Project Report : CS 7643

Clivens LaGuerre
claguerre6@gatech.edu

Jerrod Pelley
jpelley3@gatech.edu

Kevin Ayers
kayers8@gatech.edu

Jared Benedict
jbenedict9@gatech.edu

Georgia Institute of Technology

Abstract

In this paper we attempt to demonstrate how research in the field of automatic image captioning has grown tremendously. We show this through the implementation and analysis of three seq2seq models: Merge Network, Encoder / Decoder with Attention, and the state-of-the-art OFA model. Quantitative and qualitative results confirm our hypothesis and show clearly just how far this area of study has come.

1. Introduction/Background/Motivation

Modern research in Artificial Intelligence is beginning to converge upon language models as a strong candidate for generalization across a wide variety of tasks. Research organizations like Google and OpenAI are beginning to see language models achieve impressive results across an assortment of A.I. applications even without task-specific training data [16]. The bedrock to many of these language models is the Sequence-to-sequence architecture (seq2seq).

In seq2seq models, inputs to the model are sequences of data and outputs from the model are also sequences of data. Often structured so that one component of the model encodes an input sequence to a vector of fixed dimensionality while another component decodes an unfixed target sequence of data output [11].

So what makes seq2seq models so special? We've seen in the past Deep Neural Networks (DNNs) have shown strong performance in the areas of language and speech [9]. Despite this strong performance, DNNs take as input a vector of fixed dimensionality and output a vector of fixed dimensions. A significant limitation, given that many general tasks require outputs expressed as sequences of unknown lengths. Seq2seq models overcome this limitation by using Recurrent Neural Networks (RNNs) to map temporal dependencies of input sequences to output sequences of unknown length [11].

Encoding a vector of fixed dimensionality to a decoded sequence of unknown length provides an expressiveness that a fixed output can not. This is why the fixed internal representation produced by the encoder can be a combination of many pre-trained neural network components allowing seq2seq models to generalize in some cases to even multi-modal tasks [25].

The success of seq2seq models has been the inspiration for this work; our goal is to achieve a deeper understanding of seq2seq models. We aim to highlight the important role Encoder / Decoder components have in contributing to the successful results of seq2seq architecture. This work could be relevant to any domain which requires AI models to possess a many-to-many relationship between model input and model output. Our aspirations include a detailed comparison of baseline seq2seq approaches, extension to include large pre-trained models, transformer based Attention, as well as a thorough comparison with state-of-the-art approaches. We include experience, report of problems, and pitfalls that occurred while accomplishing this research, either due to time or implementation details.

We review the performance of seq2seq models in solving the complex visual linguistic task of automatic image caption generation. The task of taking an input image and outputting a general description is a difficult one to solve. In addition to classifying types of objects located within the image, a learning model must provide a description of relations between objects in the image. Moreover this combined visual linguistic knowledge must be expressed in an English sentence of unknown length.

We analyse three increasingly performant versions of seq2seq models and their performance on the task of automatic image caption generation. A simple merging Encoder / Decoder seq2seq model and another Encoder / Decoder seq2seq model with Attention are compared to a more complex modern state-of-the-art, multi-modal, massively pre-trained seq2seq model. The first two Encoder / Decoder models are trained using the Flickr8k[12] data set consisting of eight thousand images each paired with five different captions. The captions provide clear descriptions of the entities and events occurring in their respective images. On the image caption generation task we obtain BLEU[18] score results for two models and compare these results to that of a recent state of the art multi-modal seq2seq model One For ALL (OFA) [25].

Our contributions are as follows. First, we review two common versions of the seq2seq model architectures. That is, the merging Encoder / Decoder and Attention based Encoder / Decoder. With this we highlight the importance pre-trained layers and Attention based components have on the final caption BLEU[18] score results. Finally, to further investigate how these seq2seq components have contributed to modern advances we compare these results to the state art seq2seq model OFA[25].

2. Approach

2.1. Data

Merge and Attention based seq2seq model training for the target automatic image caption generation task was completed using the Flickr8k image caption data set first introduced by Hodosh et al [10]. Flick8k is a manageable dataset of eight thousand images each having five text based English descriptions. All images were manually selected to contain a variety of scenes and situations. Every text description gives a summery of the objects and events taking place within the image. By associating each image with multiple descriptions, the Flickr8k dataset captures some of the variability in human generated photo captions. Moreover, the manageable size of the Flickr8k dataset allows all models to be tested using a single GPU during training.

