Neural Network Architectures for Image Captioning

Emily Kern

June 20, 2018

Abstract

Image captioning is a popular field of research in the real of neural networks, as it combines two neural network problems into a single challenge: object recognition in images and sentence generation. Each subproblem can be dealt with using a convolutional neural network (CNN) and a recurrent or long short-term memory neural network (RNN/LSTM). Given an extensive dataset of raw images and crowd-sourced captions courtesy of Flickr8k and Flickr30k, we set up a CNN and RNN/LSTM in an encoder-decoder architecture. The CNN takes the raw image and extracts higher-level features. The RNN/LSTM receives the output of the CNN and, based on the training captions in the dataset, adjusts its weights so that when a new input image is presented, the neural network can generate an accurate caption that describes the image. Training the neural network on both datasets yielded syntactically correct caption predictions, but the captions were not descriptive of their respective images. This could be a result of word bias that accrued during the training process, causing the weights in th neural network to put extra favor on certain words.

Contents

List of Figures

List of Tables

1 Introduction

Images are an amalgamation of colors and objects in a binding frame. They provide insight into a captured instance in time and are subject to human interpretation. Each interpretation is a result of the human brain processing the image and drawing connections between the image and various elements stored in memory, such as similar looking images or descriptive words. All of these connections and resultant conclusions happen incredibly quickly. The process of image interpretation is a constant in everyday life. For instance, when drivers see a red octagonal sign, they know to come to a stop. But drivers were not born knowing what a stop sign is. They learned its meaning over time, learning by example through childhood and formal training during driver's educational courses. This process of learning by example and training is reflected in neural networks, computing systems inspired by biological neural networks.

Due to their ability to learn by example, neural networks have become popular for various object recognition tasks and output generation. Object recognition is achieved via training with example objects, which are fed into the neural network as an image. Output generation can pertain to many things, but output in the form of sentences have become popular in the neural networks field. Object recognition and sentence generation pair to make a new topic: image captioning. Image captioning requires the combined efforts of object recognition and sentence generation. Given an image, the neural network must first recognize what is in that image. Based off of its conclusion, the neural network must then be able to generate a sentence, which requires training in natural language syntax. There are many implementations of object recognizing neural networks and sentence generating (or, at least, word generating) neural networks that are quite successful ([8], [17], [3], [9]), but marrying the two types of networks remains relatively open-ended.

Vinyals et al. [16] proposed the encoder-decoder architecture as a way to approach the image captioning problem, a proposal that has been implemented in a variety of ways (more on this in Sections 2 and 3). In order to fully implement the proposed architecture, there must be a dataset that is extensive and is tailored to the image captioning challenge; such data exist in the forms of Flickr8k([4]) and Flickr30k([12]). The full semantics of Flickr8k and Flickr30k will be discussed in Section 4.2.

Using the Flickr datasets and an encoder-decoder architecture, we aim to see how successful a neural network can be in the field of image captioning. Given an image, how well can a neural network generate an appropriate caption? Using the Karpathy and Fei-Fei [6] model (which will be discussed in greater detail in 2.1), which follows the general Vinyals et al. [16] proposal and supports Flickr8k and Flickr30k, we will explore and evaluate the performance of the neural network image captioning. Based on the results of the Karpathy and Fei-Fei [6] model (5), we can give a human evaluation of the results and propose further improvements to the system so as to strengthen the captions' relevance to their respective images.

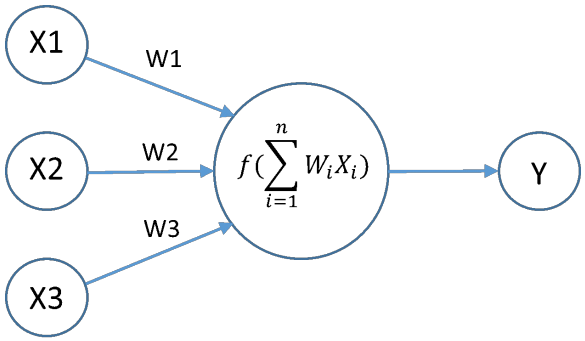


Figure 1: A diagram of a simple neural network.

2 Neural Networks

Before delving further into the specifics of the Karpathy and Fei-Fei [6] model, there should be a general understanding of how neural networks work. Neural networks are modeled after human cognition; they can learn by example and thus become "smarter". They are a machine learning technique that is used in artificial intelligence. Originally, neural networks were used mostly for classifying objects, such as recognizing numbers in photographs (Goodfellow et al. [3]). The neural network performs object classification by taking input, vectorizing it, and multiplying the vector values by certain weights and appropriate biases to classify the input properly. Figure 1 gives a simplified diagram of a neural network where x_1, x_2, and x_3 are inputs multiplied by their respective weights w_i for $1 \leq i \leq 3$. The aforementioned weights can either be hard-coded or learned automatically with example data (often referred to as the training set). Over time, several groups have expanded the field of neural networks beyond object classification. Now, there are several different kinds of neural networks that can be used for a variety of purposes, and those purposes have the chance to expand as research groups figure out ways to combine different neural network problems.

2.0.1 Biological Neural Networks

Before delving into Convolutional and Recurrent Neural Networks, let us first go over the biological inspiration behind neural networks. Neural networks were inspired by human cognition. Humans are born knowing *very* little, but when presented with new information over and over again, and with the guidance of a teacher of sorts, humans are capable of *learning*. Biological Neural Networks are, as the name implies, a network of neurons. Neurons are made up 3 types of components: dendrite, soma, and axon (Figure 2).

Dendrite. Dendrites are extensions of the neuron that receive signals from other neurons. Such signals are electric pulses that are transmitted neuron to neuron across the synaptic gaps via chemical processes.

