Application of Attention Based Models in Images– Generating Image Captions using RNNs

Enrollment No.	18104043	18104082	18104084
Name of Student	Priyanka Srivastava	Ishita Batra	Rudranil Ghosh

Name of Supervisor Dr. K. Vimal Kumar

May 2021

Report for Final Evaluation of Minor-II Project DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

Introduction

General Introduction & Use cases

A caption is an explanatory text that you put close to your images. While a picture is worth a thousand words, sometimes you need a few words to magnify its effect. Captions for photos can summarize what'sin the photograph or tie it to your story.

The caption is an important element in your blog post for several reasons.

1. It's in prime real estate.

When you flip open a magazine, you're more likely to first pay your attention to the photos than the writing. Therefore, the area surrounding a photo is where readers pay their first attention to.

This is explained by the law of proximity. We consider images and captions as a single entity and pay more attention to them. A caption can grab this attention and compel the user to read your article to find more information.

2. Captions are scannable.

Even though you would like your audience to read your story from top to bottom, that's hardly ever the case. As Steve Krug explains in his popular book, *"Don't Make Me Think"*, humans are more likely to scan your article than read it top to bottom.

Having scannable items in your article makes it easier to navigate and digest information. This is why captions usually have unique font styles. This scannability makes the caption an important factor in deciding the worthiness of your content.

3. Captions connect text and photos.

A great use of caption is to put important information from your text into them. This makes the user connect to your story via the photograph. For example, let's say you are writing about animal extinction. A caption that mentions important numbers or facts can compel the reader to read more.

4. It helps to make more sense of a photo.

They say a photo is worth a thousand words. However, this is only true if your photo's composition draws the attention of the viewer to its subject. In this sense, a caption can be thought of as a composition tool. One way to draw your viewer's eye to your photo's subject is to use a photography composition rule such as the Rule of Thirds. Similarly, a caption can draw a viewer's eye to an important aspect of your photograph. What sets a caption apart from other composition tools is it can draw attention to information that is not even present in the photo!

5. It helps Search Engine Optimization (SEO).

Technology is making a lot of progress in image recognition. That being said, search engine bots don't understand what's in a photo. At least on a massive scale. Detailed captions next to the photo aids Google in making sense of what's in the photo. In return, this can help you get more traffic from Web and Image search queries.

Problem Statement

Caption generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph.

It requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order. Recently, deep learning methods have achieved state-of-the-art results on examples of this problem.

Deep learning methods have demonstrated state-of-the-art results on caption generation problems. What is most impressive about these methods is a single end-to-end model can be defined to predict a caption, given a photo, instead of requiring sophisticated data preparation or a pipeline of specifically designed models.

We aim to develop a image caption generating system which with the help of RNNs and Computervision, can prove to be a substantial tool to ease the work of content creators.

Our problem statement is to generate relevant captions to the given images

Process Outline

1. Data Preprocessing
    - Data Cleaning, handling missing values
    - Analysing the images
    - Scraping real time data (if time allows)
2. Splitting the cleansed data into training and test set
3. Building RNN model
4. Generate predicted caption
5. Build a web app prototype to demonstrate the same

Background Study - Literature Survey

Summary of Papers

Paper 1

TITLE OF PAPER	Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures
AUTHORS	Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, Barbara Plank
YEAR OF PUBLICATION	2017
PUBLISHING DETAILS	Journal of Artificial Intelligence Research 55
SUMMARY	Automatic description generation from natural images is a challenging problem that has recently received a large amount of interest from the computer vision and natural language processing communities. In this survey, we classify the existing approaches based on how they conceptualize this problem, viz., models that cast description as either generation problem or as a retrieval problem over a visual or multimodal representational space. A detailed review of existing models, highlighting their advantages and disadvantages has been provided.

Table 1: Research Paper 1

Paper 2

TITLE OF PAPER	What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?
AUTHORS	Marc Tanti, Albert Gatt, Kenneth P. Camilleri
YEAR OF PUBLICATION	2017
PUBLISHING DETAILS	arXiv preprint arXiv:1708.02043
SUMMARY	In neural image captioning systems, a recurrent neural network (RNN) is typically viewed as the primary 'generation' component. This view suggests that the image features should be 'injected' into the RNN. This is in fact the dominant view in the literature. Alternatively, the RNN can instead be viewed as only encoding the previously generated words. This view suggests that the RNN should only be used to encode linguistic features and that only the final representation should be 'merged' with the image features at a later stage. This paper compares these two architectures. We find that, in general, late merging outperforms injection, suggesting that RNNs are better viewed as encoders, rather than generators.

Table 2: Research Paper 2

Paper 3

TITLE OF PAPER	Show and Tell: A Neural Image Caption Generator
AUTHORS	Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan
YEAR OF PUBLICATION	2015
PUBLISHING DETAILS	Proceedings of the IEEEconference on computer vision and pattern recognition

SUMMARY	Being able to automatically describe the content of an image using properly formed English sentences is a very challenging task, but it could have great impact, for instance by helping visually impaired people better understand the content of images on the web. Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. In this paper, we present a generative model based on a deep recurrent architecture that combines recent advances in computer vision and machine translation and that can be used to generate natural sentences describing an image. The model is trained to maximize the likelihood of the target description sentence given the training image. Experiments on several datasets show the accuracy of the model and the fluency of the language it learns solely from image descriptions. The model is often quite accurate, which verifies both qualitatively and quantitatively.

Table 3: Research Paper 3

Paper 4

TITLE OF PAPER	IMAGE CAPTION GENERATOR USING DEEP LEARNING
AUTHORS	B.Krishnakumar, K.Kousalya, S.Gokul, R.Karthikeyan, D.Kaviyarasu
YEAR OF PUBLICATION	2019
PUBLISHING DETAILS	2nd InternationalConference on Advances in Science & Technology (ICAST)
SUMMARY	The model was successfully trained to generate the captions as expected for the images. Higher BLEU score indicates that the generated captions are very similar to those of the actual caption present on the images. We have implemented a CNN-RNN model by building an image caption generator. Their model depends on the data, so, it cannot predict the words that are out of its vocabulary. They used a small dataset consisting of 8091 images.

Table 4: Research Paper 4

Paper 5

TITLE OF PAPER	Camera2Caption: A real-time image caption generator
AUTHORS	Pranay Mathur, Aman Gill, Aayush Yadav, Anurag Mishra, Nand Kumar Bansode
YEAR OF PUBLICATION	2017
PUBLISHING DETAILS	InternationalConference on Computational Intelligence in Data Science (ICCIDS)

SUMMARY	They created and deployed a first of its kind mobile device based application to introduce possible use-cases for the model. They also tested the model against other recent works in image captioning. By making use of Inception architecture and by simplifying the overall flow design, they optimized the model to perform well in real time (on mobile devices) and managed to obtain high quality captions.

Table 5: Research Paper 5

Paper 6

TITLE OF PAPER	Image Caption Generator Based On Deep Neural Networks
AUTHORS	Jianhui Chen, Wenqiang Dong, Minchen Li
YEAR OF PUBLICATION	2015
PUBLISHING DETAILS	Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing
SUMMARY	They analyzed and modified an image captioning method LRCN. To understand the method deeply, they decomposed the method to CNN, RNN, and sentence generation. For each part, they modified or replaced the component to see the influence on the final result. The modified method is evaluated on the COCO caption corpus. Experiment results show that: first the VGGNet outperforms the AlexNet and GoogLeNet in BLEU score measurement; second, the simplified GRU model achieves comparable results with more complicated LSTM model; third, increasing the beam size increase the BLEU score in general but does not necessarily increase the quality of the description which is judged by humans.

Table 6: Research Paper 6

Analysis, Design and Modelling

Neural networks are a specific set of algorithms that has revolutionized the field of machine learning. They are inspired by biological neural networks and the current so called deep neural networks have proven to work quite very well. This project would use **Recurrent Networks (RNNs) as they** have directed cycles in their connection graph. That means you we sometimes get back to where we started by following the arrows. However, They can have complicated dynamics and this can make them very difficult to train. For a project like this, Recurrent Neural networks were the first choice since there are a very natural way to model sequential data. They are equivalent to very deep nets with one hidden layer per time slice; except that they use the same weights at every time slice and they get input at every time slice.

We will describe the model in three parts:

- **Photo Feature Extractor**. This is a 16-layer VGG model pre-trained on the ImageNet dataset. We have pre-processed the photos with the VGG model (without the output layer) and will use the extracted features predicted by this model as input.
- **Sequence Processor**. This is a word embedding layer for handling the text input, followed by a Long Short-Term Memory (LSTM) recurrent neural network layer.
- **Decoder** (for lack of a better name). Both the feature extractor and sequence processor output a fixed-length vector. These are merged together and processed by a Dense layer to make a final prediction.

The Photo Feature Extractor model expects input photo features to be a vector of 4,096 elements. These are processed by a Dense layer to produce a 256 element representation of the photo.

