Interpreting Image Caption Generation with Attention

Olivier Filion
ofilion@andrew.cmu.edu

Michael Agaby
magaby@andrew.cmu.edu

Nicholas Amano
namano@andrew.cmu.edu

Abstract

The recent successes of deep learning methods for computer vision and natural language processing tasks have garnered a lot of attention. Automated image captioning exists at the intersection of both fields. This project implements a CNN-LSTM architecture with a visual attention mechanism which achieves a 23.9 BLEU-4 score on the MS COCO dataset using curriculum learning. We interpret the CNN encoder through visualizations and show that the model is able to automatically learn to look at the most important elements of the image when generating captions.

1 Introduction

Image caption generation is the action of generating a textual description of a given image. It is a challenging problem because it requires the model to be able to identify the different objects in an image as well as the relationships between them, making it more complex than simple object recognition tasks. In addition, the model must also be able to express these relationships in human understandable language, free of grammatical errors. An important feature of human intelligence is our ability to understand the world around us and to clearly and succinctly communicate this understanding to others. Caption generation is therefore a very difficult problem, but one that has significant potential benefits.

In 2014, there was an average of 1.8 billion photos uploaded to the web daily (Eveleth, 2015). Automatic image captioning therefore has the potential to substantially reduce the cost of labelling those images. It can also be used in AI-assisted data labelling for machine learning datasets. Additionally, automatic image captioning also has the potential to assist the visually impaired.

Recent approaches in caption generation have typically consisted of using an encoder to generate a down-sized representation of the image which is then fed into a decoder, typically consisting of a neural network type architecture. These types of approaches have been aided by the increase in computing power which has made it easier to train very deep neural networks. Attention mechanism have also been employed for image captioning tasks with great success. We employ a similar approach in this project, with an emphasis on model explainability. In particular, we apply several visualization methods to VGG-16 in order to help increase our model's explainability.

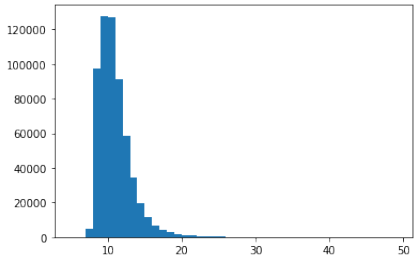
2 Data

We are training and evaluating our model on the 2017 version of the Microsoft Common Objects in Context (MS COCO) (Chen et al., 2015) dataset. The dataset contains 118,288 training images, 5000 validation images, and 40,670 testing images. For the training set, each image is accompanied by an average of 5.002 human generated captions. The mean caption length is 10.48 words and the median is 10 with the shortest caption being 5 words long and the longest being 49 words long; seen in Figure 1b. Similarly to what is done by Soh (2016), we replace uncommon words, words appearing less than 20 times, with the <UNK> token. We also add the <START> and <END> tokens to the captions which indicate the start and end of the caption, respectively. A <NULL> buffer token

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

is used to standardize the length of captions being trained and predicted. Some of the most common non-conjunction words are shown in Figure 1a. Here we can already see some andocentric bias, with man being the most common word. Additionally, we see the types of actions that are most common in our dataset.

(a) Wordcloud of most common non-conjuction words. More common words are larger



(b) Histogram of the caption length of the training dataset.

Figure 1: Visualizations of dataset

The MS COCO dataset is particularly suited for image captioning because of the quantity of images, their variety, and the number of captions. Unlike the ImageNet dataset (Deng et al., 2009) which is commonly used for classification tasks, images in the MS COCO dataset contain more complex subjects, enabling better captions. Additionally, 5 caption per image help prevent captioners from unintentionally biasing the dataset. Unfortunately, the MS COCO dataset is not fully immune to bias, particularly a strong gender bias.

i) A **man** doing tricks on a skateboard at a skate park.
ii) A **man** jumping a skateboard over a ramp.
iii) A skateboarder performing a stunt near graffiti covered concrete.
iv) The **man** is riding his skateboard practicing his tricks outside.
v) a person is doing a trick on a skateboard

Figure 2: Image and captions with gender bias from MS COCO dataset [Zhao et al. (2017)]