Soma. Also known as cell body, soma sums the incoming signals. When there is enough input, the soma

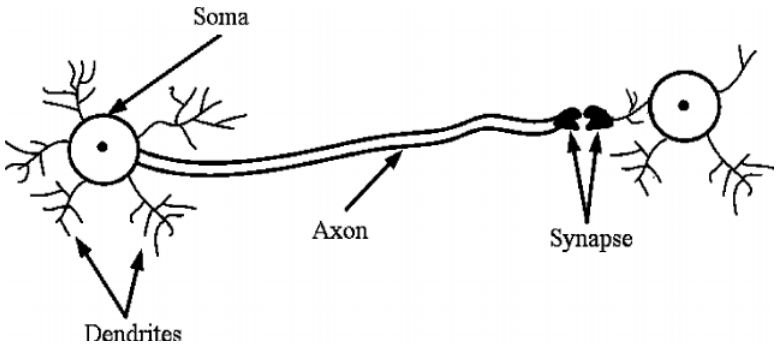


Figure 2: A diagram of a biological neuron showing the dendrites, soma, and axons.

fires, sending the signal over its axon to other soma. Soma are in a state of either firing or not firing, so we can view soma's functionality as binary.

Axon. Axons are responsible for transmitting signals between neurons. They are extensions of neurons that reach other neurons in a neural network.

2.0.2 Neurons

Neurons in the computer science neural network field are strikingly similar to the neurons in biological neural networks. Also known as nodes, individual neurons work with a summation of inputs and generate output based on that input. A generic input formula for a neuron Y would look like:

$$y_{input} = w_1x_1 + w_2x_2 + ... + w_nx_n$$

Here, Y receives input, denoted as x_i, from neurons $X_1, X_2, ..., X_n$ for $1 \leq i \leq n$. w_i is the weight attributed to the connection between X_i and Y. Y executes its activation level, which is a function of the net inputs it received, and output is sent in the form of a signal to other neurons.

$$y_{output} = f(y_{input})$$

2.0.3 Architecture

In the context of neural networks, architecture refers to the pattern of connections between neurons. This encompasses not only the way neurons are connected, but also the weights that are attributed to each neuron connection. Different architectures make for different types of neural networks that can specialize in solving certain problems. Convolutional and recurrent neural networks, which will be discussed further in Sections 2.2 and 2.3 respectively, are such examples of varied architectures.

Neural networks have layers, where each layer has a set of neurons that act in the same manner. The behavior of these neurons are determined by their activation functions and the patterns of their respective

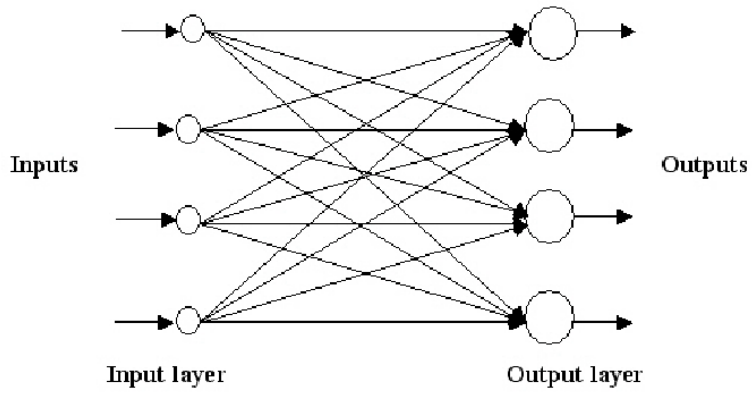


Figure 3: A simple single-layer neural network. Single-layer networks do not have any hidden layers.

connections and weights. Neurons in the same layer have similar behavior because they share the same activation function and have the same pattern of connections to other neurons.

Most neural networks have, for each layer, either complete interconnectivity or zero interconnectivity with other layers. Networks can be categorized as either single or multi-layered. This classification is determined by the number of layers the network possesses, not including the input layer, as the input layer does not perform any computations. Layers of neural networks can be categorized as follows:

- **Input:** The first layer of a neural network. Its sole purpose is to receive input from a source (usually external); it does not perform any computations on input data. Thus, it is not counted when determining a neural network is single or multi-layered.
- **Output:** The final layer of a neural network.
- **Hidden:** A layer that is neither an input nor output layer.

All neural networks have an input and output layer. A single-layer network has just the input and output layers (Figure 3). A multi-layered network has the input and output layers along with at least one hidden layer (Figure 4).

2.1 Encoder-Decoder Architecture

Going forward, we will discuss how multiple neural networks can be combined to make a more complex architecture capable of more complex problems. In particular, we are interested in the encoder-decoder architecture. This type of architecture is common in scenarios where we want to not only classify data, but also produce output. Since we want a neural network that can take an image input and generate natural language output, the encoder-decoder architecture is desirable.

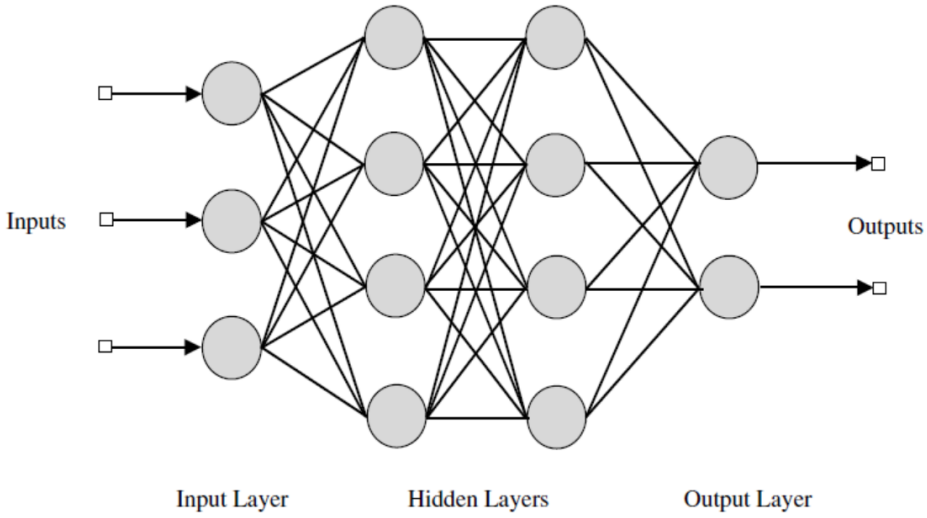


Figure 4: An example of a multi-layer neural network with multiple hidden layers.