The Sequence Processor model expects input sequences with a pre-defined length (34 words) which are fed into an Embedding layer that uses a mask to ignore padded values. This is followed by an LSTM layer with 256 memory units.

Both the input models produce a 256 element vector. Further, both input models use regularization in the form of 50% dropout. This is to reduce overfitting the training dataset, as this model configuration learns very fast. The Decoder model merges the vectors from both input models using an addition operation. This is then fed to a Dense 256 neuron layer and then to a final output Dense layer that makes a softmax prediction over the entire output vocabulary for the next word in the sequence.

Neural Networks

Neural Networks are set of algorithms which closely resemble the human brain and are designed to recognize patterns. They interpret sensory data through a machine perception, labelling or clustering raw input. They can recognize numerical patterns, contained in vectors, into which all real-world data (images, sound, text or time series), must be translated. Artificial neural networks are composed of a large number of highly interconnected processing elements (neuron) working together to solve a problem

Recurrent Neural Networks (RNN)

Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.

Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other.

Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. It trains the model by using back-propagation.

Photo Feature Extraction

What is feature extraction?

Feature extraction is a part of the dimensionality reduction process, in which, an initial set of the raw data is divided and reduced to more manageable groups. So when you want to process it will be easier. The most important characteristic of these large data sets is that they have a large number of variables. These variables require a lot of computing resources to process them. Feature extraction helps to get the best feature from those big data sets by select and combine variables into features, thus, effectively reducing the amount of data. These features are easy to process, but still able to describe the actual data set with the accuracy and originality.

Why is feature extraction useful?

The technique of extracting the features is useful when we have a large data set and need to reduce the number of resources without losing any important or relevant information. Feature extraction helps to reduce the amount of redundant data from the data set. In the end, the reduction of the data helps to build the model with less machine's efforts and also increase the speed of learning and generalization steps in the machine learning process.

What is VGG16?

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous models submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.

Attention within Sequences

Attention is the idea of freeing the encoder-decoder architecture from the fixed-length internal representation.

This is achieved by keeping the intermediate outputs from the encoder LSTM from each step of the input sequence and training the model to learn to pay selective attention to these inputs and relate them to items in the output sequence. Put another way, each item in the output sequence is conditional on selective items in the input sequence.

Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words, it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding the translation. This frees a neural translation model from having to squash all the information of a source sentence, regardless of its length, into a fixed-length vector.

This increases the computational burden of the model, but results in a more targeted and better-performing model.

In addition, the model is also able to show how attention is paid to the input sequence when predicting the output sequence. This can help in understanding and diagnosing exactly what the model is considering and to what degree for specific input-output pairs.

The proposed approach provides an intuitive way to inspect the (soft-)alignment between the words in a generated translation and those in a source sentence. This is done by visualizing the annotation weights. Each row of a matrix in each plot indicates the weights associated with the annotations. From this we see which positions in the source sentence were considered more important when generating the target word.

The Concept of Attention

1. When humans see any image, they pay attention to certain parts then they change their attention to the next few parts and so on.
2. Intuitively think of your teacher correcting your 10 mark answer in your History paper. The teacher would have a key with them in which certain important points /words are there. So in your answer sheet your teacher would look for these important words in the key. More attention is paid to the keywords.
3. Hence humans change their attention from one sequence of words to another sequence of words when given a lengthy input sequence
4. **So, instead of looking at all the parts of the image, we can increase the importance of specific parts of the image sequence that result in the target sequence.** This is the basic idea behind the attention mechanism.
5. Attention mechanism makes use of bidirectional RNNs The regular RNN is unidirectional as the sequence is processed from the first word to the last word. In bidirectional RNN we will have a connection in forward and reverse direction.
6. So in addition to the forward connection, there is also a backward connection for each of the neurons.

There are 2 different classes of attention mechanism depending on the way the attended context vector is derived:

- Global Attention-Here, the attention is placed on all the source positions. In other words, **all the hidden states of the encoder are considered for deriving the attended context vector.**
- Local Attention-Here, the attention is placed on only a few source positions. **Only a few hidden states of the encoder are considered for deriving the attended context vector.**

What is Attention?

"Can I have your Attention please!". The introduction of the Attention Mechanism in deep learning has improved the success of various models in recent years and continues to be an omnipresent component in state-of-the-art models. Therefore, it is vital that it is understood how it goes about achieving its effectiveness and thus used in this project.

When it is thought about the English word "Attention", it is known that it means directing focus at something and taking greater notice. The Attention mechanism in Deep Learning is based off this concept of directing the focus, and it pays greater attention to certain factors when processing the data. In broad terms, Attention is one component of a network's architecture, and is in charge of managing and quantifying the interdependence between the input and output elements or within the input elements

There are two different major types of Attention:

Bahdanau Attention	Luong Attention
It is commonly referred to as Additive Attention, came from a paper by Dzmitry Bahdanau.	It is often referred to as Multiplicative Attention and was built on top of the Attention mechanism proposed by Bahdanau.
It takes concatenation of forward and backward source hidden state.	It uses top hidden layer states in both of encoder and decoder.
But in the Bahdanau at time t we consider about t-1 hidden state of the decoder. Then we calculate alignment, context vectors as above. But then we concatenate this context with hidden state of the decoder at t-1.	In Luong attention they get the decoder hidden state at time t. Then calculate attention scores and from that get the context vector which will be concatenated with hidden state of the decoder and then predict.
Bahdanau has only concatenated score alignment model.	Luong has different types of alignments

Table 7: Comparison of Attention Models

(Ref: Chorowski, Jan, et al. "Attention-based models for speech recognition." *arXiv preprint arXiv:1506.07503* (2015).)

Advantages of Attention in general?

Recurrent Neural Networks (RNNs) are powerful neural network architectures used for modeling sequences. LSTM (Long Short-Term Memory) based RNNs are surprisingly good at capturing long-term dependencies in the sequences. A barebones sequence-to-sequence/encoder-decoder architecture performs incredibly well in tasks like Machine Translation. This is the diagram of the Attention model shown in Bahdanau's paper.

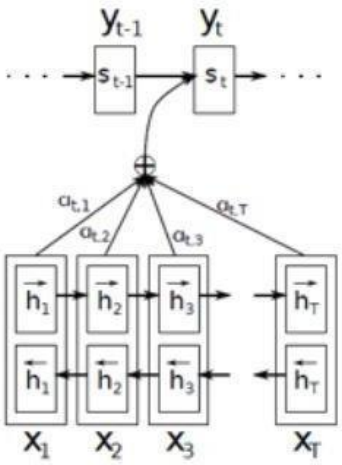


Fig 1: Bahdanau Attention Model

(Ref: Chorowski, Jan, et al. "Attention-based models for speech recognition." *arXiv preprint arXiv:1506.07503* (2015).)

A typical sequence-sequence architecture consists of an encoder and a decoder RNN. The encoder processes a source sequence and reduces it into a fixed length vector – the context, and the decoder generates a target sequence, token by token, conditioned on the context. The context is usually the final state of the encoder RNN. The final state of the encoder RNN ultimately decides the decoding process, or at least heavily influences it, while the previous states of the RNN, {h0, h1, . . h5} do not have any influence over the decoding process. Next observation is that the final encoder state, a fixed length vector, represents the essence or the meaning of the source sequence in the context of translation. When the source sequence is too long and contains multiple information-rich phrases apart from each other, the burden of capturing and condensing the information into a fixed length vector representation becomes too much for the encoder to bear. This inability leads to loss of information which gets reflected in the generated target sequence.

Thus, we need a mechanism to allow the decoder to selectively (dynamically) focus on the information-rich phrases in the source sequence. The decoder would take a peek at the encoder states that are relevant to the current decoding step. This relevance sorting is precisely what the Attention Mechanism does by allowing the decoder to pay attention to different parts of the source sequence at different decoding steps.

What is seq2seq learning with Attention?

Recurrent Neural Networks (or more precisely LSTM/GRU) have been found to be very effective in solving complex sequence related problems given a large amount of data. They have real time applications in speech recognition, Natural Language Processing (NLP) problems, time series forecasting, etc. Sequence to Sequence (often abbreviated to seq2seq) models are a special class of

Recurrent Neural Network architectures typically used (but not restricted) to solve complex Language related problems like Machine Translation, Question Answering, creating Chat-bots or as used in this project - Text Summarization. Attention was originally introduced as a solution to address the main

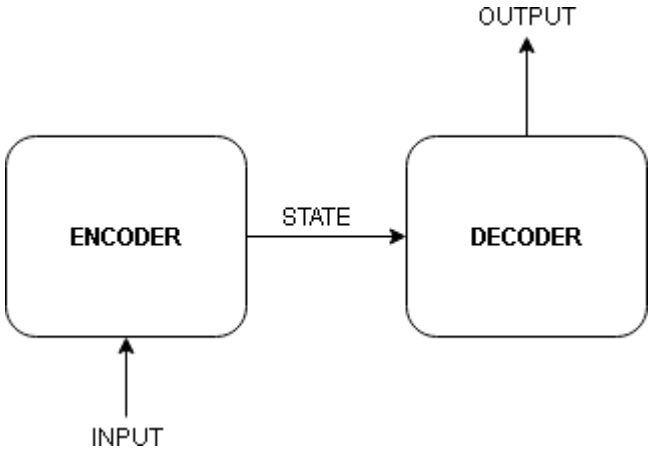


Fig 2: Working of an encoder-decoder (Created on draw.io)

issue surrounding seq2seq models, and to great success. Seq2seq model is also known as the Encoder-Decoder model.