2.2. Merge Network

When it comes to image caption generation, there are several different approaches that one could take. One of the most common choices is an Encoder / Decoder architecture. A typical Encoder / Decoder model uses a Recurrent Neural Network (RNN) to encode the input data that will later be decoded for the model's output. For image caption generation, the model is concerned with both linguistic information and image features. These two pieces of information, however, can be combined either through 'injecting' or 'merging' them within the model [15]. Injecting the image data into the model simply means that the image data will be passed through the RNN alongside the linguistic data to be encoded together. While this will work effectively, it will result in a more complex RNN whose hidden state size will be very large [15]. An alternative is to encode the image data separately and then combine it with the encoded text data from the RNN. Encoding the image and linguistic data separately allows the RNN to only have to process the text data while another pre-trained model extracts image features in a separate step. The encoded data will later be combined together through either concatenation, multiplication, or addition [2]. The combined encoded data can then be processed by a simple decoder. This effectively produces a model that given a set of image features and a starting sequence can predict each next-word in the output caption sequence until a full caption is derived.

2.2.1 Architecture and Implementation

The Merge Network model architecture consists of two separate models that each encode their own respective sets of input data that is later combined and passed through a decoder as seen in Fig. 2. One of these models is an RNN that consists of an embedding layer following by an LSTM layer. It takes in the linguistic data as input. The other is a CNN that takes the image data as input. While RNNs are typically thought of as being generative, the RNN in this case is purely focusing on encoding word embeddings and never actually sees the image features [22]. The images are passed through a CNN to extract the individual features from within. The encoded output from these two models are then merged together before passing through a decoder.

When it comes to combining the image features with the output from the RNN, adding has proven to be the most effective [22]. It is in this 'merge' layer that the generation process happens. Once the two encoded sets of data are merged together, they are then passed through a series of linear layers in order to get a probability distribution for the available words in the defined vocabulary. This distribution is passed through a SoftMax layer that will result in the final prediction for the next word in the sequence.

To facilitate controlled experiments and analysis, the Merge Network was implemented using the open source machine learning platform TensorFlow alongside the Keras deep learning framework [6]. The Merge Network implementation was directly based upon the original merge-model described by Mark Tanti et el. in their paper [22]. Further implementation guidance was received from Jason Brownlee and the Machine Learning Mastery Blog [2] along with his Keras guide [4]. Merge Network testing included the large pre-trained Keras models VGG16, InceptionV3, and ResNet152V2 for visual components. Textual components of the model were tested using Keras word embedding layers trained directly on the captions from the Flickr8K[12] data set and compared with large pre-trained word embeddings, GLOVE, created by Jeffery Pennington and the Stanford Computer Science department [19].

For our experimentation, there were a few optimizations that were made to increase the performance of the model and lower the time needed to train. The first optimization was in preparing the photo data. In our implementation, a pre-trained model was utilized for identifying the content within the image and generating the corresponding photo "features". To do this, the last layer of the pre-trained model was stripped away leaving behind the raw weights right before its classification layer. The Flickr8K [12] images were then resized to match the corresponding pre-trained model and then passed through, and their features were captured. These features were stored within a pickle file and referenced later in the training phase. While this step could have been implemented as simply another step within the overall model architecture, by performing this step once prior to training we were able to remove the redundant step and speed up train time for our model.

The second optimization was for the Flickr8K [12] dataset's text descriptions. This time we focused on cleaning up the text rather than extracting information. For this, each text description was processed to ensure that all letters were lowercase, all punctuation was removed, all words with containing only one letter or less were removed, and all words containing numbers were removed. These final descriptions were then stored in a text file and referenced later in the training phase. This helps reduce the size of the target vocabulary which will, in turn, reduce the memory size required to train our model along with speeding up the training process. Essentially, this reduces the size and complexity of the model.

In order to execute our experiments for our Merge Network implementation in an efficient and timely manner, we utilized Google Colab alongside our local GPU.

For more information on the training process see B.0.1 in the appendix.

2.3. Encoder / Decoder with Attention

Improving upon the elementary Merge network is the Seq2Seq Encoder / Decoder Network model with Attention. In this structure, the responsibility for image captioning is designated to two main components; the encoder and decoder. The encoder processes and embeds raw images using a pre-trained CNN architecture while the decoder bares the responsibility to describe the output of the encoder with the linguistic data using the proposed salient features from the Attention Network through a RNN. This model architecture evolved from models like the Merge Network by modularizing responsibilities into separate networks components and leveraging the power of the Attention Mechanism. Attention is a mechanism which began gaining prominence in the year 2000, and has since been widely used to improve the performance of language translation and image captioning [17]. It has proven to help many different models achieve SOTA performances as evident in the "Show, Attend and Tell" paper [27]. This portion of this paper is intended to demonstrate how the Seq2Seq model improved with adapting proven mechanisms such as Attention.

