CS7643 Final Project: Exploration of Transformer-based Models and Attention Mechanisms for Image Captioning

Daniel Richard Torres
dtorres44@gatech.edu

Harsh Chandak
hchandak3@gatech.edu

Patrick John Pastore
ppastore3@gatech.edu

Vipul Koti
vkoti7@gatech.edu

Georgia Institute of Technology
Atlanta, GA, USA

Abstract

Our final project explores transformer-based models for image captioning, with a focus on attention mechanisms and transfer learning. We used a multi-modal encoder-decoder architecture, combining an image encoder, a caption text encoder (for training), and a text transformer decoder for caption generation. We train the model via teacher forcing and proceed to use the trained models to generate test captions over test image dataset. Furthermore, we explored the impact on model performance of various attention mechanisms, including causal self-attention, cross-attention, and multiply cross-attention. Also, we evaluated the role of transfer learning via use of varied image encoders. Finally, we tuned our model across several hyperparameter values to optimize training and test performance on the Flickr8K captioned image dataset. Through our analysis, we demonstrate the impact of various attention mechanisms and of transfer-learned image encoders on transformer-based image captioning models. Specifically, we evaluate each model on the bases of BLEU (bilingual evaluation understudy) [10] similarity score, number of epochs required to train, top-1 masked accuracy, and masked loss.

1. Introduction/Background/Motivation

This project aims to improve the automatic generation of textual descriptions of images, otherwise known as image captioning. Image captioning has several applications such as improving accessibility of resources, enhancing content retrieval (such as image search), and improving visual sensor systems. Recent trends in image captioning show a shift from Recurrent Neural Network (RNN) caption generation / decoding to more advanced LSTM or text transformer decoding. Often, CNNs are used to encode images into embeddings that can be decoded into corresponding natural language descriptions. In this exploration, we try various CNN architectures.

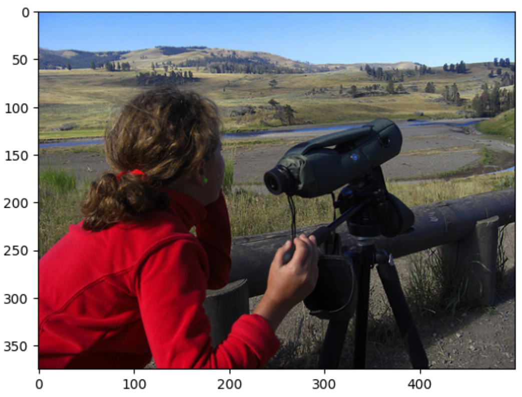

Figure 1. Flickr8K Dataset Sample Image and Captions -

A photographer looks over the hills .
A woman in a red jacket is videotaping a natural landscape .
A woman with a camera looks out over rolling hills .

In this study, we explore transformer-based models for image captioning, drawing inspiration from the ”Show, Attend, Tell”[13] paper and placing a particular focus on attention mechanisms [11] [9] and common image encoding architectures. We implement a multi-modal architecture that combines a CNN image encoder with a text transformer decoder. We vary and evaluate attention mechanisms [6] within the architecture, such as causal self-attention, cross-attention, and multiply cross-attention – all of which are extensions of the original soft attention mechanism introduced in ”Show, Attend, Tell.” These advanced attention mechanisms [2][8] play a crucial role in generating coherent and meaningful captions.

For this study, we used the Flickr8K dataset [4], a widely recognized benchmark dataset for image captioning. The dataset comprises 8,000 images, each with five captions, totaling 40,000 captions (fig.). The images cover a wide range of objects, scenes, and situations, making the dataset suitable for evaluating image captioning models on their ability to generate diverse and accurate captions. The performance of the proposed models is optimized using transfer learning, and the results are analyzed to understand the strengths and limitations of diverse attention mechanisms in transformer-based models for image captioning.

2. Approach

The following section explains our approaches with respect to the development, experimentation, and evaluation of various image caption models. Our base implementation flow is adapted from a TensorFlow tutorial[5], which itself is based on the "Show, Attend, and Tell" paper [13]. By using this base flow, we were able to focus our work on integrating parametrized attention within the decoder transformer, parameterizing encoder models, calculating various BLEU metrics, among other additions to support robust experimentation.

2.1. Multi-modal Architecture

The multi-modal architecture (fig 1) for image captioning involves two stages. Stage 1 [Training] involves training the model with batches of images and their corresponding captions (5 captions per image). In Stage 1, we utilize teacher forcing, whereby we mask all words of the "true" caption except for those we've already guessed; after the model guesses a word, we uncover its "true" value and let the model use it to guess the next words of the caption. Stage 2 [Captioning] involves invoking the trained model to generate resultant captions per input image. The model architecture does not change between Stage 1 and 2; however, it's important to note that we don't use true captions nor masking in Stage 2.