The standard seq2seq model is generally unable to accurately process long input sequences, since only the last hidden state of the encoder RNN is used as the context vector for the decoder. On the other hand, the Attention Mechanism directly addresses this issue as it retains and utilizes all the hidden states of the input sequence during the decoding process. It does this by creating a unique mapping between each time step of the decoder output to all the encoder hidden states. This means that for each output that the decoder makes, it has access to the entire input sequence and can selectively pick out specific elements from that sequence to produce the output.

Therefore, the mechanism allows the model to focus and place more “Attention” on the relevant parts of the input sequence as needed.

RNN VS LSTM

All RNNs have feedback loops in the recurrent layer. This lets them maintain information in 'memory' over time. But, it can be difficult to train standard RNNs to solve problems that require learning long-term temporal dependencies. This is because the gradient of the loss function decays exponentially with time (called the vanishing gradient problem). LSTM networks are a type of RNN that uses special units in addition to standard units. LSTM units include a 'memory cell' that can maintain information in memory for long periods of time. A set of gates is used to control when information enters the memory, when it's output, and when it's forgotten. This architecture lets them learn longer-term dependencies. GRUs are similar to LSTMs, but use a simplified structure. They also use a set of gates to control the flow of information, but they don't use separate memory cells, and they use fewer gates.

Standard RNNs (Recurrent Neural Networks) suffer from vanishing and exploding gradient problems. LSTMs (Long Short Term Memory) deal with these problems by introducing new gates, such as input and forget gates, which allow for a better control over the gradient flow and enable better preservation of “long-range dependencies”. The long range dependency in RNN is resolved by increasing the number of repeating layer in LSTM. LSTMs are often referred to as fancy RNNs. Vanilla RNNs do not have a cell state. They only have hidden states and those hidden states serve as the memory for RNNs.

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they’d be extremely useful. But can they? It depends.

Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in “the clouds are in the *sky*,” we don’t need any further context – it’s pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it’s needed is small, RNNs can learn to use the past information.

But there are also cases where we need more context. Consider trying to predict the last word in the text “I grew up in France I speak fluent *French*.” Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back. It’s entirely possible for the gap between the relevant information and the point where it is needed to become very large.

Meanwhile, LSTM has both cell states and a hidden state. The cell state has the ability to remove or add information to the cell, regulated by "gates". And because of this "cell", in theory, LSTM should be able to handle the long-term dependency (in practice, it's difficult to do so.)

Long Short-Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn! All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged.

LSTMs were a big step in what we can accomplish with RNNs. It’s natural to wonder: is there another big step? A common opinion among researchers is: “Yes! There is a next step and it’s attention!” The idea is to let every step of an RNN pick information to look at from some larger collection of information. For example, if you are using an RNN to create a caption describing an image, it might pick a part of the image to look at for every word it outputs

Architecture

The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.
Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.
Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

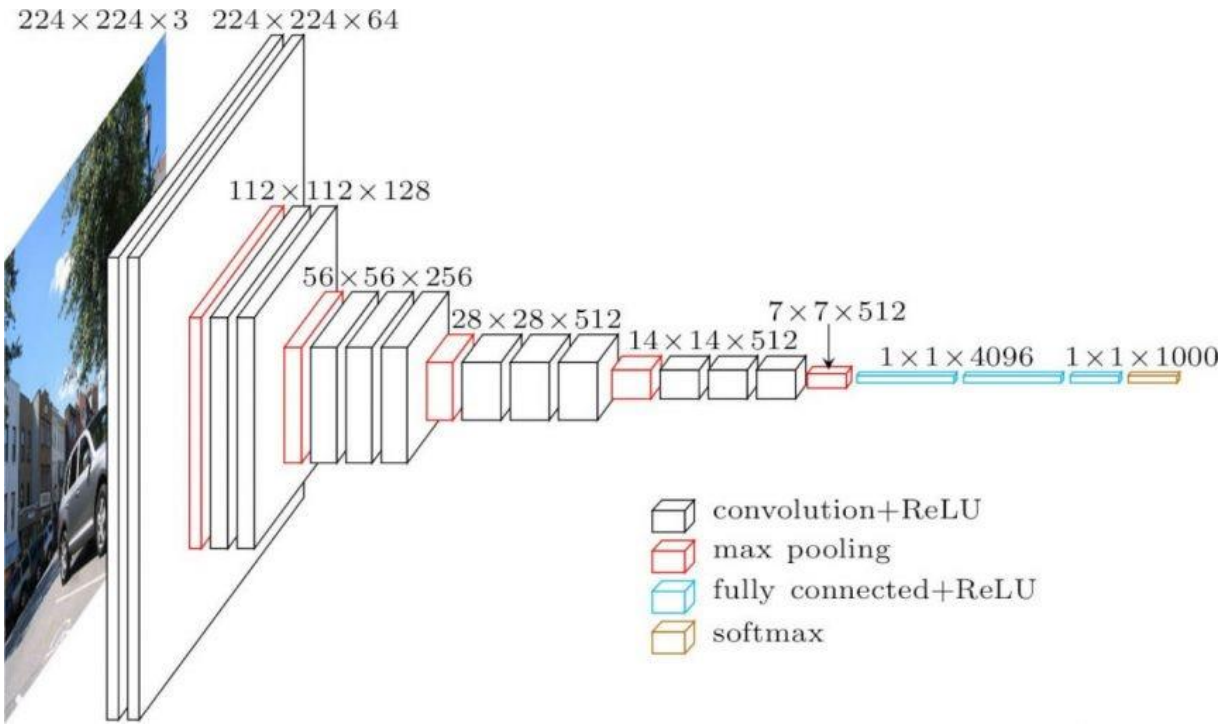


Fig 3: Architecture for the attention-based caption generator (Source: Hackernoon)

Sequence Processor

What is a sequence processor?

Sequence models are the machine learning models that input or output sequences of data. Sequential data includes text streams, audio clips, video clips, time-series data and etc. Recurrent Neural Networks (RNNs) is a popular algorithm used in sequence models.

Recurrent Neural Network (RNN) is a Deep learning algorithm and it is a type of Artificial Neural Network architecture that is specialized for processing sequential data. RNNs are mostly used in the field of Natural Language Processing (NLP). RNN maintains internal memory, due to this they are very efficient for machine learning problems that involve sequential data. RNNs are also used in time series predictions as well.

There are several RNN architectures based on the number of inputs and outputs,
1. One to Many Architecture: Image captioning is one good example of this architecture. In image captioning, it takes one image and then outputs a sequence of words. Here there is only one input but many outputs.

2. Many to One Architecture: Sentiment classification is one good example of this architecture. In sentiment classification, a given sentence is classified as positive or negative. In this case, the input is a sequence of words and output is a binary classification.

3. Many to Many Architecture: There are two cases in many to many architectures,

- **The first type:**

  When the input length equals to the output length. Name entity recognition is one good example where the number of words in the input sequence is equal to the number of words in the output sequence.

- **The second type:**

  Many to many architecture is when input length does not equal to the output length. Machine translation is one good scenario for this architecture. In machine translation, RNN reads a sentence in one language and then converts it to another language. Here input length and output length are different.

But traditional RNNs are not good at capturing long-range dependencies. This is mainly due to the vanishing gradient problem. When training very deep network gradients or the derivatives decreases exponentially as it propagates down the layers. This is known as Vanishing Gradient Problem. These gradients are used to update the weights of neural networks. When the gradients vanish then the weights will not be updated. Sometimes it will completely stop the neural network from training. This vanishing gradient problem is a common issue in very deep neural networks.

Decoder

The Encoder-Decoder LSTM is a recurrent neural network designed to address sequence-to-sequence problems, sometimes called seq2seq. Sequence-to-sequence prediction problems are challenging because the number of items in the input and output sequences can vary.

A standard encoder-decoder recurrent neural network architecture is used to address the image caption generation problem. This involves two elements:

1. Encoder: A network model that reads the photograph input and encodes the content into a fixed-length vector using an internal representation.
2. Decoder: A network model that reads the encoded photograph and generates the textual description output.