Explored thoroughly by Zhao et al. (2017) and Bhargava and Forsyth (2019), there exists a male centric bias within the MS COCO dataset. There are two major reasons for this, inaccuracy or stereotyping within the annotation process, and an unbalanced dataset. Within the MS COCO dataset, there are captions that misidentify the gender of human subjects. Either the human annotator accidentally misclassified the subject or a stereotypical bias played a roll, an example of this can be seen in Figure 2. Apart from the annotation process, the dataset is unbalanced in gender for particular activities. This leads models trained on MS COCO to attribute certain activities with a certain gender. Additionally, a recent paper by Northcutt et al. (2021) finds pervasive errors in 10 of the most popular vision datasets including ImageNet (Deng et al., 2009). Although MS COCO is not a part of the study, it is reasonable to assume that such errors exist within the COCO dataset as well.

3 Related work

An early approach for image caption generation was proposed by Farhadi et al. (2010). Their model assumes a space of *Meanings* between the spaces of *Sentences* and *Images*, which they represent using a triplet of <*object,action,scene*>. More recent approaches have instead centered around using CNN's to encode image representations and neural networks as decoders to generate the captions. The first such approach was by Kiros et al. (2014), who used feed-forward neural networks to decode joint image-sentence embeddings from a CNN and LSTM. They were also able to show that the learned embedding space has semantic meaning and respects vector space arithmetic (e.g. *image of a blue car* - "blue" + "red" gives similar results as the vector produced by *image of a red car*).

This approach was followed by Mao et al. (2014), who replace the feed-forward neural network with a LSTM. Vinyals et al. (2014) follow a similar approach but the model sees the input image only at the beginning, while Mao et al. (2014) show the LSTM the image at each step of the caption generation. Soh (2016) was able to show with a similar method that semantically similar words (e.g. "plate" or "bowl") move the hidden states of the LSTM in the same direction.

Other approaches have incorporated attention mechanisms, a technique that allows for the enhancement of important parts of the data, into their models. Xu et al. (2015) validate the use of attention in image caption generation with state-of-the-art performance on multiple benchmark datasets. They also validate their use of attention by showing through visualizations that show that the model is able to learn to focus on the important parts of the image when generating the corresponding output word.Tensorflow (2021) provides a good overview of an encoder-decoder model with attention for image caption generation. Huang et al. (2019) extend the conventional attention mechanism through attention on attention.

A common element in a lot of the mentioned papers is the use of a transfer learning with a pre-trained CNN for the encoder portion. In this paper, we use both VGG16 (Simonyan and Zisserman, 2014) and GoogleNet (Szegedy et al., 2015) pre-trained on the ImageNet Deng et al. (2009) dataset. VGG was able to outperform previous generations of CNN's in particular by replacing large kernel-sized filters by successive smaller 3x3 kernel-sized filters. GoogleNet was able to alleviate some of the over-fitting problems of previous architectures by becoming wider rather than deeper with the introduction of Inception layers, which also makes the network a lot more sparse.

Recent trends in machine learning have pushed for more methods to allow humans to easily understand the results of machine learning algorithms, often called explainable artificial intelligence (XAI). In particular, deep neural networks are often seen as "black box models". Mordvintsev et al. (2017) use activation maximization to visualize the features that the CNN detects at a given layer in an attempt to better understand the model.

4 Methods

4.1 Image caption generation

All the models that we tried in this project share a common architecture. First, there is an encoder based on the pretrained VGG16 CNN (Simonyan and Zisserman, 2014). This encoder takes as input RGB images $\mathbf{x} \in \mathbb{R}^{224\times224\times3}$ and transforms them into an input for the decoder. Then, the decoder outputs a sequence of word scores $\mathbf{p}_i \in \mathbb{R}^V$ where V is the vocabulary size and $i = 1, \ldots, S$ where S is the maximum caption length. The vocabulary is built as described in section 2, which yields a vocabulary size of $V = 5193$. The predicted word is then $\hat{\mathbf{y}}_i = \operatorname{argmax} \mathbf{p}_i$ and the label is a one-hot encoded vector denoted $\mathbf{y}_i$. To speed up inference, we set the maximum caption length to $S = 18$ and we only use a random subset of 30% of the training set for training. Using a subset of the training set is a common technique used to train complex models when training time is limited. For example, Tensorflow (2021) uses a similar idea in their implementation of an image captioning model with attention.