The architecture breaks down to an encoder and decoder. The encoder is a neural network that produces a vector representation of the input called an embedding. Embeddings are created in such a way that inputs that share similarities will end up with similar embeddings. The decoder is another neural network that takes the embedding and generates output based on that embedding. In image captioning, this means that an input image is made into an embedding, and that embedding is fed into the decoder to generate a caption based on the embedding.

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are popular in the field of neural networks and are especially useful for image recognition and classification problems. The CNN takes an input, usually an image, and extracts information, usually higher-level features, from said image. The image is processed as a vectorization of data, which allows the neural network to be able to work with it. The vectorization is a matrix of pixel values (0 to 255, representing values for RGB) and is fed into the CNN as input. Using a filter, which is like a smaller matrix that has built-in weights and biases, we slide over the input image and perform calculations on the image. The calculations can identify different key attributes based on its weights and biases to classify the image input. CNN's architecture is comprised of convolutional and pooling layers to accomplish its tasks.

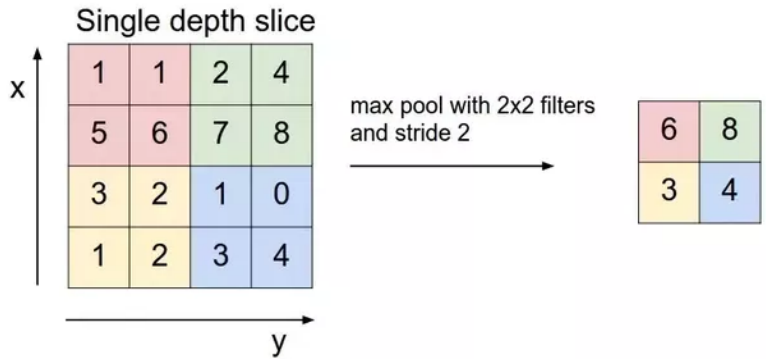


Figure 5: An example of how max pooling works. Each subregion is represented by a different color. The 2x2 measurement refers to the size of the subregion.

2.2.1 Convolutional Layer

The convolutional layer applies a specified number of convolutional filters to an image. For each subregion of the image, the convolutional layer performs mathematical functions on the subregion, ultimately producing a single value for the output. The output of a CNN in the image processing and classification field of interest is usually a feature map, so the single value output from the convolutional layer is added to the output feature map.

2.2.2 Pooling Layer

The pooling layer takes image data extracted by the convolutional layer as input. It aims to reduce dimensionality of feature. That is, the pooling layer aims to decrease the processing time when given an input image's resultant feature map. The pooling layer possesses a **max pooling** function, which partitions the input image from the convolutional layer into a set of non-overlapping rectangles, for this purpose. For each subregion (non-overlapping rectangle), a max function is applied to each value in the subregion. The result is a pool of the maximum values of each subregion. Figure 5 visualizes how max pooling works.

2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are able to run on sequences of input or output, so they are much less rigid than CNNs by nature. The RNN's most notable feature is its ability to calculate a new state (output) based on current input and the previous state (old output one time step before the current input), as demonstrated in Figure 6. Written as a recurrence formula, we get:

$$h_t = f_W(h_{t-1}, x_t)$$

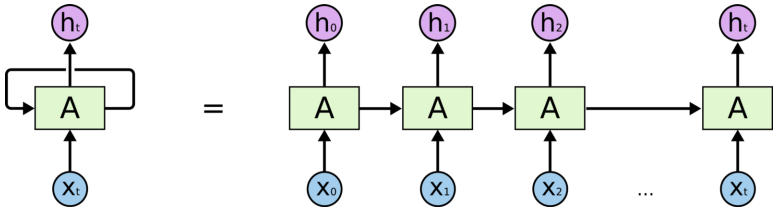


Figure 6: The structure to the left of the equals sign is what a recurrent neural network looks like. The structure to the right is a representation of a recurrent neural network if it were "unrolled".

where h_t represents the new state (or output of the RNN), f_W is a function with built-in weights and biases, h_{t-1} is the previous state, and x_t is the current input. A sequence of inputs are individually run on this same formula to create a sequence of outputs, were the output of some arbitrary input is dependent on previous output. This formula is encompassed in a single "cell" that is looped over.

The steps of a basic RNN are as follows:

1. A single time step of the input x_t is supplied to the network
2. Calculate the current state h_t using a x_t and previous state h_{t-1}
3. Current h_t becomes h_{t-1} — will be deemed the previous time step in the next loop through the RNN
4. Go as many time steps as the problem at hand requires; combine information from all previous states
5. Once all time steps are done, use final current state to calculate output

The calculated output is compared to the actual output (a given evaluation metric). An error is calculated from this comparison. An intuitive way to view the calculated error is how "off" the calculated output was from the actual output. The error is used to modify the weights in the RNN, training the RNN so that, ideally, by the final time step, the RNN's training can better the RNN's output.

Due to their dependence on previous time steps' output, RNNs are popular when dealing with data that come in a sequence. In particular, RNNs can be extremely useful when dealing with natural language. A sentence is a sequence of words in grammatical order, and words are a sequence of letters. Weights and biases, which are calculated along with a gradient — a metric used in weight computation — can help the RNN learn how natural language sequences work, making the RNN popular when natural language is a part of a neural network problem. The RNN's methods of processing input make it ideal for processing sequences of input data to perform both input sequence classification and output sequence generation.