Generally, a convolutional neural network is used to encode the images and a recurrent neural network, such as a Long Short-Term Memory network, is used to either encode the text sequence generated so far, and/or generate the next word in the sequence. There are many ways to realize this architecture for the problem of caption generation. It is common to use a pre-trained convolutional neural network model trained on a challenging photograph classification problem to encode the photograph. The pre-trained model can be loaded, the output of the model removed, and the internal representation of the photograph used as the encoding or internal representation of the input image. It is also common to frame the problem such that the model generates one word of the output textual description, given both the photograph and the description generated so far as input. In this framing, the model is called recursively until the entire output sequence is generated.

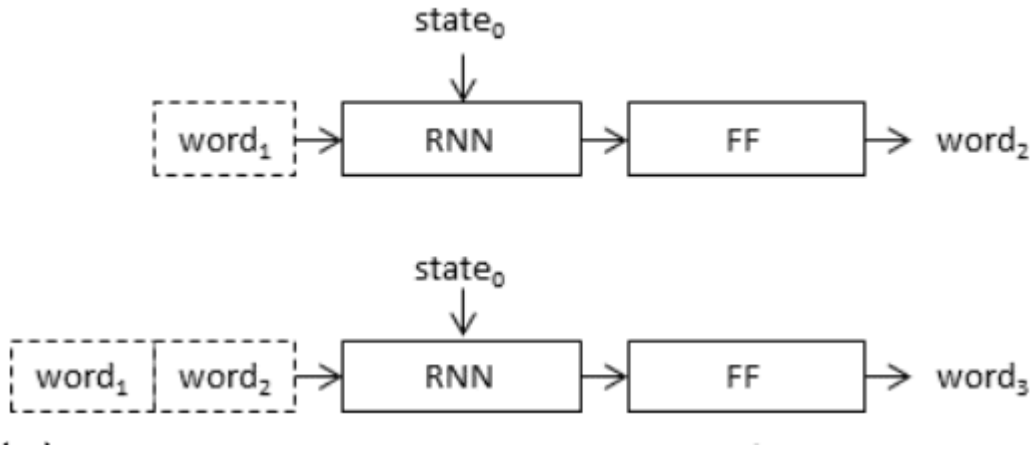


Fig 4: Working of an RNN for encoding the image. (Created on draw.io)

This framing can be implemented using one of two architectures:

Inject Model

The inject model combines the encoded form of the image with each word from the text description generated so-far. The approach uses the recurrent neural network as a text generation model that uses a sequence of both image and word information as input in order to generate the next word in the sequence.

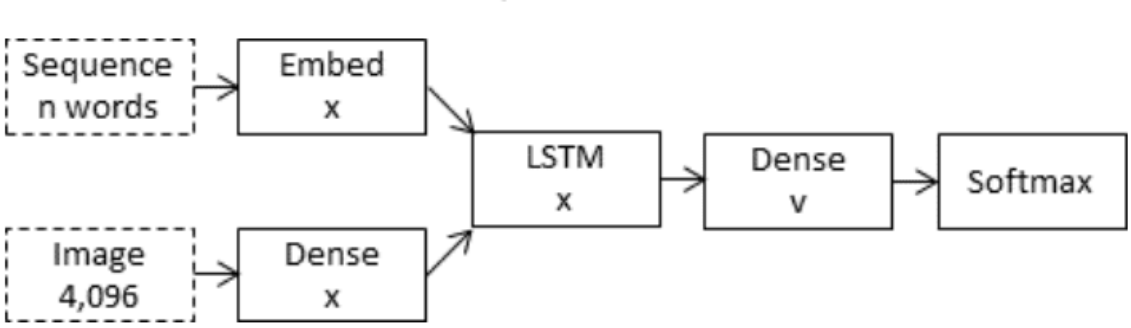


Fig 5: Inject Model for a decoder. (Created on draw.io)

Merge Model

The merge model combines both the encoded form of the image input with the encoded form of the text description generated so far. The combination of these two encoded inputs is then used by a very simple decoder model to generate the next word in the sequence. The approach uses the recurrent neural network only to encode the text generated so far.

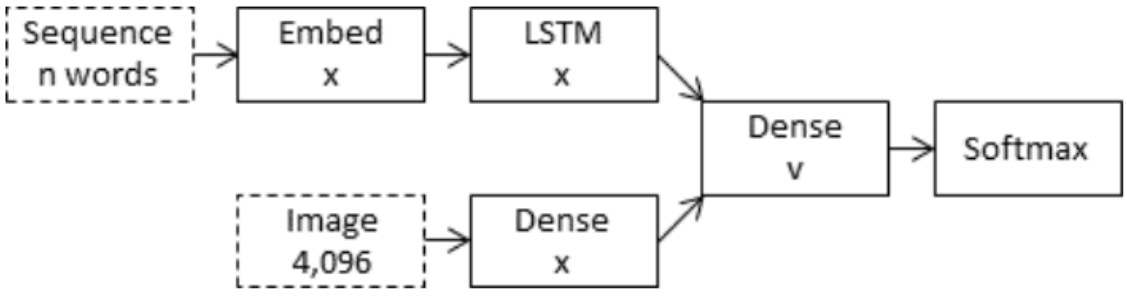


Fig 6: Merge Model for a decoder. (Created on draw.io)

We would be using the merge model of decoder as both the input models i.e. feature extractor and sequence processor produce a 256-element vector. Further, both input models use regularization in the form of 50% dropout. This is to reduce overfitting the training dataset, as this model configuration learns very fast. The Decoder model merges the vectors from both input models using an addition operation. This is then fed to a Dense 256 neuron layer and then to a final output Dense layer that makes a softmax prediction over the entire output vocabulary for the next word in the sequence.

Deep learning architectures/models to be used

Recurrent Neural Networks: Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist. A **recurrent neural network** (**RNN**) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. he main specialty in RNNs is the ability to model short term dependencies. This is due to the hidden state in the RNN. It retains information from one time step to another flowing through the unrolled RNN units. Each unrolled RNN unit has a hidden state. The current time steps hidden state is calculated using information of theprevious time step's hidden state and the current input. This process helps to retain information on what the model saw in the previous time step when processing the current time steps information.

Now, the loops make recurrent neural networks seem kind of mysterious. However, if we think a bit more, it turns out that they aren't all that different than a normal neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. These loops make recurrent neural networks seem kind of mysterious. However, if you think a bit more, it turns out that they aren't all that different than a normal

neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they'd be extremely useful. But can they? It depends.

Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in "the clouds are in the *sky*," we don't need any further context – it's pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.

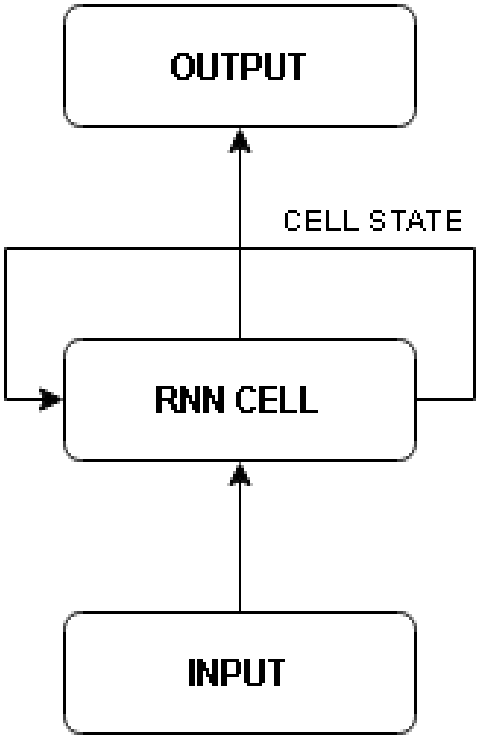


Fig 7: Rolled version of an RNN (Created on draw.io)

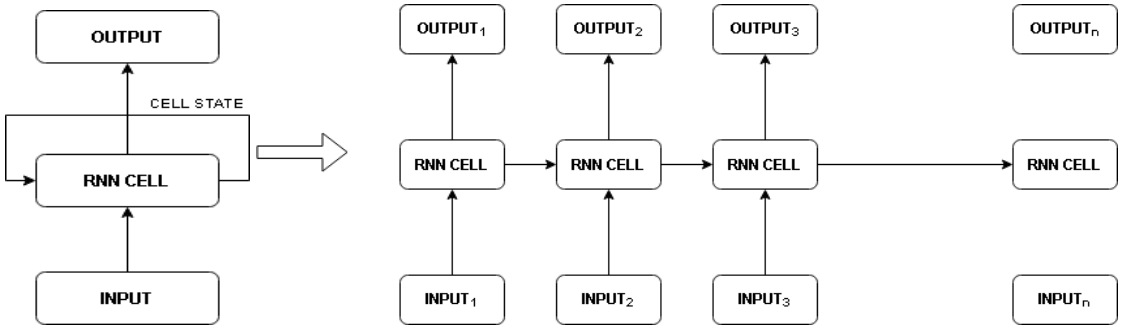


Fig. 8: An unrolled RNN (Created on draw.io)

Seq-2Seq Model – Now the question is, What if the input x is a sequence? What if x is a sequence ofwords. In most languages sequence of words matters a lot. We need to somehow preserve the sequence of words.The core idea here is if output depends on a sequence of inputs then we need to build a new type of neural network which gives importance to sequence information, which somehow retains and leverages the sequence information. Seq2Seq is a method of encoder-decoder based machine translation that maps an input of sequence to an output of sequence with a tag and attention value.