We used various pre-trained image encoders (MobileNet, VGG16, VGG19, ResNet50, ResNet101) for feature extraction and converting images to feature vectors. For the true captions (in Stage1) and predicted words (in Stage2), we do tokenizing (word-to-index mapping), token embedding, and positional encoding to embed caption vectors. These caption vectors and the image feature vectors are fed into the transformer decoder layer, where we implement Causal, Cross and multiply-cross attention. The attention layers (all implemented as soft attention) give more weight to the relevant image parts as well as the predicted words (Stage 2) / unmasked true caption words (Stage 1). The weighted learned features are then fed to the feed forward layer, which predicts the output token. This token is consumed by the output layer, which uses index-to-word mapping to predict the next word.

2.2. Data Pre-Processing and Preparation

The first step in our pipeline is to pre-process the images and captions (which will be used for training). We utilize TensorFlow's image module to resize each image to a fixed size (specific to each individual CNN's expected input shape). We then pass the resized images to the CNN encoder in question, taking the output of the final hidden layer (prior to any softmax or linear layer) as the image encoding. We finally cache these encoded images as TensorFlow "shards", such that they can be efficiently retrieved from disk by the model during training. In sharding the image encodings, we are able to keep system memory free for faster training/processing. The feature extractor (CNN) is built into our model for real-time image encoding, but sharding provides a shortcut for our model to retrieve encodings already generated prior to training.

We preprocess captions in two steps: tokenizing and embedding. We use the TensorFlow Tokenizer module to assign each unique word a unique integer identifier (ex. All instances of "dog" = 12), effectively turning a sequence of words into a sequence of integer tokens. Which creates a vocabulary (we restricted the size to 5000) and assigns indices to words. Next, we pass the tokenized caption through a positional embedding generator and a Keras Embedding layer (which embeds a sequence of tokens to a fixed size tensor), adding the two resultant vectors together to produce the final pre-processed caption. Now, the pre-processed images and captions serve as input for the model training workflow. Finally, we combine the captions and images into a TensorFlow dataset and batch them into groups of 32 to encourage faster convergence of gradient descent.

2.3. Training Configuration

Our training mechanism was configured with a few key niceties: masked loss and masked accuracy (for evaluation of model's ability to guess words during teacher forcing) and early termination. Our masked loss is calculated as cross-entropy of the consecutive guessed words. As the model learns and downweights incorrect words (according to the "true" captions), cross entropy / loss will fall. Masked accuracy is calculated as correctly guessed words per number of guesses. Per some early experimental results, we configured early termination to wait 15 training epochs for validation loss to improve. If no improvement is seen for 15 training epochs, we concede that the model may be overfitting and terminate training; otherwise, we allow the model to train for 100 epochs. Early training sessions suggested that 15 epochs of patience and 100 maximum epochs was a nice balance of preventing too much overfitting, allowing for a few consecutive "bad" epochs in validation performance, and keeping our model from training for too long (making our experiments more computationally intensive).

2.4. Attention Mechanisms and Text Transformers

Attention mechanisms have been widely studied and employed in various deep learning tasks, enabling mod-