4.1.1 Baseline model

For our baseline model, we used the encoder-decoder architecture proposed by Soh (2016).

The encoder takes the vector output by VGG16's dense layer, which is of dimension 4096, and passes it to a trained linear layer with no activation to obtain the context vector $\mathbf{z} \in \mathbb{R}^{256}$. Importantly, we do not modify any of the pretrained weights from VGG16 during training.

The decoder is built using a long short-term memory recurrent neural network (LSTM). At the first step of the LSTM, the context vector $\mathbf{z}$ is passed as input and the hidden state $\mathbf{h}_0 \in \mathbb{R}^{256}$ and the cell state $\mathbf{c}_0 \in \mathbb{R}^{256}$ are initialized to zero vectors. Then, for each subsequent step k, the input is $E\hat{\mathbf{y}}_{k-1}$ where $E \in \mathbb{R}^{256\times V}$ is a trained embedding matrix. At each step k, we calculate the output scores using the log-softmax of $L_o\mathbf{h}_k$ where $L_o \in \mathbb{R}^{V\times256}$. Note that the context vector is only passed into the LSTM at the beginning as in Vinyals et al. (2014)

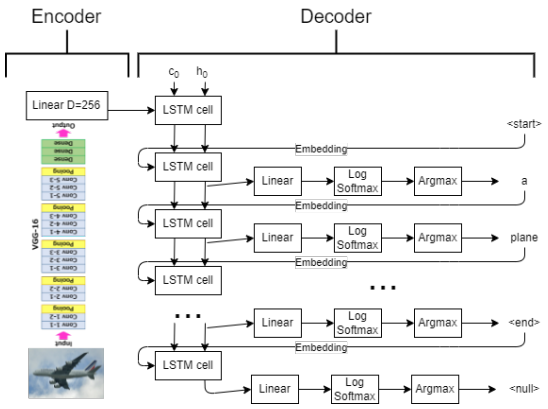


Figure 3: Diagram for the baseline model. VGG-16 image from Hassan (2018).

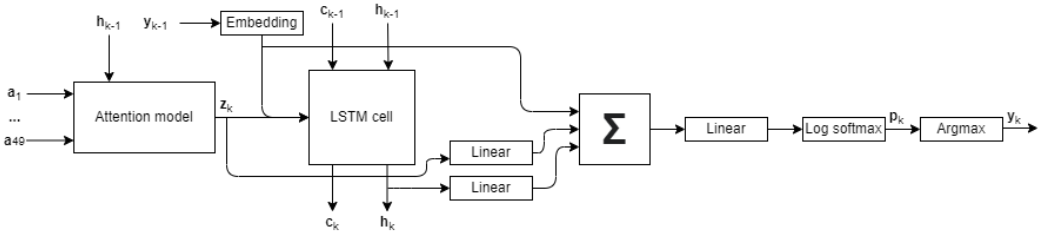


Figure 4: Diagram of a single step of the attention decoder.

For training, we use a technique called teacher forcing. In order to make training faster and more stable, we give as input to each step k of the LSTM the embedding of the label of the step $k-1$ instead of what the decoder actually predicted, i.e. we use as input $E\mathbf{y}_{k-1}$ instead of $E\hat{\mathbf{y}}_{k-1}$.

We train the weights of the model by optimizing the cross-entropy loss function as implemented in PyTorch (Paszke et al., 2019).

$$L(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{\exp\sum_{v=1}^{V} \mathbf{p}_{i,v}^{(n)}} \sum_{v=1}^{V} \mathbf{y}_{i,v}^{(n)} \exp \mathbf{p}_{i,v}^{(n)} \tag{1}$$

We use PyTorch's (Paszke et al., 2019) implementation of the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 and a batch size of 64. The Adam optimizer uses momentum and an estimation of the second order gradients to speed up convergence. Momentum refers to the idea that, if the gradient goes in the same direction for two consecutive steps, we want it to go farther in that direction and, if it goes in opposite direction, we want it to slow down. We train the model for 20 epochs. We also add a dropout layer (Srivastava et al., 2014) after each linear layer with a dropout probability of 25% to help prevent overfitting.