Despite working well with sequential inputs, basic RNNs struggle with fundamental pitfalls, particularly the exploding and vanishing problems. The exploding gradient problem concerns gradients becoming so large due to a one or many gradient values becoming extremely high; the vanishing gradient problem

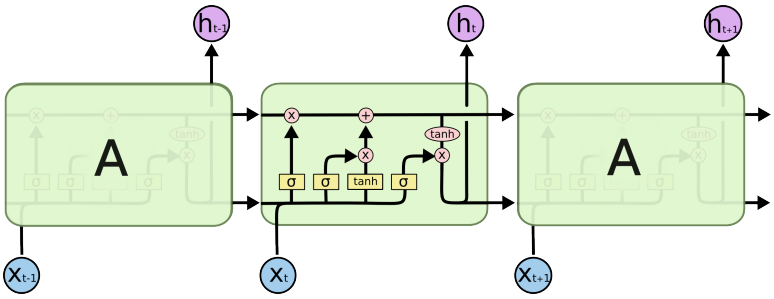


Figure 7: The inner workings of a LSTM. A single "cell" consists of four layers or "gates" that simulate longterm memory.

concerns gradients that have diminished over the course of a longer sequence. A basic RNN is not good at remembering information from several time steps ago, so an earlier gradient (recall that the gradient helps train weights in the RNN) may have been "forgotten", causing the RNN to fail to recall some relevant piece of data from the past. An intuitive way to think of the vanishing gradient problem is that a basic RNN has short-term memory loss. Alternative types of RNNs have been developed to deal with these gradient problems.

2.3.1 Long Short-Term Memory (LSTM)

A long short-term memory recurrent neural network (LSTM/LSTMRNN) patches the RNN's short-term memory loss issue by including additional gate functions that simulate longterm memory. The gates represent three components: input, forget, and output.

Figure 7 offers a visualization of the steps a LSTM takes to complete training:

1. Given the previous cell state h_{t-1} and its computed output, the LSTM determines what information gained from the previous state is worthwhile keeping; using a sigmoid layer to calculate whether some piece of information should be forgotten or not, where the computation n is between 0 and 1 (0 means forget everything; 1 means remember everything)

2. Decide what new information is to be stored in the cell state; a sigmoid layer acting as the input gate layer decides which values ought to be updated (training the LSTM); a tanh layer creates a new vector consisting of new candidate values that could potentially be added to the current cell state

3. Update the old cell state h_{t-1} to new cell state h_t

4. Calculate the output based on h_t and one more sigmoid layer which essentially decides what to output; feed h_t through the tanh function and multiply that value by the output of the sigmoid gate; the output consists of only the parts we decided to keep

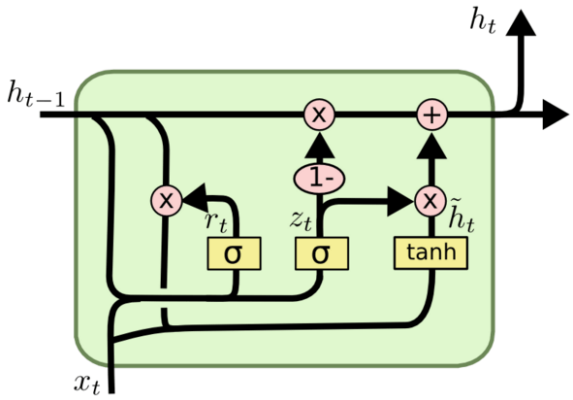


Figure 8: The inner workings of a GRU. GRUs are like a middle-ground between basic RNNs and LSTMs in that they have fewer gates than LSTMs, but have better memory than basic RNNs.

2.3.2 Gated Recurrent Unit (GRU)

A gated recurrent unit (GRU/GRULSTM) is a variant of the LSTM model, but is simpler to train due to it having fewer layers. Figure 8 illustrates how a GRU works:

1. Calculate the update gate z_t for time step t, where $z_t = \sigma(W^z x_t + U^z h_{t-1})$, W^z is the weight respective to input x and U^z is the weight respective to the previous state h_{t-1}; this computation allows the GRU to determine how much previous information from past time steps need to be passed along to the future (the next time step)

2. Decide on how much previous information to forget via the reset gate using the computation $r_t = \sigma(W^r x_t + U^r h_{t-1})$

3. Calculate the final memory of information at the current time step t to obtain the new h_t

With fewer gates than a LSTM but still maintaining use of update and reset gates, which are simpler versions of LSTM's forget gates, the GRU is easier to implement and faster to train, since computations for each gate can end up being very time consuming, especially when a set of input training data becomes extremely large.

3 Related Work

Research in image captioning has been a popular topic and has been explored in a variety of capacities. For example, Vinyals et al. [15] generate captions for single images, Antol et al. [1] generate answers when given an image and a natural language question, and Sadeghi and Farhadi [13] understands visual concepts.

3.1 Log-Bilinear Model for Image Captioning

Kiros et al. [7] produces captions for image input by using a basic neural network that performs linear computations. The linear computations allow the the network to make a prediction for a caption's content given a new image as input.

3.2 Visual Storytelling (VIST)

The VIST problem addresses a unique iteration of the caption generation problem that concerns sequenced captions creating a story based on given a sequence of images. It is actively being explored in workshops and challenges (Huang et al. [5]). VIST has referenced the works of Devlin et al. [2] and Karpathy and Fei-Fei [6], and aligns with Mathur et al. [11], which is largely inspired by Vinyals et al. [15]. That is, before we can generate a series of captions that create a story based on a sequence of images, we first have to be able to generate captions for a single image. This type of caption generation thus requires an image as input and produces text output. Extending this technique to take on multiple images as input would, according to VIST, require another RNN.