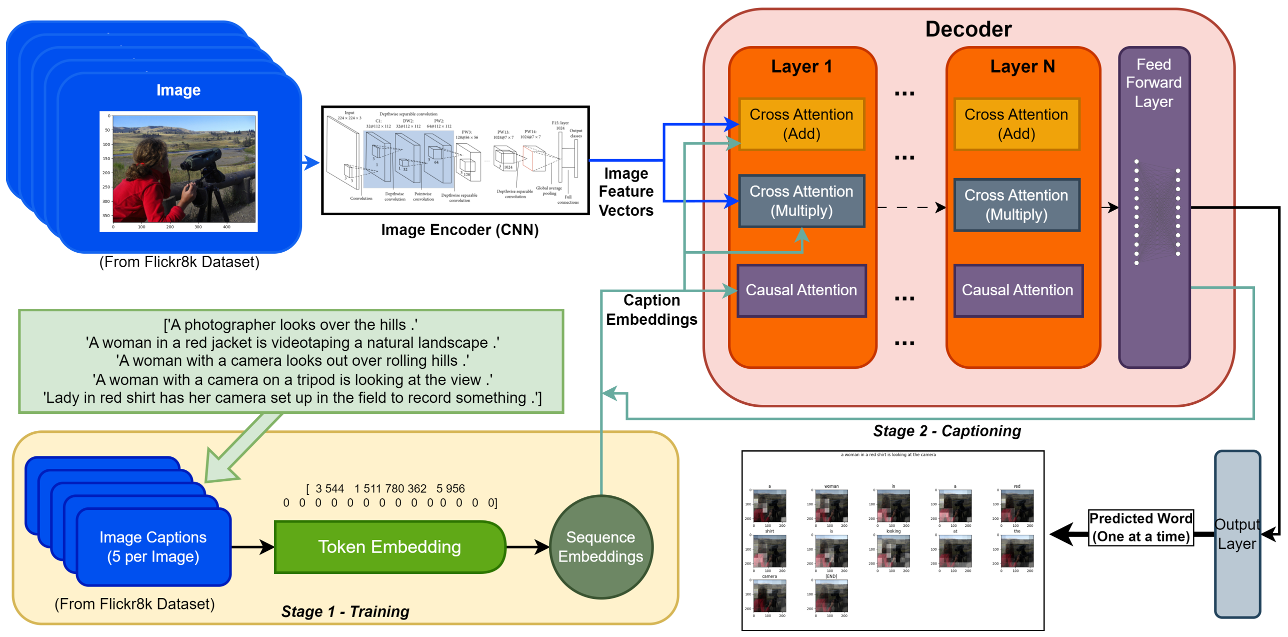
Image
(From Flickr8k Dataset)
Image Encoder (CNN)
Image Feature Vectors
Decoder
Layer 1
Cross Attention (Add)
Cross Attention (Multiply)
Causal Attention
Layer N
Cross Attention (Add)
Cross Attention (Multiply)
Causal Attention
Feed Forward Layer
Caption Embeddings
['A photographer looks over the hills .'
'A woman in a red jacket is videotaping a natural landscape .'
'A woman with a camera looks out over rolling hills .'
'A woman with a camera on a tripod is looking at the view .'
'Lady in red shirt has her camera set up in the field to record something .']
Stage 2 - Captioning
Image Captions (5 per Image)
(From Flickr8k Dataset)
[3 544 1 511 780 362 5 956 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Token Embedding
Sequence Embeddings
Stage 1 - Training
Predicted Word (One at a time)
Output Layer

els to selectively focus on specific parts of the input when making predictions or decisions [12, 3]. In our work, we explore several attention mechanisms, including cross-attention (addition and multiplication), causal attention, and their combination. Cross-attention allows the model to concentrate on different parts of the image when generating captions, while causal attention is a popular choice in language modeling tasks, as it restricts the model to attend only to previous tokens in the sequence [12].

We investigate additive and multiplicative attention as different ways of combining attention scores in cross-attention [3]. In both cases, the resulting attention scores are utilized to weight the input sequence, yielding a context vector that is then fed into the model. Our text generation model leverages a Transformer-based architecture that generates captions based on image features and previously generated words [12]. We experimented with multiple combinations of these attention mechanisms to understand the impact of each on image captioning training and predictions. The results will be discussed in detail in later section[].

2.5. Experiments

We train our image captioning model using the Adam optimizer with respect to minimizing the categorical masked cross-entropy loss [5]. As mentioned prior, the training runs for a maximum of 100 epochs with early termination expected when model validation loss fails to improve over 15 epochs. To identify the best performing model, we tune each model on multiple parameters, including: (1) Choice of pre-trained CNN model type, (2) Text transformer attention mechanism(s), (3) Units := uniform dimensionality used for both our caption sequence embeddings and decoder computations, (4) Dropout rate := Frequency with which random node output is chosen to be zeroed out, (5) Number of serialized decoders, (6) Number of heads in decoder multi-head attention, (7) Learning rate of Adam optimizer.

The following “Experiment and Results” section shows the effectiveness of the image captioning models. We evaluated all trained models with masked loss, masked top-1 accuracy, and a set of BLEU metrics . The BLEU metric measures the similarity between the generated captions and the ground truth captions [10]. We used Natural Language Toolkit: BLEU Score Corpus [10] method as we have 5 captions for one image. For our experiments, we calculated BLEU score for the test set with 1000 images and evaluated averages of top 20, 50,100, 200 and 400 BLEU scores for comparison. For example, average of top 20 BLEU scores can show how models perform at their most confident, whereas average of top 400 BLEU scores shows how models perform across a large subset of images. The experiment shows that the image captioning model is able to generate accurate captions for a variety of test images. It generates attention plots that capture the attention area in the image for each generated word.