4.1.2 Adding attention

We extend the baseline model defined above by implementing a soft attention mechanism as described in Xu et al. (2015). The idea behind attention is to make the model learn to automatically look at different parts of the input image for each word it generates, just as a human would do if writing a caption. For the encoder, the only difference is that instead of using the output of the pooling layer of VGG16, we use the output of its last convolutional layer with dimension $7 \times 7 \times 512$, which we flatten and pass into a linear layer with no activation to obtain 49 annotation vectors $\mathbf{a}_j \in \mathbb{R}^{256}$. For the decoder, we now initialize the hidden state and cell state of the LSTM using a non-linear transformation of the average annotation vector.

$$\mathbf{h}_0 = \tanh\left(L_{\mathrm{h}}\left(\frac{1}{49}\sum_{j=1}^{49}\mathbf{a}_j\right)\right) \qquad \mathbf{c}_0 = \tanh\left(L_{\mathrm{c}}\left(\frac{1}{49}\sum_{j=1}^{49}\mathbf{a}_j\right)\right) \tag{2}$$

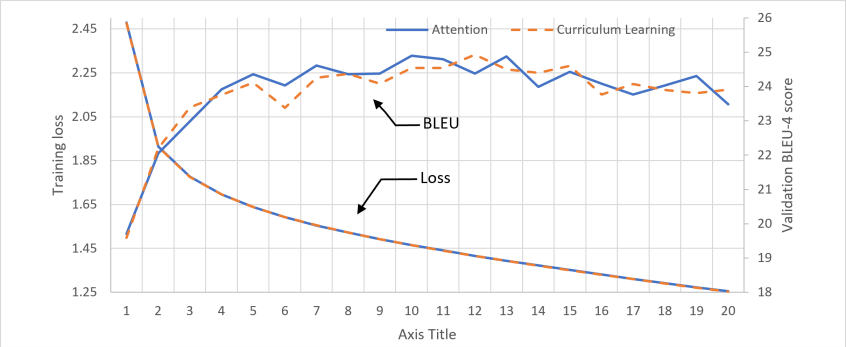


Figure 5: Progression of the training loss and validation BLEU-4 score

At each LSTM step k, we want to calculate a context vector $\mathbf{z}_k$. We first need to compute weights α_j for each annotation vector. This is done by taking the softmax of the $\tilde{\alpha}_j$'s calculated as:

$$\tilde{\alpha}_j = L_{\text{att}}(\tanh(L_{\text{ann}}\mathbf{a}_j + L_{\text{hid}}\mathbf{h}_{k-1})) \tag{3}$$

Then, the context vector is the weighted average of the annotation vectors.

$$\mathbf{z}_k = \sum_{j=1}^{49} \alpha_j \mathbf{a}_j \tag{4}$$

Once we have the context vector, we append to it the embedded output from the previous step and pass it to the LSTM cell. We can then calculate the scores for each word of the vocabulary.

$$\mathbf{p}_k = L_{\text{out}}(E\hat{\mathbf{y}}_{k-1} + L_{\text{oh}}\mathbf{h}_k + L_{\text{oc}}\mathbf{z}_k) \tag{5}$$

The training procedure is the same as the baseline model, but we use a learning rate of 0.0001.

4.1.3 Curriculum learning

Teacher forcing, as described above, makes a significant difference in the quality of sequence generation tasks. However, it can also have some issues. Lamb et al. (2016) note that teacher forcing can lead to small error compounding during inference when we don't give the model the ground truth labels anymore. To help counteract this, we trained the model using curriculum learning (Bengio et al., 2009). With curriculum learning, we progressively make the task more difficult for the model during training. In our context, that means using the previous label as output with a probability β and using the previous output with probability $1 - \beta$ for some $\beta \in [0, 1]$. During the first five epochs, $\beta = 1$. Then, at every 5 epochs, we reduce β by 0.1, resulting in a teacher forcing probability of 70% at the 20th epoch.

4.2 Interpreting the black box

4.2.1 Intermediate activations

When an image is encoded by the CNN, it passes through all layers and the model outputs a feature vector. Instead of extracting the feature representation at the end, we can instead extract any layer's output. These are known as intermediate activations. The brighter spots represent image features. We are only interested in the intermediate activations corresponding to convolution layers, since pooling layers are just downsampled versions. Intermediate activations can give us some insight about our encoder for a specific sample, making them a local model interpretation technique.