3.3 Captioning Images in Real-Time

Inspired by the approach of Vinyals et al. [15], Mathur et al. [11] created a neural network that uses an encoder-decoder architecture. The group used the MSCOCO dataset (Lin et al. [10]) along with Flickr8K and Flickr30K, all of which are a set of various images with captions, with TensorFlow to train the model. The encoder-decoder architecture is indicative of the neural network aiming to accomplish a goal that goes beyond solely classifying objects and actively produces output. The model uses a pre-trained encoder, which uses a CNN, and a decoder that uses LSTM. The model takes an image I and encodes it, vectorizing the information held in I and turning it into an embedding, F. It then suggests a caption for the image based on F and a probabilistic model that maximizes the probability of word p in said caption.

3.4 Patterns in Approaches of Related Works

The aforementioned works of Mathur et al. [11] and Karpathy and Fei-Fei [6] used some form of an RNN in achieving caption generation for images. However, each work took a different approach for encoding the input image. Using a variant of a CNN seemed popular for encoding the input image, as CNNs are usually implemented to categorize objects in an image. Since caption generation requires more than just categorization, it is imperative to include another function that can generate output. Since the output of

concern in image captioning is a caption that uses natural language, a RNN is a suitable model to use for decoding.

4 Experiment

4.1 Design

We use the Karpathy and Fei-Fei [6] model to approach the image captioning problem. This model implements an encoder-decoder architecture that uses a CNN as the encoder and a RNN for the decoder. Given an input image, the CNN, acting as the encoder, extracts the higher-level features of the image and vectorizes them into an embedding. The embedding is then fed into the RNN, acting as the decoder, so the RNN can generate a caption based off the input image.

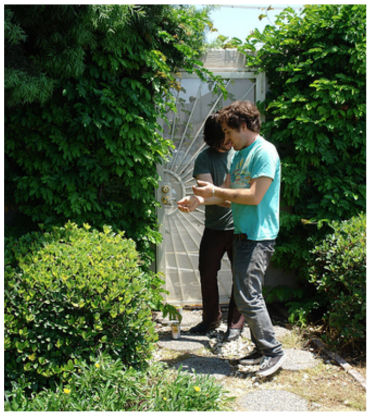
But how does the RNN *know* what words to generate, given that images made up of pixel values and not words? Flickr datasets([4], [12]), known as Flickr8k and Flickr30k (indicating the size of each dataset), supply captions for images that neural networks can use for training.

4.2 Dataset

The Flickr datasets contain thousands of images and data that can be used for training neural networks. Raw images are kept in a separate folder, identified only by a stream of numbers, from the JSON files that include image identification numbers, crowd-sourced captions that allow for neural network training, and what part of the neural network training it is a part of (Figure 9. Since neural network training is generally broken into three parts — training, validation, and testing — the Flickr datasets are split similarly. The training portion trains the network by having it perform computations and adjust its weights accordingly; the validation portion is run through the network after training is done as a simulation of testing new input images, but may also tweak the network weights; the testing portion contains new images that can be tested on the refined network. It should be noted that despite having similar formats and coming from the same source, the Flickr datasets are completely separate entities, so there is no overlap between the two. A Python script was written to confirm this exclusivity.

4.3 Methods

The Karpathy and Fei-Fei [6] model implements a CNN encoder, but has a RNN and LSTM decoder. Since we want captions to be robust and detailed, we use the LSTM to avoid short-term memory loss that can occur in longer sequences, as discussed in Section 2.3. To run the network, a Flickr dataset was loaded into



{'filename': '1000092795.jpg', 'imgid': 0, 'sentences': [{'tokens': ['two', 'young', 'guys', 'with', 'shaggy', 'hair', 'look', 'at', 'their', 'hands', 'while', 'hanging', 'out', 'in', 'the', 'yard'], 'raw': 'Two young guys with shaggy hair look at their hands while hanging out in the yard.', 'imgid': 0, 'sentid': 0}, {'tokens': ['two', 'young', 'white', 'males', 'are', 'outside', 'near', 'many', 'bushes'], 'raw': 'Two young, White males are outside near many bushes.', 'imgid': 0, 'sentid': 1}, {'tokens': ['two', 'men', 'in', 'green', 'shirts', 'are', 'standing', 'in', 'a', 'yard'], 'raw': 'Two men in green shirts are standing in a yard.', 'imgid': 0, 'sentid': 2}, {'tokens': ['a', 'man', 'in', 'a', 'blue', 'shirt', 'standing', 'in', 'a', 'garden'], 'raw': 'A man in a blue shirt standing in a garden.', 'imgid': 0, 'sentid': 3}, {'tokens': ['two', 'friends', 'enjoy', 'time', 'spent', 'together'], 'raw': 'Two friends enjoy time spent together.', 'imgid': 0, 'sentid': 4}], 'split': 'train', 'sentids': [0, 1, 2, 3, 4]}

Figure 9: The format of a single element for an image in the Flickr dataset with its respective image.

the neural network script and then executed for several hours to train the network's weights. For successful encoding, a .mat file containing the extracted higher-level features was required.

A vocabulary was maintained during training, which helped make caption generation more robust. The vocabulary was built during the LSTM's decoding process. Since the LSTM uses the extracted, vectorized data features from the CNN as a sort of reference for what words are associated with which features. The LSTM is, by nature (recall Figure 7), taking the extracted features *and* a tokenized version of sample captions (crowd-sourced captions written for images in the training set) such that for each image, each token of the training set caption for that image is somehow accounted for in the vocabulary. As the LSTM performs more computations on more training images, the vocabulary grows and becomes more refined.

Checkpoints stored information on the neural network's current state in training and were saved intermittently throughout training. We can run new testing data on a checkpoint to see any adjustments the neural network may have made. Any evident adjustments would be spotted in the generated captions (more on this in Section 5). When the network finished its training, the test portion of the dataset was executed and had its results recorded in a separate JSON file. The CNN-LSTM encoder-decoder network was first run on the Flickr8k dataset, taking approximately 12 hours to complete training. The Flickr30k run took approximately a week to complete its training. Results can be visualized using a corresponding HTML file (Figure 10).