3. Experiments and Results

In our image captioning project, we conducted two main experiments to optimize the performance of our model. First, we searched a space of hyperparameters for our decoder using VGG16 image encoder to optimize validation loss. Second, we searched a space of attention mechanisms and transfer-learned image encoders to further optimize validation loss.

Units

Units	Loss	Mask Acc	Val Loss	Val Masked Acc	Bleu 20	Bleu 50	Bleu 100
128	2.648	0.4314	3.0633	0.3777	33.13	31.18	28.57
256	2.2939	0.4695	2.972	0.3799	31.83	29.25	26.02
512	2.2094	0.4783	2.9466	0.3812	33.55	30.32	27.03

Dropout

Dropout Rate	Loss	Mask Acc	Val Loss	Val Masked Acc	Bleu 20	Bleu 50	Bleu 100
0.1	2.2079	0.5018	3.0515	0.38	33.49	30.53	27.45
0.2	2.1299	0.5046	3.046	0.3808	33.34	30.86	27.49
0.3	2.3193	0.4785	2.9915	0.3796	34.83	32.43	28.78
0.5	2.2939	0.4695	2.972	0.3799	31.83	29.25	26.02

Num Layers, Num Heads

Layers, Heads Num	Loss	Mask Acc	Val Loss	Val Masked Acc	Bleu 20	Bleu 50	Bleu 100
1	2.669	0.4179	3.0197	0.3812	34.08	31.8	29.27
2	2.2939	0.4695	2.972	0.3799	31.83	29.25	26.02
3	2.2765	0.4802	3.053	0.3697	31.59	28.81	25.23

Learning Rate

Learning Rate	Loss	Mask Acc	Val Loss	Val Masked Acc	Bleu 20	Bleu 50	Bleu 100
0.001	5.0424	0.135	5.0362	0.1315	26.47	22.12	14.51
0.0001	2.2939	0.4695	2.972	0.3799	31.83	29.25	26.02
0.00001	3.5016	0.3667	3.5401	0.346	33.64	32.43	30.83

Table 1. Experiment 1- Hyperparameters tuning on the model using VGG 16 pre-trained CNN encoder

Model	**MobileNet**		**VGG16**		**VGG19**		**ResNet50**		**ResNet101**	
Attention	Loss	Val Loss	Loss	Val Loss	Loss	Val Loss	Loss	Val Loss	Loss	Val Loss
Causal	2.4858	3.1439	2.5789	3.0772	2.6385	3.0148	2.5579	3.0538	2.6903	3.0893
Cross	2.5733	3.4946	2.7119	3.5161	2.5483	3.6026	2.316	3.4133	2.2293	3.3808
MultiCross	2.8984	3.3881	2.4689	3.2664	2.84	3.2558	2.4538	3.3065	2.4703	3.1774
Casual+cross	2.2454	3.0336	2.2679	2.9199	2.2106	2.913	1.9676	2.9577	1.925	2.957
Causal+MultiCross	2.4315	3.2018	2.0748	2.999	2.2734	2.9583	2.1244	3.0361	2.0957	2.9286

Table 2. Experiment 2- Evaluation metric Masked Loss & Validation Loss

3.1. Experiment 1: Hyperparameter Tuning with VGG16

The goal of this experiment was to optimize the performance of a base model by tuning several hyperparameters. Our search heuristic assumes that the decoder hyperparameters that optimize model performance are agnostic to image encoder. In practice, this assumption allowed us to perform a realistic number of experiments (i.e. search heuristic in place of exhaustive grid search). To optimize this VGG16 caption generator, we focused on the following parameters: (1) Common dimensionality of the embeddings, fully connected layer in attention, hidden and feedforward layers of the decoder (a.k.a. units), (2) Common Dropout rate for Feedforward and attention layers in the decoder, (3) Number of decoder layers, (4) Number of attention heads of the attention layers in the decoder, and (5) learning rate of the Adam optimizer.

We experimented with various permutations of these parameters (as shown in Table 1) and evaluated each resultant model's performance on training and validation sets. First, we find that as dimensionality (units) of the decoders increase, test and validation loss decrease, the model learns more complex and nuanced relationships between image features and encoded predicted captions. In contrast, we find that increasing the number of sequential transformers and the number of heads in transformer multi-head attention increases the model capacity to capture the finer relationship between image vectors and predicted captions. This additional complexity caused the model to overfit; we see improvement in training loss, but validation loss and accuracy falls (to a slight degree). We expected dropout to improve generality by regularizing the model, encouraging it to avoid overreliance on any small subset of parameters; per experimental results, dropout rate had little to no discernible effect on loss / accuracy of the model on either training or validation sets. With respect to Adam optimizer learning rate, we find that rate of 1xe^{-3} is too high (i.e. causes large steps toward gradient that overshoot optimal minima and