2.3.1 Architecture and Implementation

The Encoder / Decoder model is a two part network architecture with each part being their own separate networks. The encoder network is largely a pretrained CNN which takes images as raw inputs and encodes them into smaller data representations. An Average Pool (Adaptive) layer is added to the end of the CNN to normalize the output of the encoder network. The output of the encoder is then fed into the decoder network.

The Decoder Network is the main component which specifically encapsulates the image-captioning downstream task. In addition to the Attention Network and embedding layers for linguistic processing, an LSTM layer is included for propagating relevant information as the process moves further in the sequence. The input to the Decoder Network is the encoded captions and the length for each caption. Within the decoder, the Attention Network is invoked and weights are learned and used during next word predictions [23].

After each step of the decoding process Beam Search is used to determine the best sequence of words that describes the image; we experimented with varying values for k to gain a better understanding of Beam Search.

To explore, experiment, and evaluate the effects of the Attention mechanism in a Seq2Seq model architecture we employed the work demonstrated in the "Show, Attend and Tell" paper [27]. We endeavored to implement and customize the model architecture presented in the paper to perform similar experiments as performed in the Merge Network model for comparison analysis. Our implementation guidance was received from Sagar Vinodababu in his image captioning tutorial [23]. The testing and experimentation for the Encoder / Decoder model also included the same large pre-trained CNN models, however, these are provided through the Pytorch Model Zoo[1]. We primarily used Google Colab for our efforts to explore the Encoder / Decoder model with Attention. The resource used for the Encoder / Decoder implementation is the open-source code provided in [23], however, we customized it to achieve our experimentation goals.

As a normalization technique, the Flicker8k [12] dataset was split into predefined sets (train, validation, test) using the Andrej Karpathy's splits configurations [13]. Using these splits, the images and captions are transformed into their data representation using a $word\_map$ for the index of the captioning words within the corpus. We limited five captions for each image. For each image and caption, the data is normalized into two json formatted files and one HDF5 formatted file for the captions, caption lengths, and the images respectively.

The encoder component is customized to support the three pre-trained CNN architectures. A passed-in parameter instructs the model of which CNN to execute. Each CNN architecture was inspected for their expected inputs and output dimensions for proper connections between layers and blocks. Since the last few layers of each CNN architecture are used for classification, they were removed to retain the raw image receptive fields. An additional layer, Average Pool (Adaptive), is appended to the network for resizing the output of the encoder network [23].

The output of the encoder network, the encoded image, is passed into the decoder component along with the encoded captions and lengths. A distinction from the Merge Network is that the Encoder / Decoder with Attention model is not using a pre-trained word-embedding, however, the option is open to do so. In the decoder component, all weights are initialized uniformly. During the decoding sequential process, at each time step, the Attention Network is used to provide a weighted encoding of the image. This weighted image encoding is concatenated with the word embeddings and both are decoded by the LSTM network. Dropout is used to regularize the output from the LSTM network before receiving the next word prediction from the final fully connected layer. Upon the completion of the entire sequence, extra paddings are removed and CrossEntropyLoss is used as the loss function. All the weights in the decoder component are adjusted through backpropagation using the Adam Optimizer; these weights include word embedding, Attention Network, layers to support the LSTM network, and the output layer.

The version of the Attention mechanism adopted is the *soft* Attention. It may provide higher weights for more than one region of the image, however, all the weights sum up to one [23]. The output for training, validation and evaluation are measured using BLEU scoring metric (BLEU-1, BLEU-2, BLEU-3 and BLEU-4).

For more information on the training process see B.0.2 in the appendix.

2.4. OFA

For our culminating approach, we selected the OFA (One For All) architecture [25]. The OFA architecture is a multi-modal, pre-trained model which achieves state-of-the-art performance on the COCO Captions dataset [5] for Image Captioning tasks.

OFA is a pre-trained sequence to sequence transformer model which has learned on a variety of pre-training tasks, such as visual grounding, image-text matching, and others. The heart of OFA is that it has been trained under different modalities, including Vision, Language, and Vision + Language. The multi-modal pre-training of the OFA model allows for the architecture to have great generalizability; including to unseen tasks and tasks which combine Vision and Language. The multi-modal nature of OFA also prevents the need for additional task specific, learnable modules.

2.4.1 Architecture and Implementation

The OFA architecture uses the Transformer as the underlying architecture mechanism; although the transformer took its foothold in the Natural Language Processing domain, it has also recently been shown that can perform very well in the Computer Vision domain [8]. The architecture consists of multiple transformer Encoder / Decoder layers, with other mechanisms that have been used with transformers prior (normalization and self-attention). This transformer architecture is jointly pre-trained on a large combined image and text dataset [25]. Image patches extracted from the original image, as well as object detection information are combined with the corresponding language information in order to embed the image and textual information into a combined feature space.