The resultant captions were rated by BLEU, a popular automatic evaluation metric that scores machine translation based on the following perspective: "*The closer a machine translation is to a professional human translation, the better it is.*". Since there was no translation between languages (like what one might witness when using, say, Google Translate), how is BLEU relevant to rating a plain-English caption? BLEU simply checks the captions for syntactic correctness, similar to how a native English speaker might review a non-

Figure 10: A visualization of predicted captions on new images. 'logprob' refers to the neural network's confidence in its prediction based on the evaluation metrics courtesy of the actual captions and BLEU.

native English speaker's English translation and notice grammatical errors. Scores are calculated based on a machine's translation's closeness to a (correct) provided human translation(s). Thus, BLEU possesses a corpus of good quality human translations for reference.

5 Results

Results were gathered by running a python evaluation script. The script ran the test portion of the Flickr datasets and predicted captions for new images based on the weights that have been adjusted several times throughout the neural network's training. A log probability 'logprob' was calculated to represent the confidence the neural network's evaluation metric was in the prediction. The closer the probability was to 0, the more confident the prediction. However, a low log probability does not necessarily indicate a confident prediction. In fact, a low log probability with a blatantly wrong caption (i.e. Figure 13, right picture) is worse to have, since that implies something went wrong during training.

5.1 Flickr8k

The Flickr8k run culminated in several checkpoints that initially showed promising caption predictions, but fell victim to word bias (Figure 13). In particular, the words *man*, *red*, *shirt*, *standing*, *sitting*, and *bench*

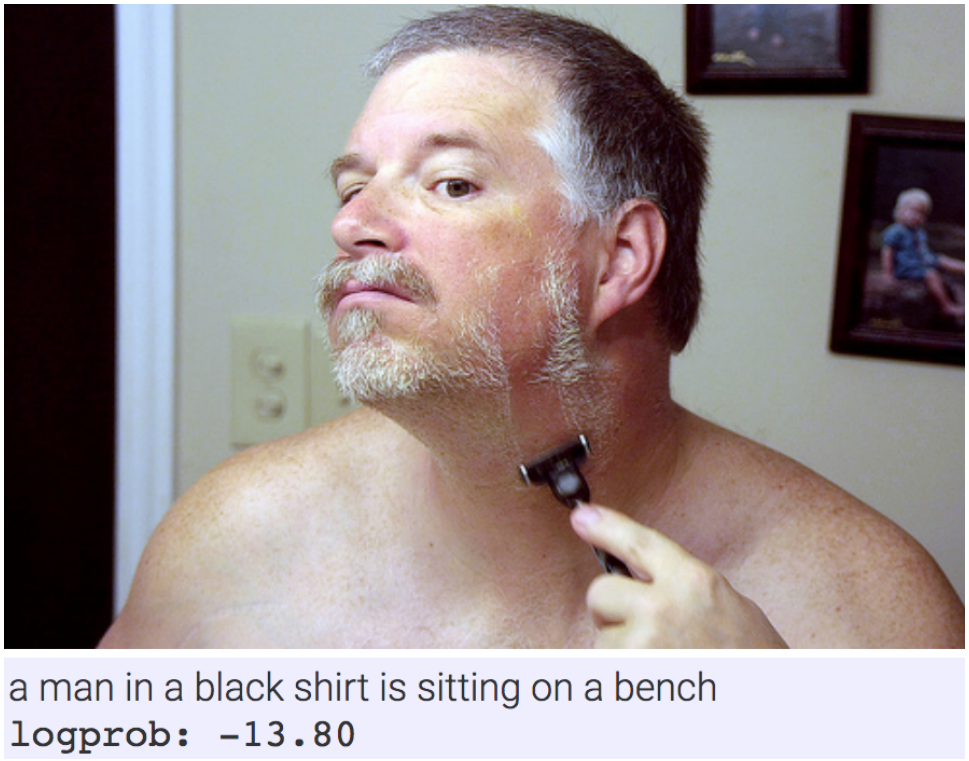


Figure 11: An example of a poor prediction based on training with Flickr30k.

appeared in more than half the predicted captions, and they frequently appeared together in some capacity. More often than not, the images described involved few to none of those words. Despite the glaring inaccuracies, the model was erroneously more confident in its prediction with the later checkpoints.

5.2 Flickr30k

Due to its significantly larger size, the Flickr30k dataset yielded several more checkpoints than Flickr8k. With more data to train on, the Flickr30k run theoretically would have smoothed out some of the word biases found in Flickr8k. However, the captions exhibited similar word bias, but with particular favoritism for the words *man*, *black*, *shirt*, *standing*, *sitting*, *bench*, *playing* and *guitar* (Figures 11 and 12). Similar to what likely happened with Flickr8k, the neural network must have had some weights adjusted incorrectly while training on the Flickr30k. With less accurate caption predictions toward the end of the neural network's training, the model was still more confident in its predictions toward the end of its training. That said, there are some exceptions that showed some more accurate captions, as shown in Figure 14, although those number of exceptions were very few.

a man in a black shirt is playing a guitar
logprob: -11.85

Figure 12: An example of poor prediction and erroneous object identification with Flickr30k.

Figure 13: Comparison of the first (left) and last (right) checkpoints in the Flickr8k test evaluation.