Model	MobileNet		VGG16		VGG19		ResNet50		ResNet101	
Attention	Max Loss	Median Loss	Max Loss	Median Loss	Max Loss	Median Loss	Max Loss	Median Loss	Max Loss	Median Loss
Causal	46	85*	49	82*	49	71*	42	80*	39	61*
Cross	45	59*	56	66*	57	85*	39	58	38	58
MultiCross	40	46*	32	50*	30	35*	29	39	29	39
Casual+cross	30	49	45	77	22	42	20	38	20	38
Causal+MultiCross	29	49	27	39	25	39	20	39	20	38

Table 3. Experiment 2- Evaluation metric Median & Max Loss Epochs Count

lead to poor results) and rate of $1\mathrm{x}e^{-5}$ is too low (i.e. model fails to converge within our reasonable expected time – 100 epochs). Per our various experiments, we found the best hyperparameter combination to be as follows: 512 units, 0.3 dropout rate, 2 decoder layers, 2 attention heads, $1\mathrm{x}e^{-4}$ learning rate.

3.2. Experiment 2: Tuning Image Feature Extraction Models

Following Experiment 1 and our determination of optimal hyperparameters for the decoder, we proceeded to evaluate the impact of various attention mechanisms permuted with various pretrained image feature extractors (CNNs) on Image Captioning performance. We explored 5 attention mechanisms within the decoder transformer(s): (1) Causal self-attention, (2) Cross (addition) attention, (3) Multiply Cross attention, (4) Casual & Cross attention (as is used in standard transformer architecture), and (5) Causal & Multiply Cross attention. We explored these five attention mechanisms in permutation with five image feature extractor models: (1) MobileNet, (2) ResNet50, (3) ResNet101, (4) VGG16, and (5) VGG19 (pretrained on ImageNet dataset). In total, this amounted to 25 experiment runs, the results of which are captured in Tables 2, 3, and 5. We evaluated these models across the same metrics as in Experiment 1.

Observations from the experiments: Starting with Table 2, capturing train and validation loss, we see that the choice of image feature extractor has an impact on the performance. Resent50 and Resnet101 perform better than other models; it's possible that their performance edge is because of their use of residual connections, which helps to mitigate the vanishing gradient problem in pre-training. Models with MobileNet generally show higher losses than models with other image encoders; since MobileNet uses significantly fewer parameters than the other models and therefore has less expressiveness than the other models, we expect its image encodings to be less sophisticated. For reference, MobileNet uses 4.3M parameters vs. ResNet50's 25.6M parameters and VGG19's 143.7M parameters [7]. Observe that the dual-attention-mechanisms (cross & causal, multiply-cross & causal) generally performed the best. We expected this behavior, as standard, popular transformers utilize both causal and cross attentions. Models that use causal self-attention alone do not attend to the image vector; loss here is highest and generated captions were worst (and most agnostic of the images) with these models.

In table 3, we show the number of epochs for which each model trained to reach two points: (1) the max loss of all models and (2) the median loss of all values. Where values are followed by an asterisk, training terminated early before reaching the values. These two points were arbitrarily chosen to compare training velocity of each model. First, we clearly see that simpler pretrained models like MobileNet train much slower than complex/advanced models like VGG19, Resent50, and Resnet101. In addition, this table accentuates the importance of both Causal and Cross attention(s); when a model uses only one of them, the learning is slow and inefficient in comparison to when we use both. In essence, when the model uses both causal self-attention and cross attention, it is able to capture finer details of relationships between the image vector and predicted word which improves the learning.

On our Flickr8k dataset, we saw all models generally converging to similar losses in under 100 epochs. To further compare the models, we utilized BLEU scoring (as shown in Table 5). Per our experiments, Causal&Cross attention generated the best captions (according to BLEU comparison with "true" captions"). Although Causal&MultiplyCross trains models in fewer epochs (generally) than Causal&Cross, its captions were not quite as similar to truth.

Our experimental results suggest that ResNet50 with Causal&Cross attention balances best between metrics of training velocity, BLEU similarity with true captions, and training loss. Furthermore, ResNet50 (with the exception of MobileNet) uses the fewest parameters (i.e. holds the smallest memory footprint).