4.2.2 Activation maximization

Activation maximization uses gradient ascent to maximize the activation function for a specific filter by modifying the input image. This technique starts with an image of random noise, and iteratively constructs an image that maximizes the activation function at that particular filter. We use Pytorch's (Paszke et al., 2019) implementation of Adam (Kingma and Ba, 2014) optimizer with a learning rate of 0.05 and weight decay of 0.000001 for 30 iterations to produce our visualizations. Activation maximization is a global interpretation method, since it is not specific to a single sample.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Baseline (VGG16)	56.7	37.0	23.5	15.6
Baseline (GoogleNet)	65.1	46.8	32.3	22.2
Attention (VGG16)	65.7	47.6	33.4	23.5
Curriculum (VGG16)	**66.7**	**48.4**	**34.1**	**23.9**
Soh (2016)	N/A	N/A	N/A	24.4
Xu et al. (2015)	71.8	50.4	35.7	25.0

Figure 6: Experimental results compared to previous papers

4.2.3 DeepDream

DeepDream (Mordvintsev et al., 2017) applies activation maximization to modify a specific input image with the goal of detecting the features learnt by a specific filter. Our implementation of DeepDream uses the same optimizer and hyperparameters as our activation maximization.

4.2.4 Saliency maps

Saliency maps are a technique first proposed by Simonyan et al. (2013) to measure the spatial support for a specific class within a given image in the context of image classification using CNN's. A saliency map tries to find pixels in the image that when perturbed the least, produce the largest change in the output by examining the gradients. Given an image I and class c, define $S_c(I)$ to be the class score computed by the model. We can approximate the score with a linear function in a small neighbourhood of an image I_0 with a Taylor expansion:

$$S_c(I) \approx w^T I + b \tag{6}$$

where w is the derivative of the score S_c with respect to the input at the image I_0. Thus given an image I_0, the saliency map can be computed as follows:

Perform a forward pass of I_0 through the network, and obtain w during the back-propagation. For gray scale image, since w will have as many elements as I has pixels, we can simply compute the saliency map M by $M_{i,j} = |w_{\text{index}(i,j)}|$ where index(i,j) refers to the index of the element of w corresponding to the pixel $I_{i,j}$. For colored images, we instead take the max over the 3 channels: $M_{i,j} = \max_c |w_{\text{index}(i,j,c)}|$. This method is often referred to as a "vanilla" saliency map.

5 Results

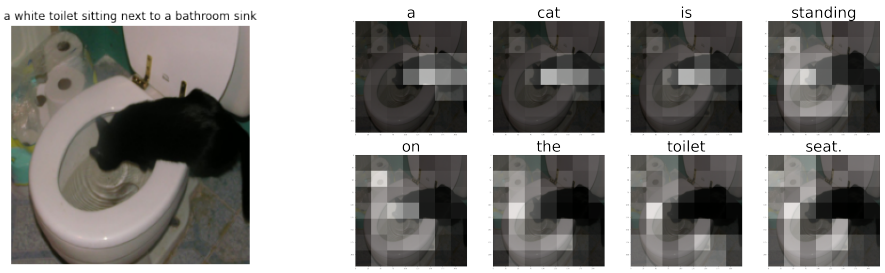
We use the Bilingual Evaluation Understudy (BLEU) score, introduced by Papineni et al. (2002) for evaluating our caption models. The BLEU score requires a size of n-gram to use when comparing sentence similarity, this is denoted BLEU-n where n is the size of the n-gram and usually is a value between 1 and 4. BLEU score uses the following:

$$\text{count}_{\text{clip}}(\text{n-gram}) = \min(\text{count(n-gram)}, \text{maxrefcount(n-gram)}) \tag{7}$$

Where the count is the number of times the n-gram occurs in the candidate caption and maxrefcounts is the maximum count of the n-gram that occurs a single reference caption. Finally, we add a brevity penalty (BP) of 1 if the candidate caption is longer than the reference caption and $e^{(1-r/c)}$ otherwise.