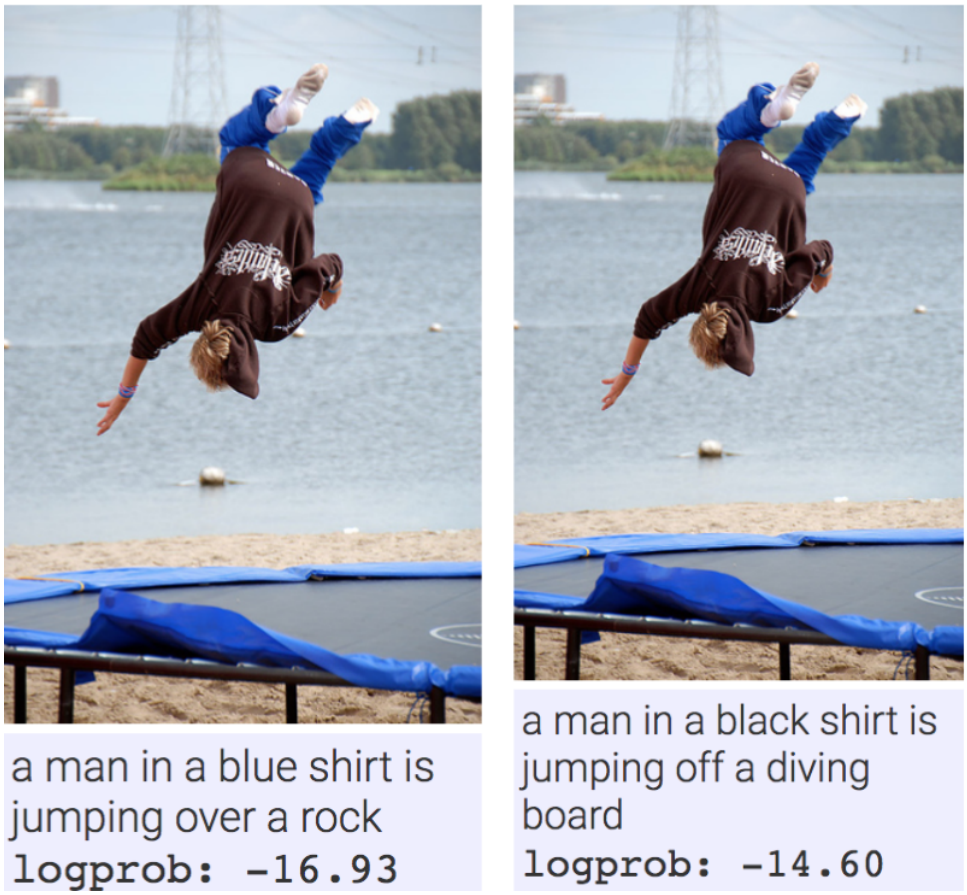


Figure 14: Comparison of the first (left) and last (right) checkpoints in the Flickr30k test evaluation.

6 Conclusion

The CNN/LSTM encoder-decoder is a promising model for image captioning. We have successfully generated captions for new input images that are syntactically sound. However, the training of the neural network needs to be tweaked, as word bias became evident and caused many captions to be inaccurate despite greater confidence in the predictions.

Appendices

A VIST Dataset

The VIST dataset can be obtained from the VIST site. VIST constructed the dataset from online image and video hosting website Flickr and by crowd-sourcing for image captions via the Flickr data release (Thomee et al. [14]). The dataset includes the following sets for both DII and SIS: training set, validation set, and test set. Recall that DII stands for ”descriptions-in-isolation” and SIS stands for ”stories-in-sequence”, where DII refers to captions for single images and SIS refers to captions that are a part of a story sequence. The DII and SIS contain information on the images in the training set (e.g. album source, date taken, URL) and annotations that serve as captions for respective images. The training set contains over 200,000 unique photos and is used to train neural networks. The validation set is used to make general improvements on the network, acting as an error checker or a boundary test. The test set simulates completely new input data for the network — the network's programmer(s) are not allowed to see the elements of the test set so as to avoid subconscious coding that could make special case dealings for the new input, which would be detrimental to the development of a truly robust neural network.

In the training set, each image is titled with a photo identification number and is in JPG format. For the purpose of this project, 13 GB of the original dataset were downloaded. Most, but not all, of the photos in the training dataset have accompanying information and annotations found in the DII and SIS sets (some photos may have been taken down by their respective owners).

The DII and SIS sets have very similar format, as shown in Figure 15. Each contain JSON files, which follow the general JSON format. In both DII and SIS, there are five attributes: images, info, albums, type, and annotations, all of which are Strings. The values of info and type are the version type (SIND v1.0) and description-in-isolation or story-in-sequence, respectively.

The 'images' and 'albums' attributes contain information on the training images' minute details pertain-