4. Conclusion

In summary, we aimed to improve the automatic generation of captions for images by leveraging encoder-decoder architecture with image encoder (via CNN) and text decoder (via text transformer). We paid particular focus to attention mechanisms (such as causal, cross, and multiply-cross) within the text transformer and properties of image

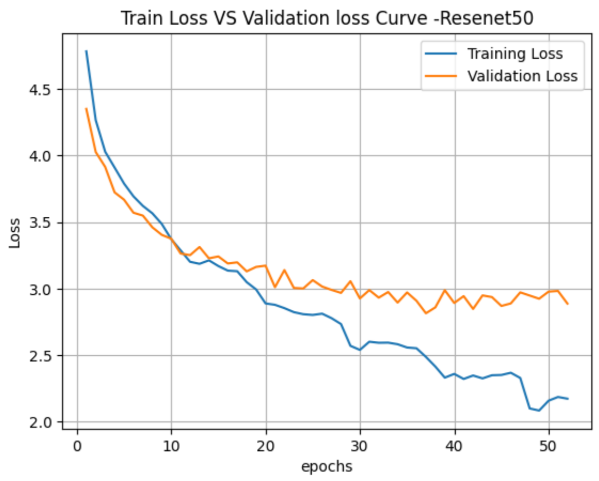

Figure 2. Loss Curve for best ResNet50

encoders. We optimized our models further with hyperparameter tuning for best performance on Flickr8k image caption dataset. We successfully demonstrated the potential of diverse attention mechanisms and image encoders to alter model performance on the Flickr8k dataset, albeit marginally.

4.1. Challenges

During our research, we faced multiple implementation challenges worth discussing:

1. Experimentation Resources: This image captioning model was the largest model any of us have ever trained. We learned quickly that each of our experiments would take approximately 30 minutes to train and 30 minutes to evaluate (i.e. generate captions via for test images and compare BLEU results). We had to make very intentional decisions about what and how we'd like to test. Furthermore, we had to learn how to store encoded images on disk for training, as everything could not co-exist in memory. Finally, we settled on Flickr8k dataset (rather than the larger Flickr30k dataset) in the interest of training time. We used Google Colab to alleviate our personal machine limitations, and will consider further utilization of these cloud resources for later deep learning projects.
2. TensorFlow Knowledge Barrier: We chose to use TensorFlow for our project, as to expose ourselves to a new deep learning framework (besides PyTorch). This framework came with several new challenges that we did not foresee; namely, TensorFlow Tensors are immutable. We approached this project with two additional ideas: (1) Add an object detector in parallel with our CNN image extractor to encode local image information and (2) Implement top-5 masked accuracy (i.e. give model credit if the "true" masked word is in its top 5 guesses, rather than just top 1). We spent considerable time trying to implement top-5 accuracy from scratch within our TensorFlow training flow, yet ran into many technical challenges with respect to looping over tensors. We spent even more time trying to integrate a pre-trained single-shot object detector into our image encoding workflow, yet ran into more challenges when trying to vectorize the output and append to encoded image matrices – approach seen in paper [1].

4.2. Findings

Our main findings suggest that the encoder-decoder architecture using a ResNet50 image encoder and standard text transformer (with Causal&Cross attentions) as text decoder most effectively maximizes image captioning performance across our search space of various image encoders and attention mechanisms. We further tuned decoder hyperparameters to optimize model performance across metrics of masked loss, BLEU similarity scores, and training velocity. Do note that, although MobileNet encoding architectures generally were the poorest-performing models, they are significantly smaller than the others; a strong argument could be made for using MobileNet encoding, as performance was only marginally worse than the optimal model.

4.3. Future Work

Over the span of this project, we researched several topics / mechanisms with which we could further improve our image captioning model. First, we'd like to experiment with a larger dataset. Each of our models underfit the data (maximum training masked accuracy never exceeded 54%). Adding more data and more epochs could give our model better opportunity to learn critical image-to-text patterns. Next, we'd like to incorporate object detection into our image encoding framework; as explained in Challenges section above, we were close to succeeding, but ran out of time. Next, we'd like to incorporate visual transformers as image encoders. Self attention in the encodings could lead to more sophisticated input representations for our decoder. Exploring alternative attention mechanisms, pre-trained models with different datasets, and unsupervised learning techniques could lead to further improvements in performance. Lastly, we'd like to explore the relationship between BLEU similarity and masked loss. Increased loss sometimes corresponds to increasing BLEU score; intuitively, we'd expect the opposite – increasing loss to mean worse model to mean worse captions (i.e. lower BLEU score). Understanding this relationship could allow us to develop more sophisticated loss functions or training methods.

