function, we utilized CrossEntropy loss and our optimizer was an AdamOptimizer. This was primarily on the Flickr8K Dataset. While our model performed somewhat well on the train dataset, it seemed to struggle to generalize to larger/different data. As Figure 5 shows, the loss on the validation set didn't decrease and was rather scattered, while the training loss across 10 epochs decreased. This suggested overfitting and a lack of ability to capture relationships in new sentences. Furthermore, Figure 3 shows the distribution of the model's outputted caption lengths. There is a significant tendency for the model to output captions of lengths 10 and 14, which could explain some of its poor performance. We believe this issue came from Greedy Search.

4.1. Technical Discussion

We experimented with padding the captions to an input length of 15 as well as (for every non-start token), repeating the last token multiple times to reach length 15. We found that padding worked better, as the model seemed to understand that the ¡PAD¿ token (which we fed into our tokenizer) was insignificant, while seeing the same token over and oveer casued the model to settle into a local minima where it would keep regenerating the same 2-4 gram words every time. We settled on using padding, but we weren't certain that the model understood the complexity of padding enough.

A major obstacle for us in improving the performance of our own model was computational complexity. One method we tried was replacing greedy search – searching for the next word by simply taking the highest probability word – with BEAM search. However, we quickly ran into complexity problems and had trouble using our Google Cloud coupons, shown in Figure 4. From the training and testing loss curves (Figure 5), our model never performed better on a validation set even after several epochs of training. This led us to believe our underlying architecture was at fault and a more complex model might have addressed this deficiency. Attempts at training with more data (Flickr30k, COCO) ran into similar computational concerns.

4.2. Socially-responsible Computing Discussion via Proposal Swap

Critique 1: Training image set bias

Since most of our training occurred on niche datasets, this ended up being a less relevant concern for our process. We understand there may be biases with the foundational models we used – however, our research if anything shows these can be corrected by fine tuning.

Critique 2: Image copyright

The images in the Flickr8k dataset as well as the Pokemon dataset are for educational and non-profit uses. Our models are not intended to be monetized and our research has been to further our own and others' understanding of image captioning models.

Critique 3: Deployment bias

As stated above, our research here serves mainly as proof of concept: even with a relatively small dataset, transfer learning on leading image captioning models improves performance significantly. None of the models described or built in our research are intended to be deployed into a broader system – rather, architects should design their own models and use our research to improve their own performance based on specific task.

Critique 4: BLEU score weakness

While both BLEU and METEOR are popular evaluation metrics for machine translation and image captioning, our field research indicates that it is still regarded among leading researchers as one of the best metric for comparing linguistics. This is because BLEU is specifically designed to measure the overlap between the generated and actual captions based on n-gram matches, which is crucial for image captioning tasks where the generated captions should accurately describe the image. In contrast, METEOR relies on semantic similarity and may not capture the exact phrasing of the actual captions, which can lead to inaccurate evaluations of the model's performance.

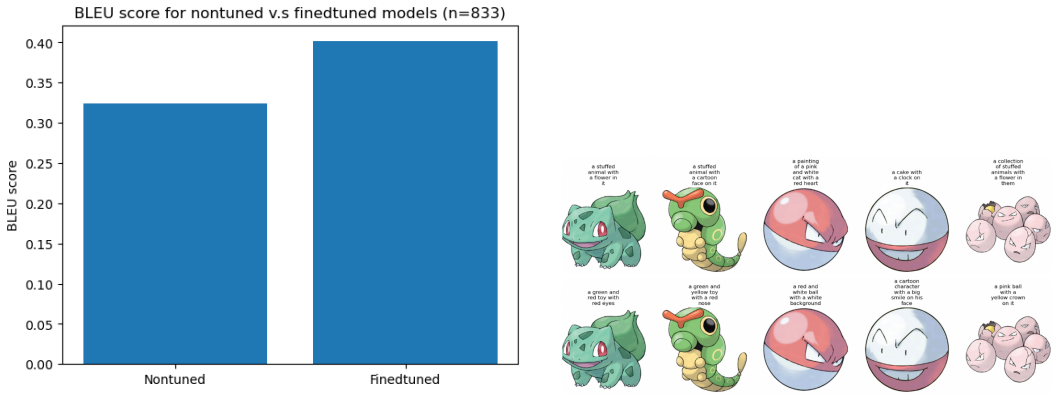

Figure 2. Double-wide figure. *Left:* The finetuned model had significantly better performance on the Pokemon dataset. *Right:* The captions more aptly described the Pokemon in the dataset post fine-tuning.

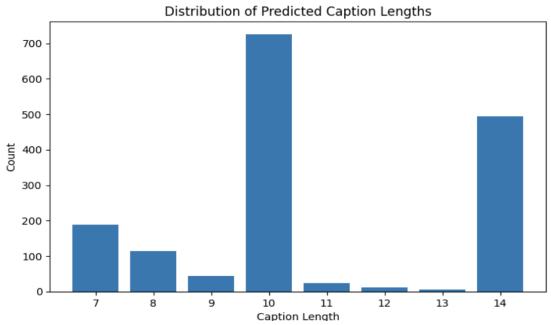

Figure 3. Caption lengths for basic model.