The open-source OFA implementation [26] is used for our experiments. We used the *OFA-Large* pre-trained checkpoint, (approximately 470 million parameters) with the fine-tuning checkpoint for image captioning on COCO. Additional custom code was written to generate inference results such as BLEU scores [18] on our test set (which is common to all three of our experimenting architectures), as well as for generating captions with custom images.

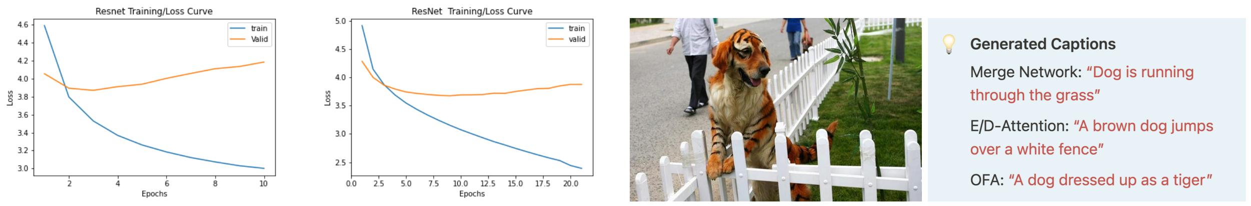
3. Experiments and Results

We performed an extensive set of experiments to assess the effectiveness of Merge, Attention, and OFA models on the task of automatic photo caption generation. Each model was tested using several metrics including qualitative and quantitative techniques. In addition, multiple pre-trained visual and linguistic architecture components were tested in combination with Merge and Attention based seq2seq models to assess the effects of large scale, pre-trained components have on task performance.

3.1. Scoring and Metrics

For quantitative analysis we use the Bilingual Evaluation Understudy metric (BLEU)[18] score across all three models for evaluation of the performance of each architecture. We use 1-gram, 2-gram, 3-gram, and 4-gram metrics, each with uniform weighting. This allows us to measure performance consistently and accurately across all three models.

To gain a qualitative understanding of performance for the Merge, Attention and OFA models, a human evaluation metric was collected for each model. To collect this metric, 13 custom images were selected. Generated captions from each model resulting from the randomly selected images were used during a human graded scoring process. Four



Generated Captions

Merge Network: "Dog is running through the grass"

E/D-Attention: "A brown dog jumps over a white fence"

OFA: "A dog dressed up as a tiger"

Figure 1: Merge and Attention Networks Loss (Left), Caption Comparison (Right)

Model	Merge-VGG	Merge-InceptionV3	Merge-ResNet	Attention-VGG	Attention-InceptionV3	Attention-ResNet	OFA
BLEU1	0.5294	0.5191	0.5264	0.6401	0.5843	0.6400	0.5978
BLEU2	0.2871	0.2715	0.2840	0.4591	0.3954	0.4605	0.4107
BLEU3	0.1948	0.1832	0.1964	0.3606	0.2985	0.3601	0.3097
BLEU4	**0.0846**	**0.0827**	**0.0827**	**0.2245**	**0.1729**	**0.2202**	**0.1765**

Table 1: BLEU Scores - Merge Network, E/D-Attention , OFA

caption graders were asked to score each generated caption with a rating of 1-5. Final average score per model was collected for thorough qualitative analysis and can be seen in Fig. 5.

3.2. Experiment Details

To assess differences in large scale, pre-trained visual components Merge and Attention based networks were trained in combination with VGG, ResNet, and Inception V3. BLEU and Qualitative scores were collected for all model comparisons and compared with the state-of-the-art OFA model. Additionally, the effects of large, pre-trained word embeddings were investigated on the more elementary Merge Network. Pre-trained word embeddings showed no performance improvements. However, training speeds increased by a factor of 2 when using pre-trained linguistic layers.

3.3. Results and Analysis

We report our main results on Flickr8k in Fig. 1 and Table 2. Results show the Attention model using the pre-trained VGG component is a able to achieve a BLEU-4 score of 0.22 on par with current state-of-the-art approaches for the Flick8k dataset [7]. Moreover, this is similar to average human BLEU-4 scores of .22 collected on other large datasets [24].

Human performance level BLEU-4 scores do not tell a complete picture. Some captions produced by the Attention model show clear errors that humans would most likely not make. This provides some indication that improvements can be made over the BLEU score metric. Moreover, qualitative human ratings show the transformer based OFA model produces higher quality image captions for Flickr8k test images (see Fig. 5). The average OFA score is 4.92, average Encoder/Decoder with Attention score is 2.35, and average Merge score is 1.5; this clearly reflects our hypothesized score progression over the increasingly performant model architectures.