The idea is to use 2 RNNs that will work together with a special token and trying to predict the next state sequence from the previous sequence. Seq2Seq Model is a kind of model that use Encoder and a Decoder on top of the model.

The context vector may be a fixed-length encoding as in the simple Encoder-Decoder architecture, or may be a more expressive form filtered via an attention mechanism.

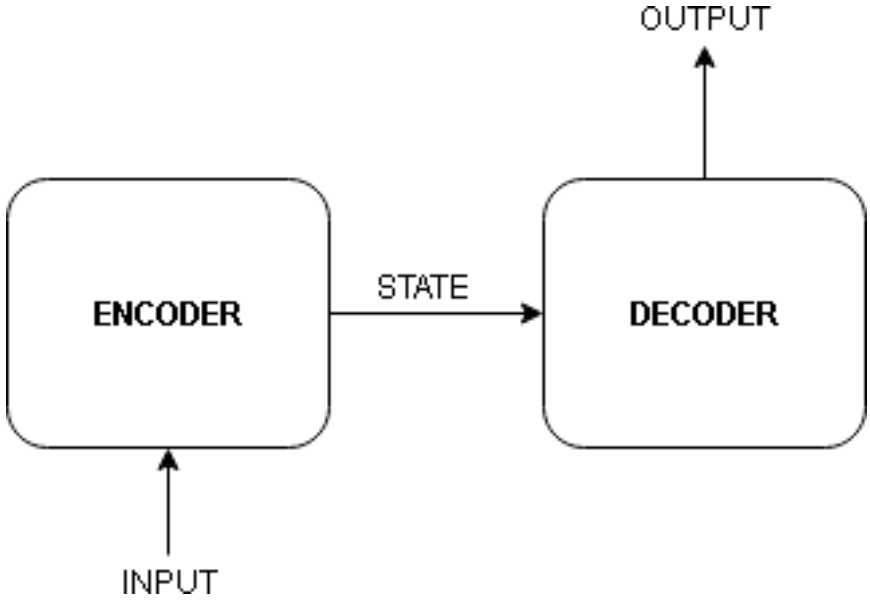


Fig 9: Basic Functioning of an Encoder-Decoder (Created on draw.io)

Long Short-Term Memory (LSTM): Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behaviour, not something they struggle to learn. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

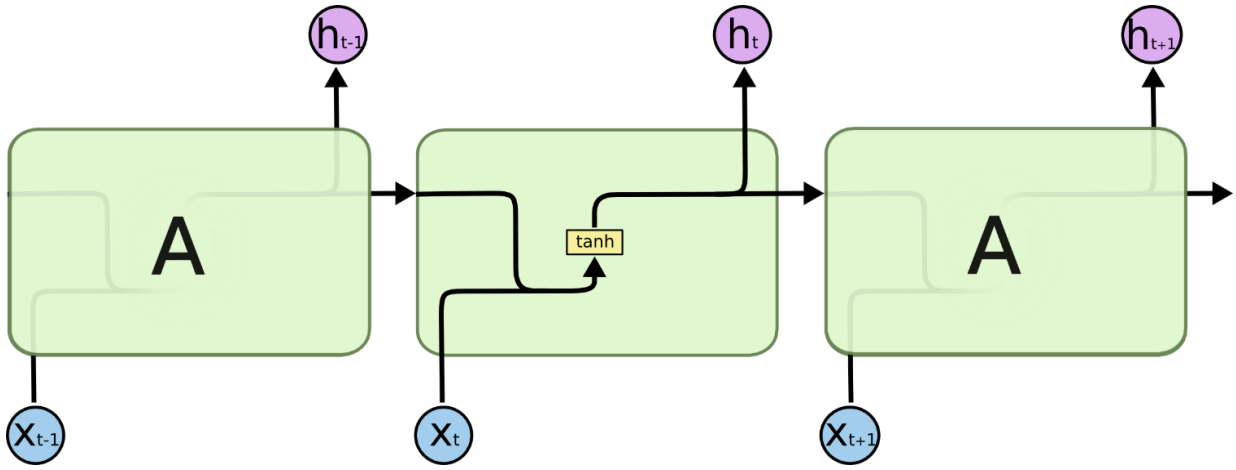


Fig 10: The repeating module in a standard RNN contains a single layer.

(Source – Hackernoon)

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four of them.

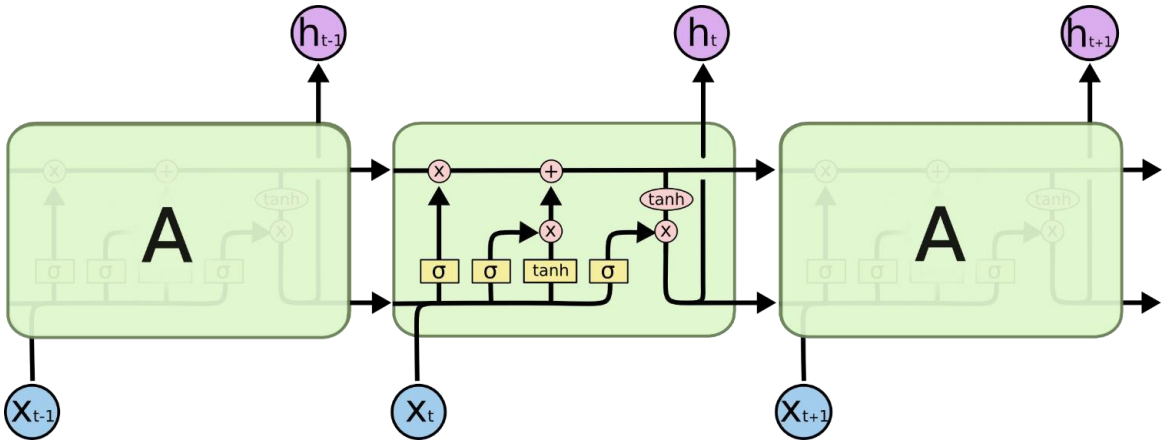


Fig 11: The repeating module in an LSTM contains four interacting layers.

(Source Hackernoon)

Screenshots of the outputs

```
The number of unique file names : 8092
The distribution of the number of captions for each image:
                    filename index  \
0  1000268201_693b08cb0e.jpg     0
1  1000268201_693b08cb0e.jpg     1
2  1000268201_693b08cb0e.jpg     2
3  1000268201_693b08cb0e.jpg     3
4  1000268201_693b08cb0e.jpg     4

                                             caption
0  a child in a pink dress is climbing up a set o...
1              a girl going into a wooden building .
2   a little girl climbing into a wooden playhouse .
3  a little girl climbing the stairs to her playh...
4  a little girl in a pink dress going into a woo...
```