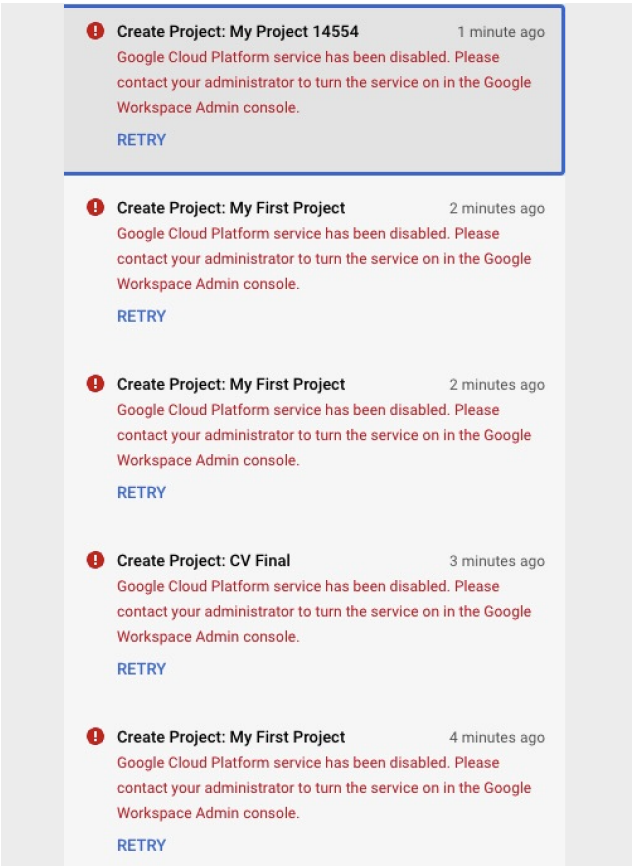

Figure 4. Error messages when trying to use Google Cloud coupons to train our model.

5. Conclusion

Our research serves as a proof-of-concept that the current landscape of image captioning models can learn niche datasets and terminology with relatively little training data. We also confirmed that these models perform well on data they have not seen previously and that they are relatively complex – our own attempts at replicating their performance stalled without computational horsepower.

There are many applications of what our research shows here. Primarily, captioning technology is advanced despite its recency – foundational models perform well out of sample and even on niche datasets. This means they can be deployed in certain contexts that need general captioning capacity (of course, task and bias should be considered before making this decision). Furthermore, the applications of the promising transfer learning research we conducted are limitless: medical imaging, astronomy photography, and species differentiation could all be aided by a large model trained on a subset of the more specialized data.

Appendix

Team contributions

Kevin Kang I specifically was tasked with spearheading research into pre-existing image captioning models to test their efficacy. As a result, I coded the notebook that

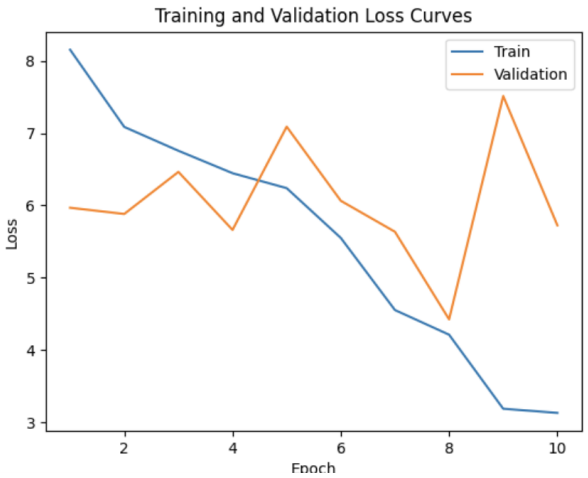

Figure 5. Training and testing loss for basic model.

imported all the various Hugging Face model weights, allowing us to compare the BLEU score vis-a-vis each model. As for my capstone part of the project, I implemented the fine-tuning aspect of our project that took a pre-existing multi-nodal model from Hugging Face and re-trained the model locally using PyTorch to a niche dataset, independent of the model's initial training data. I also made some of the visualizations that you see on this report, which can also be accessed in our GitHub repository. As a group effort, we all worked on the poster as well.

Sid Somasi I developed the ImageCaptioningModelTransformer and NonTransformer imagecaptioningmodel, conducted hyperparameter experiments, and implemented multithreading on the COCO120k dataset. Additionally, I implemented Beam Search for evaluation and developed a BERTEmbeddings-Tokenizer class for processing textual data. My contributions for my capstone mainly were in optimizing model performance and improving training efficiency. As a group effort, we all worked on the poster as well.

Ethan Polley I contributed to the server and poster components of the project, building and connecting the server to the image captioning models – this served as my capstone requirement. I also worked on visualizations and experimented with hyperparameters for the TransformerModel. In addition, I tested the METEOR and BLEU metrics for image captioning and explored changing the seq2seq model to a single-word prediction model. I also helped with the initial project proposal and critique swap. As a group effort, we all worked on the poster as well.

Mit Ramesh As an extension of our project, I attempted to expand the original Flickr8k dataset to include the Flickr30k dataset and the COCO120k dataset; the preprocessing classes can be found in our github repository. To improve the training efficiency, I implemented multithreading on both the Flickr8k and Flickr30k datasets. Additionally, I developed a tokenizer and embedding class to process the textual data in the datasets. To evaluate the performance of our seq2seq model, I also implemented the Greedy Search method as an evaluation method. As a group effort, we all worked on the poster as well.