Results in Fig. 1 left show the learning curve characteristics during training for Merge and Attention models. Training details provided here show a clear over fitting that is occurring for both models. The training set size of Flickr8k allows these deeper models to easily overfit the training data. Interestingly adding Attention to the Encoder / Decoder architecture reduces overfitting by small amounts even on this smaller training data size. Regardless, both models fit the training data in a small number of training epochs taking an average time of 30 minutes on a single GPU. Larger training data sizes would most likely help both models generalize more and reduce overfitting.

Fig. 1 right compares a difficult photo caption, this shows the impressive performance provided by transformer based Attention. Both simple Merge, and Attention based models have a difficult time generating context of the image properly and incorrectly describe the dog in the picture [21]. Only the OFA model is able to properly describe context by noting the dog is dressed up as a tiger. This underscores the strength of adding transformer based self-attention to the seq2seq architecture.

Collectively the results in these experiments show a clear progression from an elementary Merge Encoder / Decoder to an Attention based Encoder / Decoder and finally a Transformer based Encoder / Decoder. This highlights the expressiveness of the seq2seq architecture. Pre-trained components can be used with Encoder/Decoder modules to facilitate complex artificial intelligence tasks. Furthermore, adding Attention to seq2seq architecture improves model performance by more than a factor of two. This shows some limitation of the recurrent architecture. Without Attention, information found early on in large sequences is lost thus degrading model performance. Adding Attention allows the model to retain more information. This effect is increased with the more advanced transformer based Attention of the SOTA OFA model.

4. Experience

4.1. Pitfalls and Challenges

As with any project, we expected to run into challenges throughout our experimentation. One of the main challenges that we faced was finding a state-of-the-art (SOTA) model that was tailored to image caption generation tasks. We originally planned on using the ViLBERT[14] model however we were unable to find an implementation that we could successfully get working. We then faced challenges trying to find an alternative model. While there are many SOTA models, few of them appeared to be specifically for image caption generation. We finally settled on using OFA[26] for this part of our experimentation.

Another issue that we ran into was with the implementation of the Merge Network model. We first attempted to recreate Jason Brownlee's Keras implementation[4] using Pytorch. We were able to create a model that did run and appear to train successfully, however, when we tested the quality of its generated captions, they were very poor. After thorough testing and debugging of our model, we were not able to discern the cause of this. Due to time constraints, we then pivoted our approach to adapting the Keras implementation directly as opposed to converting it to Pytorch.

Code execution times also proved to be a major issue for various parts of our project. For example, without special configuration, TensorFlow would not work with our GPUs. After significant amount of trial and error, we were able to successfully get our model training on our local GPU. In the meantime, we were able to get around this by training our models on Google Colab. However, we had to change our data loading step so that it did not attempt to load all of the data into memory at one time because it would cause the Colab session to break due to running out of allocated memory.

Regarding the Encoder / Decoder with Attention, We did anticipate that it may not work as described, therefore assigned two team members to collaborate and tried to get it working. We were cautiously optimistic since the code presented some similarity to what we have been exposed to from the lesson of the CS7643 class, we were somewhat certain that we would be able to get it to run. We discovered that, due to some library requirements, it wouldn't work on Microsoft Windows OS. After resolving some other bugs, we finally were able to get it to run from a Linux environment as well as Google Colab.

Finally, another challenge we experienced related to the BLEU scores we achieved between the three models. We expected that the BLEU scores would grow increasingly from Merge $\rightarrow$ Encoder / Decoder $\rightarrow$ OFA, but what we found were resulting scores of Merge $\rightarrow$ OFA $\rightarrow$ Encoder / Decoder (see Table 2). We also found that the captions generated by the OFA model were qualitatively the best of all three models, especially when testing with samples outside our dataset wholly.

We hypothesize that the source of this problem is either BLEU scores (and N-gram metrics in general) are not sophisticated enough compared to more modern evaluation metrics [20]; or it is possible that there is a source of data leakage which is causing increased performance on our test set within the Encoder / Decoder model. Further analysis and experimentation could reveal definitively the source of this issue.