Fig 12: Importing the image dataset and its respective captions

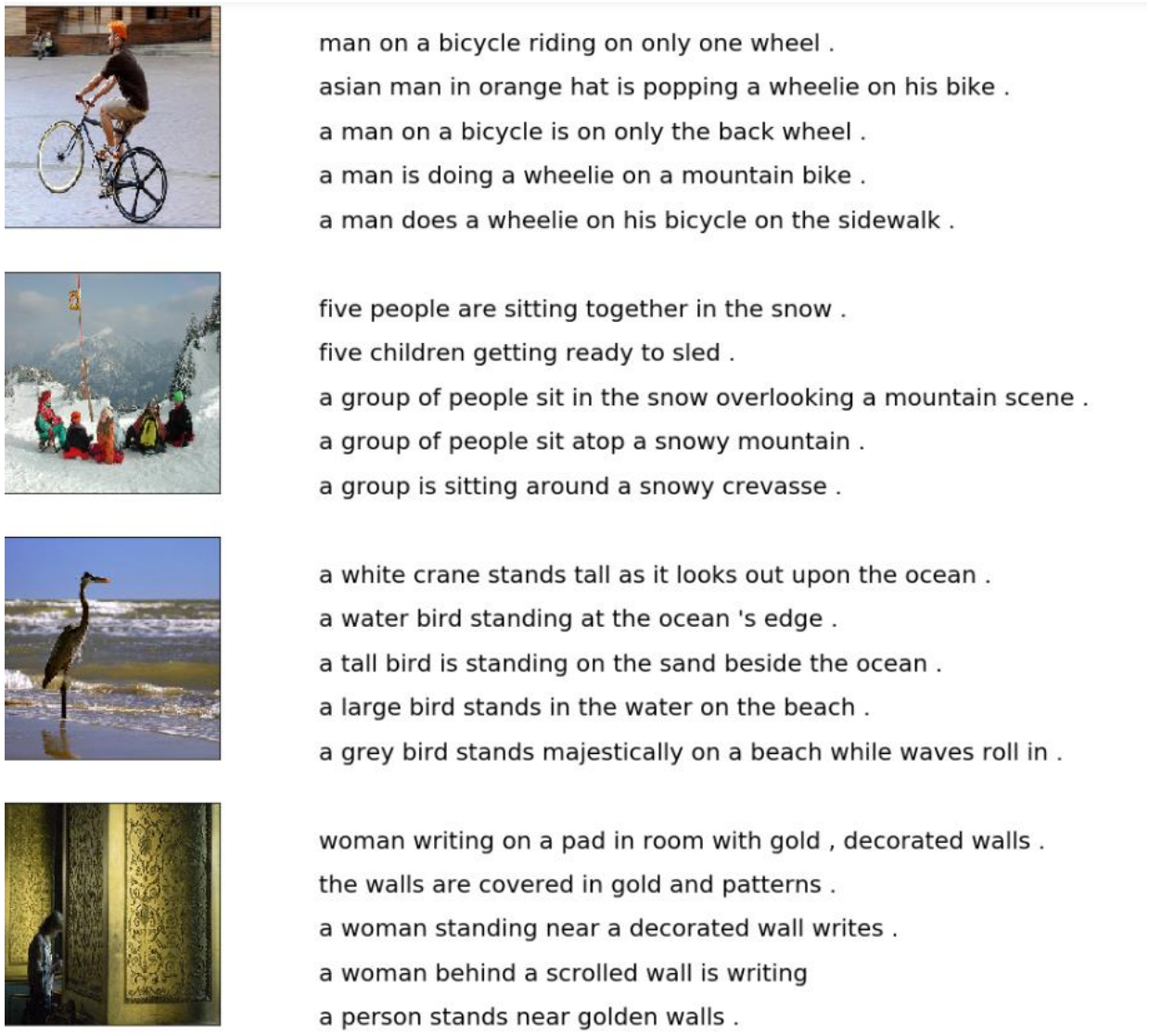


Fig 13: Plotting few images and their captions from the dataset

```
I ate 1000 apples and a banana. I have python v2.7. It's 2:30 pm. Could you buy me iphone7?

Remove punctuations..
I ate 1000 apples and a banana I have python v27 Its 230 pm Could you buy me iphone7

Remove a single character word..
 ate 1000 apples and banana have python v27 Its 230 pm Could you buy me iphone7

Remove words with numeric values..
    ate         : True
    1000        : False
    apples      : True
    and         : True
    banana      : True
    have        : True
    python      : True
    v27         : False
    Its         : True
    230         : False
    pm          : True
    Could       : True
    you         : True
    buy         : True
    me          : True
    iphone7     : False
 ate apples and banana have python Its pm Could you buy me
```