References

[1] Mohammad A Al-Malla, Abdulaziz Jafar, and Nabil Ghneim. Image captioning model using attention and object features to mimic human image understanding. *Journal of Big Data*, 9(1):20, 2022. 6

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. *arXiv preprint arXiv:1707.07998*, 2017. Accessed: May 3, 2023. 1

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. 3

[4] Sayan Faraz. Flickr8K Dataset. `https://www.kaggle.com/datasets/sayanf/flickr8k`, 2021. [Online; accessed 16-Feb-2021]. 1

[5] Google. Image captioning with tensorflow. `https://www.tensorflow.org/tutorials/text/image_captioning`, 2021. Accessed: May 1, 2023. 2

[6] Chen Wei Huang, Wang. Attention on attention for image captioning. `https://openaccess.thecvf.com/content_ICCV_2019/papers/Huang_Attention_on_Attention_for_Image_Captioning_ICCV_2019_paper.pdf`, 2019. Accessed: May 3, 2023. 1

[7] Keras Team. Keras documentation. `https://keras.io/api/applications/`. Accessed: May 1, 2023. 5

[8] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Accessed: May 3, 2023. 1

[9] Wrucha Ameet Nanal. Image captioning using transformer architecture. 2020. Accessed: May 3, 2023. 1

[10] NLTK. Bleu score. `https://www.nltk.org/_modules/nltk/translate/bleu_score.html`. Accessed: May 1, 2023. 1, 3

[11] Atul Ranjan. Image captioning with an end-to-end transformer network. *Plain English Python*, 2022. Accessed: May 3, 2023. 1

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 3

[13] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention, 2016. 1, 2

Student Name	Contributed Aspects	Details
Daniel Richard Torres	Preprocessing, Modularization, Implementation, Analysis, Reporting	I worked early on researching various topics and supported the team in consolidating our goals. I also worked heavily on a modular version of our code that could be used to preprocess that data in a hyperparamerterized way. Most of my code was in python files. My modular code didn't get used heavily in the final approach, but was a significant chunk of my time. I also worked on running experiments in stage two for Mobilenet. I also worked heavily on understanding our entire code and experiment process in order to provide a paper that is accurate our process and clearly explains our approach to the grader. Also owned report writing and formatting, editing and revisions.
Harsh Chandak	Implementation, Analysis and Reporting	Pre-processed both the images and text data used in the model, which involved cleaning and formatting the data to ensure that it was in a usable format for the model; Implementation of the base model and running it on Inception and MobileNet to initiate testing; set up the complete notebook for the base model, which involved organizing and documenting the code and experiments to ensure that they were easily replicable and understandable for the rest of the team; Additionally, ran tests and tuned the model to optimize its performance; Also owned report writing and formatting
Patrick Pastore	Project Research and Scoping, Project Management, Implementation, Hyperparameter Tuning Experiments, Analysis, Report Writing	I took a lead role in scoping our project. I heavily researched in preparation for both our first project idea (sound recognition) and this project idea, helping us to pivot when the first idea's project proposal seemed to cause confusion. I took ownership of researching and understanding common image encoding techniques (CNNs, Visual Transformers, Object Detection). I modularized much of our code (with goal of easily-paremetrized experiments). I also worked to optimize our models through extensive hyperparameter tuning (running about 40% of experiments) and normalization work. I attempted to add an object detector to our image encoder. Lastly, I owned/wrote/edited a significant portion (40%) of the final report.
Vipul Koti	Project Management, Project Research and Scoping, Architectural Analysis, Hyperparameter Tuning Experiments Scoping, Successful Implementations (4 pre-train models, MultiCrossAtenntion, Top Bleu Score code from scratch and integration), 50+% experiments run, Analysis, and Report Writing.	I played a role with multiple hats (Project Owner, Hands-on Developer, Lead resource, Project Manager and as a Data Analyst), starting from the project scoping and requirement gathering, I read a lot of article and shared it with the team to get everyone excited abut Image captioning. On our base Keras tutorial, I did make the most significant successful integrations, First integration helped us to analyze how the "Imagenet" Pre-train models(VGG16, VGG19, Resnet50 and Resnet101) impacts the image captioning i.e. how well the image features are captured during image preprocessing and what is the impact of this on image captioning. Second, I worked on integrating different kind of attention on the Keras tutorial base model which had causal and cross attention. I tried three different kinds of attention, but I was successful in implementing and integrating MultiCrossattention. Third, the tutorial didn't have BLEU score implementation, so I implemented and integrated BLEU score which we run on test captions (1000 images) and we output average of top 20,50,100,200 and 400 BLEU scores. After these implementations, I worked on the optimization of our models through hyperparameter tuning. I ran more than 50% of the experiments our team did. After capturing these experiments, I worked on the analysis and captured the results on the reports. Finally, contributed (30%) of the final report writing.