4.2. Changes in Approach

Compared to our original scope for this project, we did not have to change our planned approach very much. The most drastic change was with the SOTA model that we were planning on using for comparison. Due to the issues mentioned in the above section, we had to shift our focus from the ViLBERT[14] model to the OFA[25] model. This allowed us to move forward with our experimentation while not compromising on the quality of our SOTA model nor the scope of our analysis. Additionally, we had originally planned on developing the Merge Network model and the Encoder / Decoder with Attention models ourselves. However, due to time constraints and overall complexity of the models themselves, we opted to use more out-of-the-box solutions that we were then able to tailor to our specific needs

4.3. Project Success

Despite the few changes in our approach due to unforeseen complications, we were still able to deliver exactly what we proposed for this project. We successfully created, trained, and experimented with a Merge Network model, Encoder / Decoder with Attention model, and the SOTA OFA[25] multi-modal, pre-trained model. We were also able to experiment with various optimizations and improvements to these models as well. The overall goal of our project was to compare these models to show how image caption generation has improved over time. Through our experimentation and analysis of BLEU[18] scores and the overall quality of the output captions, we were able to not only quantify this improvement but also qualify our original hypothesis.

5. Work Division

Summary of contributions are provided in Table 3.

References

[1] Pytorch - model zoo. 3

[2] Jason Brownlee. Caption generation with the inject and merge encoder-decoder models, Aug 2019. 2

[3] Jason Brownlee. How to prepare a photo caption dataset for training a deep learning model, Aug 2019. 8

[4] Jason Brownlee. How to develop a deep learning photo caption generator from scratch, Dec 2020. 2, 6

[5] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server, 2015. 4

[6] Francois Chollet et al. Keras, 2015. 2

[7] Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge J. Belongie. Learning to evaluate image captioning. *CoRR*, abs/1806.06422, 2018. 5

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. 4

[9] Li Deng George E. Dahl, Dong Yu and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012. 1

[10] Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *J. Artif. Intell. Res.*, 47:853–899, 2013. 2

[11] Oriol Vinyals Ilya Sutskever and Quoc V. Le. Sequence to sequence learning with neural networks. *NeurlIPS*, 2014. 1

[12] Aditya Jain. Flickr 8k dataset, 2020. 2, 3

[13] Shravan Himanshu777 Kumar. Karpathy splits for image captioning, 2021. 3

[14] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks, 2019. 6

[15] Kenneth P. Marc Tanti, Albert Gatt. Where to put the image in an image caption generator. *Natural Language Engineering*, 24(3):467–489, apr 2018. 2

[16] Sharan Narang and Aakanksha Chowdhery. Pathways language model (palm): Scaling to 540 billion parameters for breakthrough performance, 2022. https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html. 1

[17] BM Nechu. What is attention mechanism?, 2020. 3

[18] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation, 2002. 2, 4, 6

[19] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014. 2

[20] Vikash Sehwag and Qasim Nadeem. Towards image captioning and evaluation. 6

[21] SALON ART STAFF. The week in pictures: June 5-11, 2010, 2010. 5

[22] Marc Tanti, Albert Gatt, and Kenneth P. Camilleri. What is the role of recurrent neural networks (rnns) in an image caption generator?, 2017. 2

[23] Sagar Vinodababu, Mario kmario23, Jason Ren, and ngshya. Show, attend, and tell — a pytorch tutorial to image captioning, 2020. 3, 4, 8

[24] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014. 5

[25] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. *arXiv preprint arXiv:2202.03052*, 2022. 1, 2, 4, 6

[26] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework code, 2022. 4, 6

[27] Jimmy Kiros Ryan Cho Kyunghyun Courville Aaron Salakhutdinov Ruslan Zemel Richard Bengio Yoshua Xu, Kelvin Ba. Show, attend and tell: Neural image caption generation with visual attention, 2016. 3

A. Model Architectures

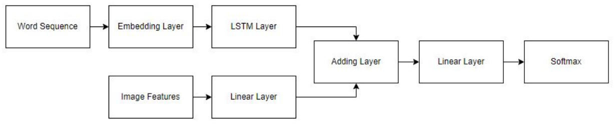


Figure 2: Merge Architecture

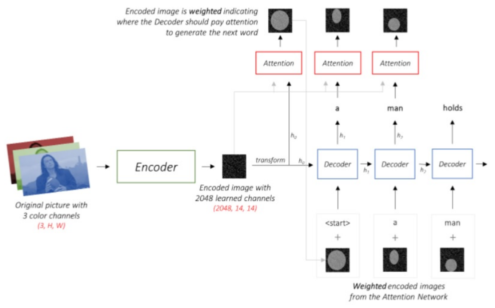

Encoded image is **weighted** indicating where the Decoder should pay attention to generate the next word
Attention
Attention
Attention
a
man
holds
Encoder
transform
h_0
h_0
h_1
h_2
h_1
h_2
Decoder
Decoder
Decoder
Original picture with 3 color channels
(3, H, W)
Encoded image with 2048 learned channels
(2048, 14, 14)
<start>
a
man
Weighted encoded images from the Attention Network

Figure 3: E / D with Attention Architecture, extracted from [23]