Fig 14: Cleaning the captions for pre-processing

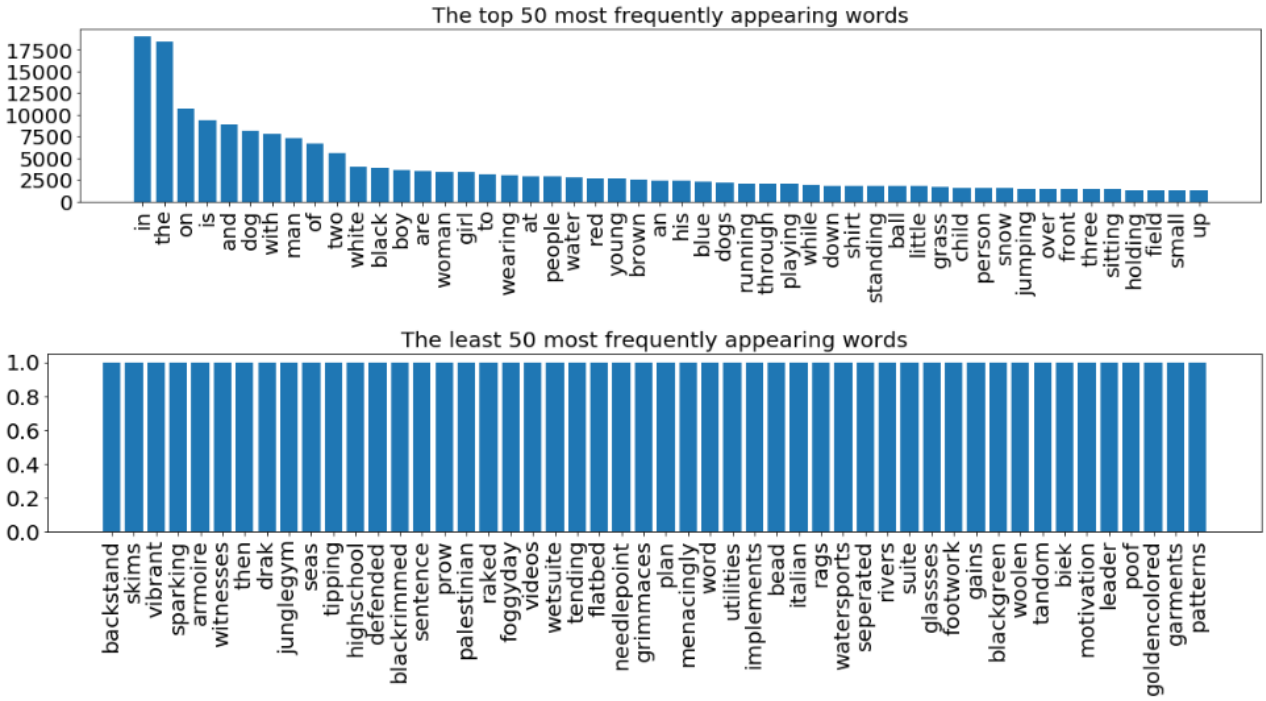


Fig 15: Plotting top 50 words that appeared in the dataset

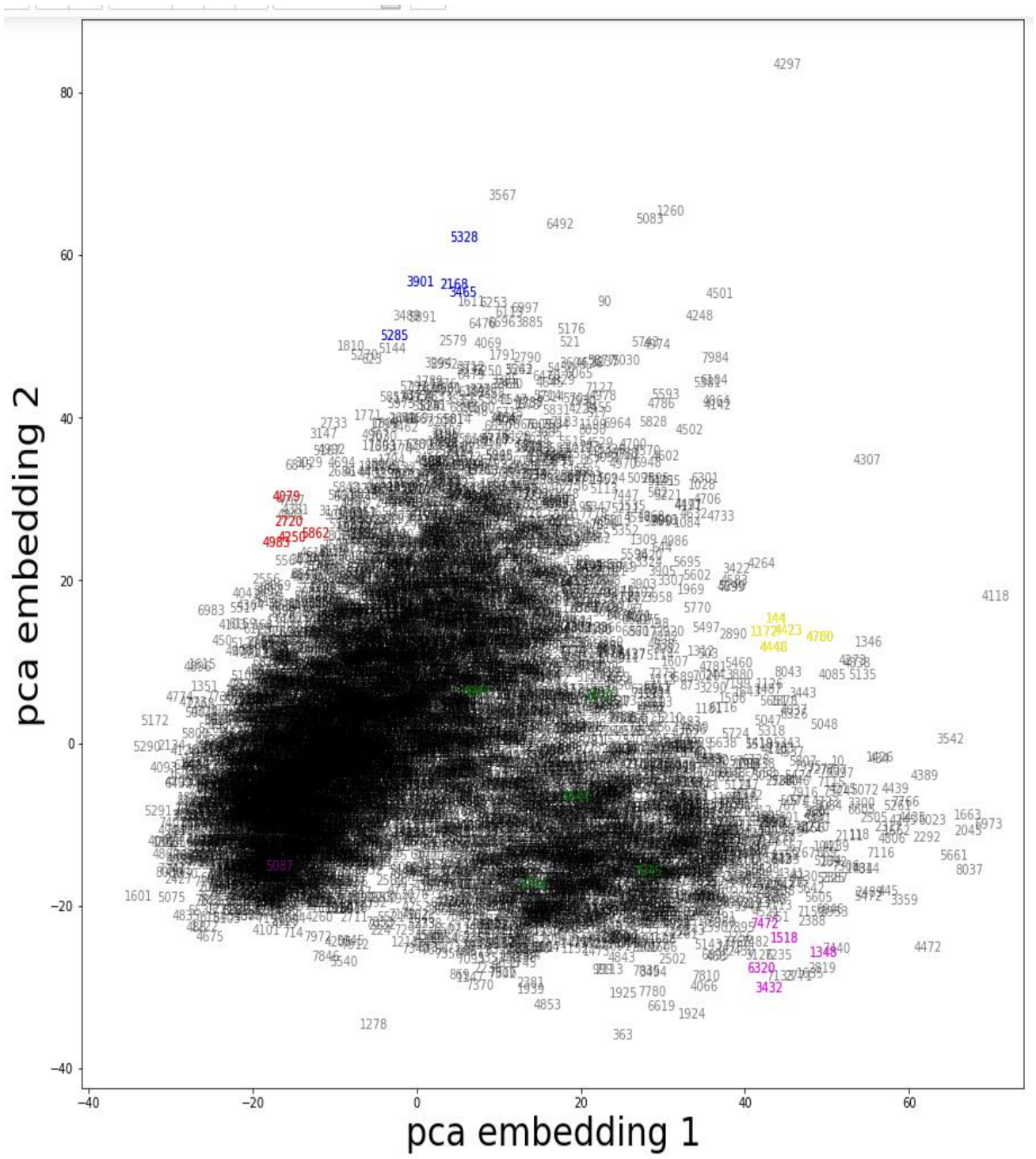


Fig 16: A visual of PCA embeddings

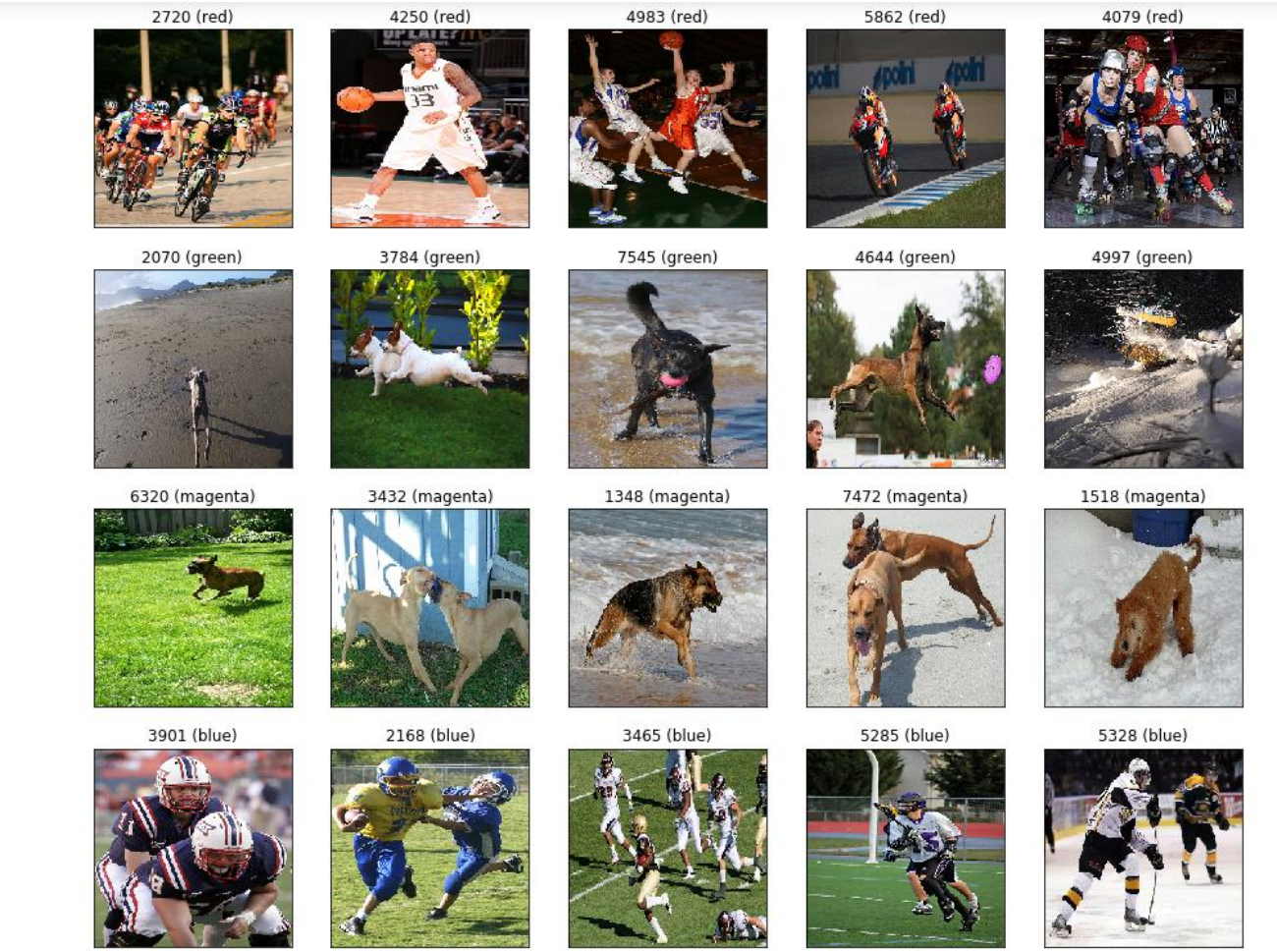


Fig 17: Plotting similar images from the dataset

```
4476
```

Layer (type)	Output Shape	Param #	Connected to
input_9 (InputLayer)	(None, 30)	0	
embedding_4 (Embedding)	(None, 30, 64)	286464	input_9[0][0]
CaptionFeature (LSTM)	(None, 30, 256)	328704	embedding_4[0][0]
dropout_4 (Dropout)	(None, 30, 256)	0	CaptionFeature[0][0]
input_8 (InputLayer)	(None, 4096)	0	
CaptionFeature2 (LSTM)	(None, 256)	525312	dropout_4[0][0]
ImageFeature (Dense)	(None, 256)	1048832	input_8[0][0]
add_4 (Add)	(None, 256)	0	CaptionFeature2[0][0] ImageFeature[0][0]
dense_7 (Dense)	(None, 256)	65792	add_4[0][0]
dense_8 (Dense)	(None, 4476)	1150332	dense_7[0][0]

```
Total params: 3,405,436
Trainable params: 3,405,436
Non-trainable params: 0
```

```
None
```

Fig 18: Building the LSTM model

```
Train on 49631 samples, validate on 16353 samples
Epoch 1/6
 - 463s - loss: 5.3759 - val_loss: 4.8776
Epoch 2/6
 - 460s - loss: 4.5156 - val_loss: 4.5810
Epoch 3/6
 - 455s - loss: 4.1009 - val_loss: 4.4676
Epoch 4/6
 - 456s - loss: 3.8300 - val_loss: 4.4382
Epoch 5/6
 - 460s - loss: 3.6239 - val_loss: 4.4181
Epoch 6/6
 - 460s - loss: 3.4465 - val_loss: 4.4253
```

Fig 19 Training the LSTM model

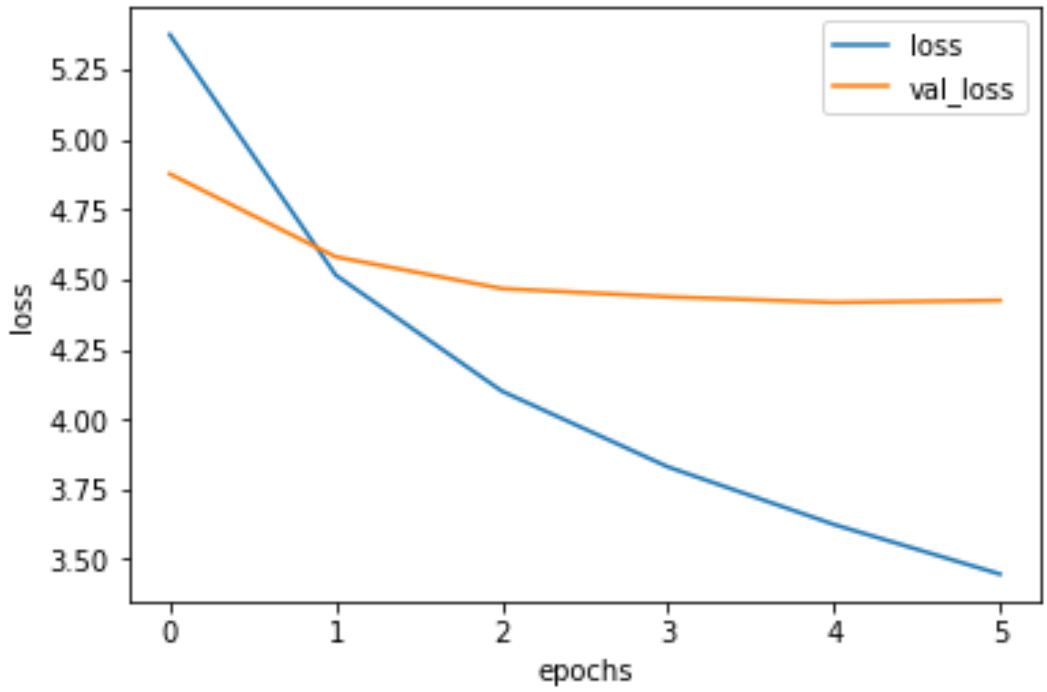


Fig 20: Epochs v/s loss

startseq man in blue shirt is standing on the street endseq

startseq black and white dog is running through the grass endseq

startseq brown dog is running through the snow endseq

startseq man in red jacket is riding the snowboard endseq

Fig 21: Sample captions generated for a small set of images

Bad Caption

true: little girl covered in paint sits in front of painted rainbow with her hands in bowl