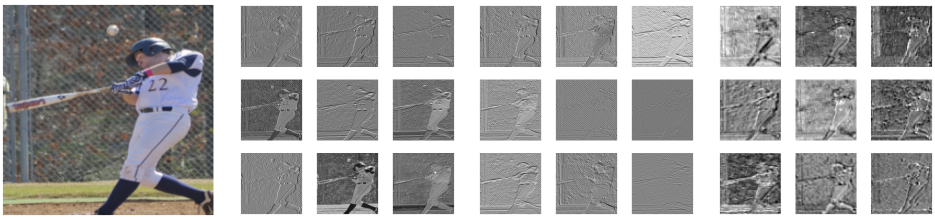
$$p_n = \frac{\sum_{\text{n-gram} \in \text{candidate}} \text{count}_{\text{clip}}(\text{n-gram})}{\sum_{\text{n-gram(n-gram)} \in \text{sentence}} \text{count(n-gram)}} \qquad \text{BLEU-N} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} \frac{1}{n} \log p_n\right) \tag{8}$$

The BLEU scores for the trained models are summarized in figure 6. We see that the curriculum learning model outperforms the others, which aligns with our expectations. We also test our baseline architecture with a different pretrained CNN, GoogleNet (Szegedy et al., 2015). We see that the baseline performs significantly better with GoogleNet. Due to time constraints, we were not able to train the attention model with GoogleNet. The curriculum learning model almost matches the performance of the two papers we looked at in this project, despite using only 30% of the training set and a weaker pretrained encoder (both papers use GoogleNet). Also, note that the Xu et al. (2015) paper uses early stopping based on BLEU score. Our results are presented without early stopping but our curriculum learning model reaches a BLEU-4 score of 24.92 at epoch 12, as seen in figure 5.



(a) Baseline model (VGG16) (b) Curriculum learning with the attention weights

Figure 7: Example prediction for the baseline model and curriculum learning model



(a) Original image (b) Activation at layer 1 (c) Activation at layer 3 (d) Activation at layer 11

Figure 8: Intermediate activations at 9 different filters for a sample image

6 Discussion and analysis

6.1 Analyzing results

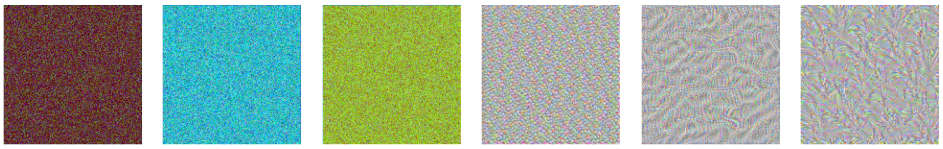
In line with other papers, we show that the CNN-LSTM model with attention was well suited for the caption generation problem, achieving similar results to Xu et al. (2015) with significantly less data and a less powerful encoder. Additionally, the use of attention in our model significantly improves performance over the baseline as shown in the table 6. In Figure 7 we see an example of a caption that is vastly improved by the addition of attention to the model. By visualizing the attention weights, we see that the model has now learned to automatically shift its gaze to the different elements in the image, while the baseline model has difficulty focusing on more than one component, getting stuck on the "bathroom" theme.

6.2 Interpreting the model

In this project, we leverage several visualization methods to better understand our model. Figure 8 presents some of the intermediate activations for 9 different filters at different convolutional layers for an example image. We notice that by layer 3, the CNN is encoding mostly the important elements of the image, notably the silhouette of the baseball player as well as the texture of the fence background. By layer 11, the image has been significantly down-sampled having gone through multiple pooling layers, but we can still clearly see the outline of the player as well as the texture of the background.

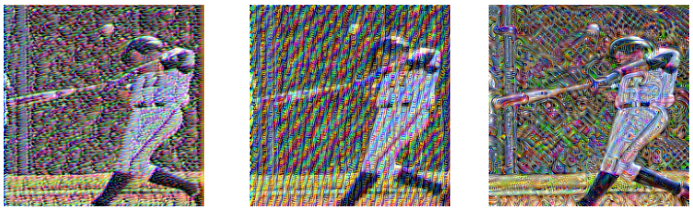
As mentioned, activation maximization can allow us to visualize specific filters in a CNN. In Figure 9 we see that the earlier filters of the network are more focused on picking up particular colors, while at the later convolutional layers, the network is more and more focused on complex patterns and textures. The combinations of filters that extract color, pattern, and texture information allow the CNN to encode images using only their important features.