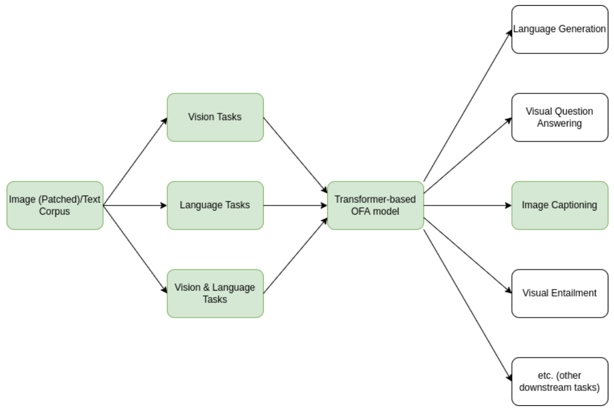


Image (Patched)/Text Corpus
Vision Tasks
Language Tasks
Vision & Language Tasks
Transformer-based OFA model
Language Generation
Visual Question Answering
Image Captioning
Visual Entailment
etc. (other downstream tasks)

Figure 4: OFA Architecture

B. Training

B.0.1 Merge Network

In order to train our Merge Network model, we had to first load in the saved image feature data along with the cleaned image descriptions. This data was compiled and split into our training and validation datasets. Keys that map each input image to its corresponding data are generated using the input image's file name. Before the descriptions can be passed into the model for training, they first must be converted to an integer representation by encoding the words within the vocabulary. This was done by using a tokenizer to learn the mapping between words and their corresponding integer keys.

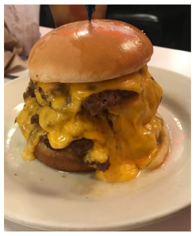
This model works by predicting captions one word at a time. This means that given an input image and its currently predicted caption, the model will keep predicting the next word of the sentence until it thinks that the caption is complete. Because of this, we must train the model in the same manner. In order to achieve that, each description is broken into its individual words that are then encoded using the before-mentioned tokenizer. These are used to create an array of sequences for each image – description pair. These sequences are then compiled together and used for generating the training and validation datasets.

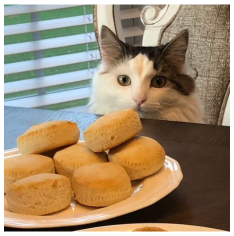
Since we are dealing with large amounts of data, training is very process-intensive and requires large amounts of memory to store all the image features and corresponding descriptions. In order to help alleviate this issue, we implemented a mechanism called Progressive Loading[3]. This concept utilizes a data generator that will yield a smaller batch of data each time it is called that will eventually iterate through the entire dataset. You can think of this as generating all the data for a given set of images as opposed to loading all the data for all the images. This requires significantly less memory since we are no longer loading all the information at one time, rather small chucks as needed.

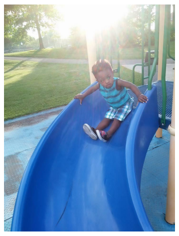
B.0.2 Encoder / Decoder Attention

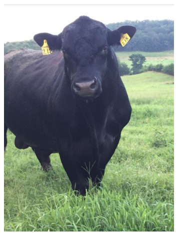
In order to train our Encoder / Decoder with Attention model, we used similar techniques used in the Merge Network for data normalization but with the Karpathy's splits configurations to derive the train, validation, and test sets. The training process is set up to execute for a maximum of 120 epochs with early stopping. The early stopping is triggered after the performance of 10 subsequent epochs diverge from the best using their BLEU-4 score. The decoder learning rate is initialized at 4e-4 and starts to decay after eight epochs with unimproved performance. During training, the best and last checkpoints of the learned parameters are stored for evaluation and recovery. Loss curves are observed for debugging and tuning adjustments, see Figures 1.