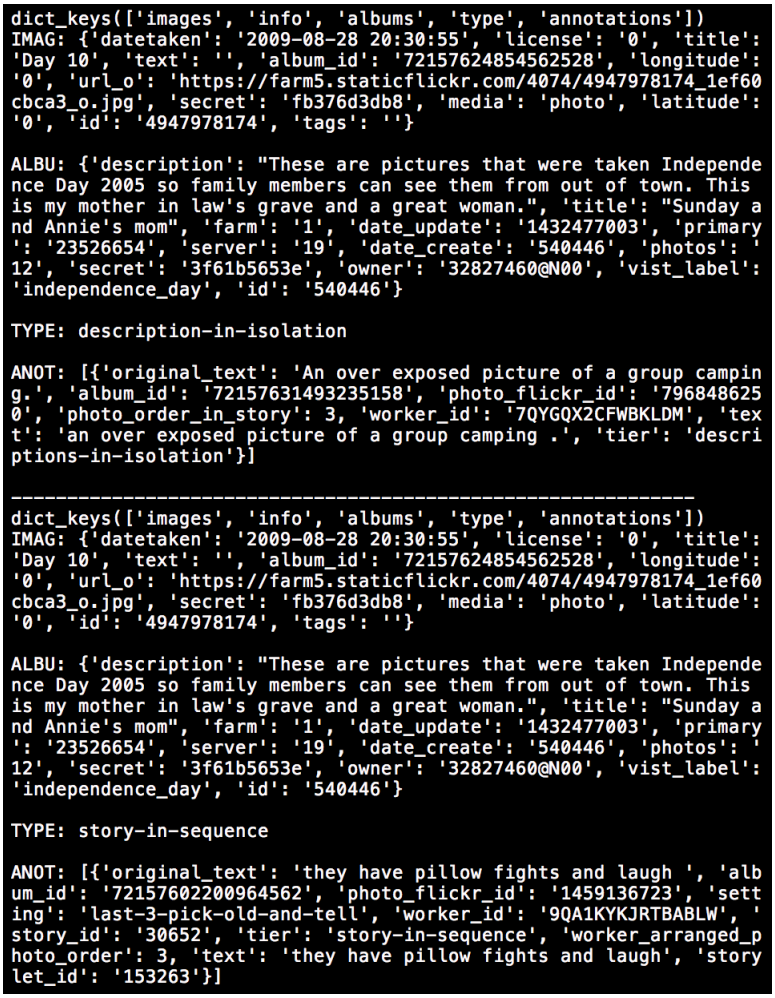


Figure 15: Printed output of a single image's attributes in the DII and SIS sets. The 'info' attribute was not printed because it is the same between DII and SIS.

Figure 16: An image that is a part of the training set.

ing to the images' respective sources. For instance, the 'images' attribute contains the image identification number, the original Flickr URL, and date it was taken, among other details. The 'albums' attribute contains details such as album identification numbers, which works similarly to the 'images' attribute's image identification number, the number of photos the album contains, and the album's title. These attributes are useful for parsing through the training set, especially when partitioning the dataset as a whole. In other words, we can observe the attributes of specific images in the training set by keying into the 'images' or 'albums' attributes, where the identification numbers can be used to extract data from the set.

The 'annotations' attribute contains the information needed to train the AI to generate captions for images. 'Annotations' has two versions of the caption accompanying a given photo: 'original text' and 'text'. Both versions have the same content, but 'text' spaces out the words and punctuation so that the whole string can be easily processed using a .split(" ") function. This way, words found at the end of a clause will not be erroneously recognized as a different word (e.g. "word" $\neq$ "word."). Thus, the 'text' value of the 'annotations' key of a photo will be used as input for the AI.

Table 1 corresponds to the image in Figure 16. The DII annotations give a top-level explanation of what is happening in the image while the SIS annotations seem to be fragments of a story. Since the annotations were generated by humans, we can observe some unique lingual divergences in annotations, but a general theme of birds preventing a car from getting somewhere appears to be evident.

B Notes Regarding New Images

The original aim of this thesis was to test the CNN-LSTM model on at least a portion of the VIST dataset. The Karpathy and Fei-Fei model has an image extraction script for both Matlab and Python. The Matlab

Type	Annotation
DII	a number of geese crossing a busy six lane street as seen from inside of a car .
	cars that are waiting as a group of birds cross the road .
	grey car stopped in the road to let ducks cross .
SIS	on our way to a wedding and we stop for a family of ducks !
	they are so funny when they waddle at high speeds .
	we 're on our way to watch [male] and [female] get married !
	the couple was on their way to the wedding .
	ran in to a family of ducks on the way to my cousins wedding .

Table 1: A side-by-side comparison of Descriptions-in-Isolation and Stories-in-Sequence annotations.

script is said to yield better feature extractions, but the Python extraction also works. Since both extraction scripts rely on a library called "caffe", anyone who wishes to implement the scripts must have all caffe dependencies installed. While most dependencies are supported on most common operating systems, there remain a few that do not support mainstream operating systems such as Mac OSX.

References

[1] Stanislaw Antol et al. "VQA: Visual Question Answering". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2425–2433.

[2] Jacob Devlin et al. "Language models for image captioning: The quirks and what works". In: *arXiv preprint arXiv:1505.01809* (2015).

[3] Ian J Goodfellow et al. "Multi-digit number recognition from street view imagery using deep convolutional neural networks". In: *arXiv preprint arXiv:1312.6082* (2013).

[4] Micah Hodosh, Peter Young, and Julia Hockenmaier. "Framing Image Description As a Ranking Task: Data, Models and Evaluation Metrics". In: *J. Artif. Int. Res.* 47.1 (May 2013), pp. 853–899. ISSN: 1076-9757. URL: http://dl.acm.org/citation.cfm?id=2566972.2566993.

[5] Ting-Hao K. Huang et al. "Visual Storytelling". In: *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*. 2016.

[6] A. Karpathy and L. Fei-Fei. "Deep visual-semantic alignments for generating image descriptions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, pp. 3128–3137. DOI: 10.1109/CVPR.2015.7298932.

[7] Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. "Multimodal neural language models". In: *International Conference on Machine Learning*. 2014, pp. 595–603.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[9] Steve Lawrence et al. "Face recognition: A convolutional neural-network approach". In: *IEEE transactions on neural networks* 8.1 (1997), pp. 98–113.

[10] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[11] P. Mathur et al. "Camera2Caption: A real-time image caption generator". In: *2017 International Conference on Computational Intelligence in Data Science(ICCIDS)*. June 2017, pp. 1–6. DOI: 10.1109/ICCIDS.2017.8272660.

[12] Bryan A Plummer et al. "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models". In: *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE. 2015, pp. 2641–2649.

[13] Mohammad Amin Sadeghi and Ali Farhadi. "Recognition using visual phrases". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 1745–1752.

[14] Bart Thomee et al. "The New Data and New Challenges in Multimedia Research". In: *CoRR* abs/1503.01817 (2015). arXiv: 1503.01817. URL: http://arxiv.org/abs/1503.01817.

[15] Oriol Vinyals et al. "Show and tell: A neural image caption generator". In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE. 2015, pp. 3156–3164.

[16] O. Vinyals et al. "Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (Apr. 2017), pp. 652–663. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2587640.

[17] Kelvin Xu et al. "Show, attend and tell: Neural image caption generation with visual attention". In: *International Conference on Machine Learning*. 2015, pp. 2048–2057.