In Figure 10. we see the effects of activation maximization on a particular image. We note that the different filters are "looking for" different pattern and textures in the image. We observe that this particular filter at layer 28 (Figure 10 (c)) is extracting complex swirl patterns from images.



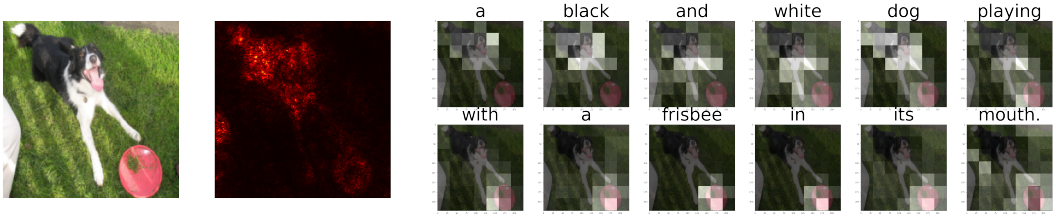
(a) Layer 1 (b) Layer 1 (c) Layer 1 (d) Layer 12 (e) Layer 19 (f) Layer 24

Figure 9: Activation maximization for several fiters at different levels



(a) Layer 7 (b) Layer 12 (c) Layer 28

Figure 10: DeepDream visualizations for a filter at different layers for a sample image



(a) Saliency map (b) Attention plot

Figure 11: Attention and saliency plots on same image with caption

Figure 11a shows the saliency map for an image of a dog. We clearly see that the CNN is able to easily detect the dog in the image. Figure 11b shows the predicted caption of the model for the same image and also shows where the model is directing its attention when predicting each word. We see that when predicting the words "black" and "white" that the model is focused on the parts of the dog corresponding to those colors. The model also directs its attention to the frisbee when predicting it. Interestingly, when predicting the word dog, the model's attention is directed towards exactly the pixels that the saliency map predicts as being the most important. This provides further justification for the use of attention in our model.

6.3 Limitations and future directions

As mentioned, time and computation limitations were a factor in this project. As such, the use of VGG16 over GoogleNet for our attention models as well as training on only 30% of the data affected the quality of our results. We hope that using the entire dataset with GoogleNet encoder along with our attention implementation would yield better BLEU score than Xu et al. (2015). In addition, our model with curriculum learning slightly outperforms our regular attention model. We believe that more aggressive curriculum learning (e.g. lowering the probability of teacher forcing more quickly) might yield better results on the test set. We note in 5 that the model with curriculum learning has a very similar loss progression as the model without, which leads us to believe that we can increase the difficulty of the learning task through a more aggressive curriculum.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Shruti Bhargava and David Forsyth. 2019. Exposing and correcting the gender bias in image captioning datasets and models. *arXiv preprint arXiv:1912.00578*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Rose Eveleth. 2015. How many photographs of you are out there in the world? *The Atlantic*. [Online; accessed November 24, 2021].

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. *Computer Vision, ECCV 2010 - 11th European Conference on Computer Vision, Proceedings (PART 4 ed., pp. 15-29). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 6314 LNCS, No. PART 4)*.

Muneeb Hassan. 2018. Vgg16 - convolutional network for classification and detection. [Online; accessed December 6, 2021].

Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. 2019. Attention on attention for image captioning. *arXiv:1908.06954*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Ruslan Salakhudinov, and Rich Zemel. 2014. Multimodal neural language models. *Proceedings of the 31st International Conference on Machine Learning, PMLR 32(2):595-603*.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29.

Jonhua Mao, Wei Xu, Yi Yang, Jiang Wang, Huang Zhiheng, and Yuille Alan. 2014. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv:1412.6632*.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. 2017. Inceptionism: Going deeper into neural networks. *Google AI Blog*.

Curtis G. Northcutt, Anish Athalye, and John Mueller. 2021. Pervasive label errors in test sets destabilize machine learning benchmarks. *rXiv:2103.14749v4*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv:1312.6034*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Moses Soh. 2016. Learning cnn-lstm architectures for image caption generation. *Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Tensorflow. 2021. Image captioning with visual attention. [Online; accessed November 17, 2021].

Oriol Vinyals, Alexander Toshev, Sammy Bengio, and Dumitru Erhan. 2014. Show and tell: A neural image caption generator. *arXiv:1411.4555*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *arXiv preprint arXiv:1707.09457*.