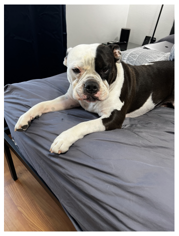
C. Example Captions

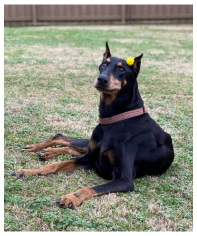

(a)
Merge: 1.0
E/D w/ Attn: 1.0
OFA: 5.0

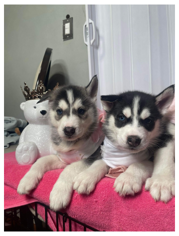

(b)
Merge: 1.0
E/D w/ Attn: 1.25
OFA: 5.0

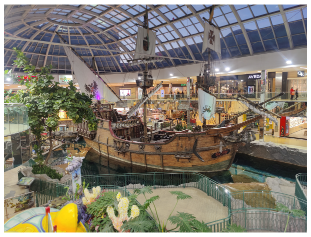

(c)
Merge: 2.0
E/D w/ Attn: 5.0
OFA: 5.0

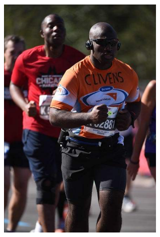

(d)
Merge: 1.75
E/D w/ Attn: 2.25
OFA: 5.0

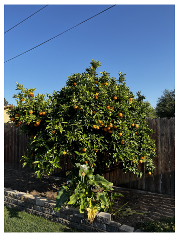

(e)
Merge: 2.0
E/D w/ Attn: 3.75
OFA: 5.0

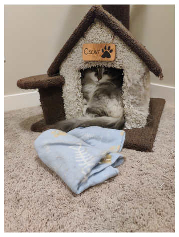

(f)
Merge: 2.25
E/D w/ Attn: 3.0
OFA: 5.0

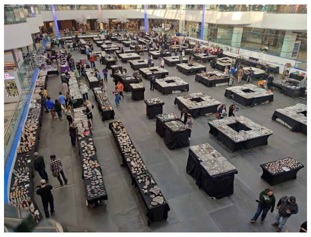

(g)
Merge: 2.0
E/D w/ Attn: 3.0
OFA: 4.75


(h)
Merge: 1.0
E/D w/ Attn: 1.25
OFA: 4.75


(i)
Merge: 2.5
E/D w/ Attn: 4.0
OFA: 5.0


(j)
Merge: 1.0
E/D w/ Attn: 1.0
OFA: 5.0


(k)
Merge: 1.0
E/D w/ Attn: 2.0
OFA: 5.0


(l)
Merge: 1.0
E/D w/ Attn: 2.0
OFA: 5.0

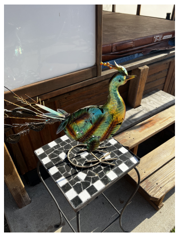

(m)
Merge: 1.0
E/D w/ Attn: 1.0
OFA: 4.5

Figure 5: Qualitative Assessment of Captions (Averaged Across Four Reviewers)

Picture	Merge	Attention	OFA
(a)	two people are walking on the beach	a brown dog is eating a <unk>	a cheeseburger with cheese on a white plate
(b)	two girls are playing on trampoline	a small brown and white dog is laying on the ground with its mouth open	a cat sitting at a table looking at a plate of biscuits
(c)	two children are playing in the grass	a young boy sliding down a blue slide	a little boy is going down a blue slide
(d)	dog is running through the grass	a black dog runs through a grassy field	a black cow standing in a field of grass
(e)	dog is running through the grass	a black and white dog with a black collar is laying on its hind legs	a black and white dog laying on a bed
(f)	two dogs are running through the grass	a black and brown dog is running through the grass	a dog sitting on the grass with a flower in its head
(g)	dog is running through the grass	two dogs are playing with a toy in a backyard	two puppies sitting on a table next to a stuffed animal
(h)	man in red shirt is standing on the street	a person is standing in front of a large building	a large boat is displayed in a display case in a mall
(i)	man is black shirt is standing on the sidewalk	a group of men play in a race	a man in an orange shirt is running a race
(j)	man in red shirt is standing on the beach	a man in a yellow shirt is walking on a sidewalk	an orange tree with oranges on it next to a fence
(k)	man in black shirt is standing in front of crowd	a small dog is laying in the sand	a cat sleeping in a cat house next to a towel
(l)	man in red shirt is standing on the street	a group of people are standing in front of a crowd	an aerial view of a large room filled with tables with vendors and people walking around
(m)	man in red shirt is standing on the street	a girl in a green shirt is sitting on a skateboard	a statue of a peacock sitting on a bench

Table 2: Generated Captions

Student Name	Contributed Aspects	Details
Clivens LaGuerre	Implementation, Analysis, and Writing	Implemented and trained the Encoder / Decoder with Attention model. Performed qualitative and quantitative analysis. Contributed to writing paper.
Jerrod Pelley	Implementation, Analysis, and Writing	Implemented and trained the Encoder / Decoder with Attention model. Setup OFA model. Performed qualitative and quantitative analysis. Contributed to writing paper.
Kevin Ayers	Implementation, Analysis, and Writing	Implemented and trained the Merge Encoder / Decoder model. Performed qualitative and quantitative analysis. Contributed to writing paper.
Jared Benedict	Implementation, Analysis, and Writing	Implemented and trained the Merge Encoder / Decoder model. Performed qualitative and quantitative analysis. Contributed to writing paper.

Table 3: Contributions of team members.