Table 4. Contributions of team members.

APPENDIX -

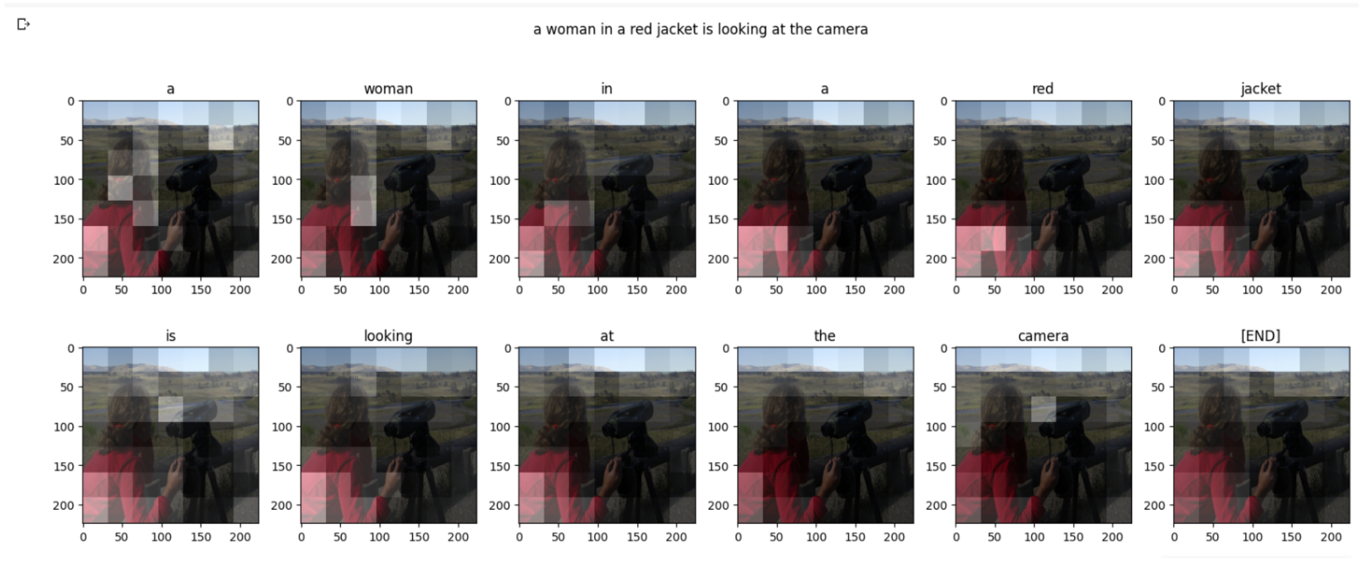

Figure 3. Generated Caption with Attention Map

Model	MobileNet		VGG16		VGG19		ResNet50		ResNet101	
Attention	Bleu 50	Bleu 400	Bleu 50	Bleu 400	Bleu 50	Bleu 400	Bleu 50	Bleu 400	Bleu 50	Bleu 400
Causal	9.472	9.472	0.000	0.000	8.095	8.095	0.000	0.000	0.000	0.000
Cross	30.811	15.695	32.189	14.727	30.552	12.418	30.178	11.355	30.109	12.797
MultiCross	28.259	14.438	31.139	16.950	31.548	16.833	30.817	14.485	30.972	12.399
Casual+cross	34.000	19.198	27.388	12.339	28.761	13.540	32.279	13.983	28.613	11.701
Causal+MultiCross	30.677	15.928	29.126	12.663	30.310	14.672	31.782	15.493	29.187	14.304

Table 5. Experiment 2- Evaluation metric BLEU score

Model	MobileNet		VGG16		VGG19		ResNet50		ResNet101	
Attention	Mask Acc	Val Mask Acc	Mask Acc	Val Mask Acc	Mask Acc	Val Mask Acc	Mask Acc	Val Mask Acc	Mask Acc	Val Mask Acc
Causal	0.434	0.367	0.414	0.362	0.413	0.373	0.418	0.379	0.410	0.364
Cross	0.431	0.326	0.406	0.318	0.416	0.315	0.457	0.330	0.467	0.337
MultiCross	0.400	0.326	0.462	0.347	0.411	0.353	0.475	0.339	0.463	0.360
Casual+cross	0.482	0.373	0.472	0.381	0.495	0.393	0.526	0.387	0.533	0.389
Causal+MultiCross	0.468	0.359	0.524	0.377	0.490	0.388	0.517	0.384	0.528	0.394

Table 6. Experiment 2- Evaluation metric Masked Accuracy