pred: group of people are sitting in the street

BLEU: 0.2601300475114445

true: black and white dog is running in grassy garden surrounded by white fence

pred: brown dog is running on the grass

BLEU: 0.1744739429575305

true: collage of one person climbing cliff

pred: man in blue shirt is standing on the air in the air

BLEU: 0

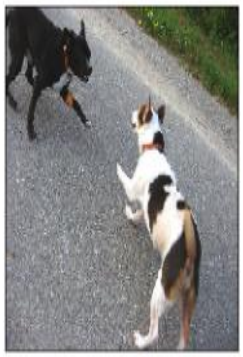
true: black and white dog jumping in the air to get toy

pred: dog is jumping in the grass

BLEU: 0.22083358203177395

Fig 22: Examples of bad captions generated

Good Caption



true: black dog and spotted dog are fighting

pred: black and white dog is playing in the grass

BLEU: 0.7598356856515925

true: man drilling hole in the ice

pred: man in blue shirt is jumping on the air

BLEU: 0.7598356856515925

true: man and baby are in yellow kayak on water

pred: man in blue wetsuit is playing in the water

BLEU: 0.7598356856515925

true: man and woman pose for the camera while another man looks on

pred: man in black shirt and blue shirt is standing in the street

BLEU: 0.7071067811865476

Fig 23: Examples of good captions generated

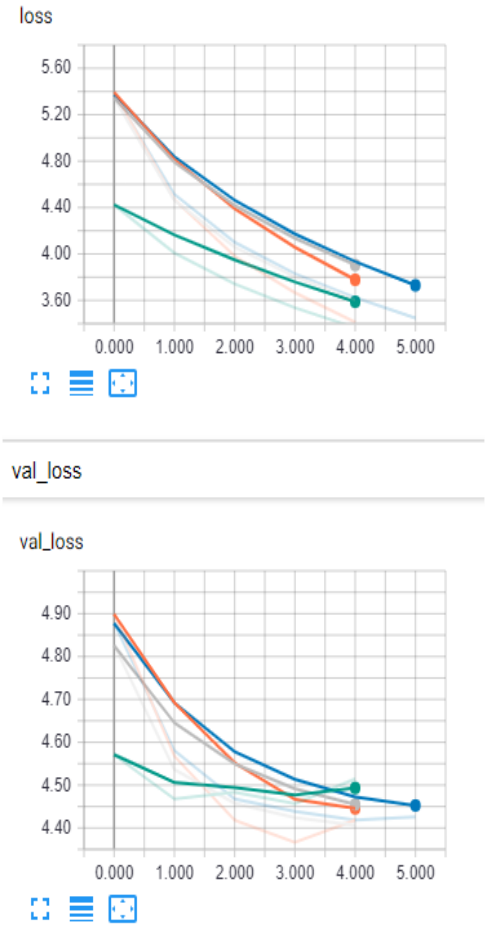
Conclusion

The model has been successfully trained to generate the captions as expected for the images. The caption generation has constantly been improved by fine tuning the model with different hyper parameter. Higher BLEU score indicates that the generated captions are very similar to those of the actual caption present on the images. Below you will find a table displaying different BLEU scores obtained by tuning the parameters:

1	A	B	C	D	E	F	G	Output
2	DEEP LEARNING MODEL	ACTIVATION FUNCTION	COST FUNCTION	EPOCHS	GRADIENT ESTIMATION	NETWORK ARCHITECTURE	NETWORK INITIALIZATIOI	Mean BLEU score
3	**Gradient Estimation**							
4	1	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.37
5	2	ReLU	Cross-Entropy	6	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.351
6	3	ReLU	Cross-Entropy	5	Adagrad	3 layer, 256 nodes, LSTM, vgg16	default	0.404
7	4	ReLU	Cross-Entropy	5	RMSProp	3 layer, 256 nodes, LSTM, vgg16	default	0.374
8	5	ReLU	Cross-Entropy	5	Adadelta	3 layer, 256 nodes, LSTM, vgg16	default	0.353
9	6	ReLU	Cross-Entropy	5	Nadam	3 layer, 256 nodes, LSTM, vgg16	default	0.353
10	7	ReLU	Cross-Entropy	5	SGD	3 layer, 256 nodes, LSTM, vgg16	default	0.028
11	**Cost Function**							
12	1	ReLU	mean_squared_error	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.215
13	2	ReLU	hinge	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0
14	3	ReLU	kullback_leibler_divergence	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.373
15	4	ReLU	cosine_proximity	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0
16	**Network Initialization**							
17	1	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	glorot_uniform	0.381
18	2	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	random_uniform	0.388
19	3	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	lecun_uniform	0.367
20	4	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	he_uniform	0.389
21	5	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	glorot_normal	0.398
22	**Activation Function**							
23	1	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.374
24	2	tanh	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.384
25	3	elu	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.392
26	4	selu	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.363
27	5	linear	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.192
28	6	sigmoid	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.375
29	7	softsign	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.396
30	8	softplus	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.381
31	**Epochs**							
32	1	ReLU	Cross-Entropy	3	Adam	3 layers, 256 nodes each	default	0.429
33	2	ReLU	Cross-Entropy	4	Adam	3 layers, 256 nodes each	default	0.394
34	3	ReLU	Cross-Entropy	5	Adam	3 layers, 256 nodes each	default	0.408
35	4	ReLU	Cross-Entropy	6	Adam	3 layers, 256 nodes each	default	0.38
36	5	ReLU	Cross-Entropy	7	Adam	3 layers, 256 nodes each	default	0.405
37	**Network Architecture**							
38	1	ReLU	Cross-Entropy	5	Adam	3 layers, 256 nodes each	default	0.407
39	2	ReLU	Cross-Entropy	5	Adam	3 layers, 128 nodes each	default	0.405
40	3	ReLU	Cross-Entropy	5	Adam	3 layers, 512 nodes each	default	0.394
41	4	ReLU	Cross-Entropy	5	Adam	4 layers, 256 nodes each	default	0.406
42	5	ReLU	Cross-Entropy	5	Adam	4 layers, 128 nodes each	default	0.386

Fig 24: Conclusive table created on google sheets

With the help of Tensorboard, we were able to see how different training process had an impact on the model.



The validation loss falls upto 5th epoch and then increases afterwards, while the training loss still continues falling

Fig 25: Epochs v/s validation loss curve

Tools and Technologies

Python: It is an interpreted, high-level, general programming language. It is powerful, fast and open source. With a rich support for tons of libraries and frameworks related to Deep Learning & ComputerVision.

Recurrence Neural Networks: A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior.

NLTK: Natural Language Toolkit is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an activediscussion forum.

Keras: Keras is a neural networks library written in Python that is high-level in nature – which makes it extremely simple and intuitive to use. It works as a wrapper to low-level libraries like TensorFlow or Theano high-level neural networks library, written in Python that works as a wrapper to TensorFlow or Theano.

Pickle: Python pickle module is used for serializing and de-serializing python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling.

Word Embeddings: An **embedding** is a relatively low-dimensional space into which you can translate high-dimensional vectors. Embeddings make it easier to do machine learning on large inputs like sparse vectors representing words. Word embeddings are computed by applying dimensionality reduction techniques to datasets of co-occurrence statistics between words in a corpus of text.

CONCLUSION & FUTURE SCOPE

Caption generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph. In this paper, we have implemented a deep learning approach for the captioning of images. It requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order. With the help of deep learning methods, in this report we have tried to achieve state-of-the-art results on examples of this problem. A single end-to-end model has been defined to predict a caption, given a photo, instead of requiring sophisticated data preparation or a pipeline of specifically designed models.

Thus, we would be able to develop an image caption generating system which with the help of RNNs and Computer vision, will prove to be a substantial tool to ease the work of content creators.

Image captioning has various applications such as recommendations in editing applications, usage in virtual assistants, for image indexing, for visually impaired persons, for social media, and several other natural language processing applications. These mentioned applications are going to be in high demand for the next couple of decades as anything and everything is taking an online and digital turn in the society which opens many doors for applications like these. Be it education, health, logistics, farming, spirituality; all of these sectors have demands and scope and the usage of image caption generator would benefit and ease the process as well as efficiency. Thus, with the goal of betterment of the future of society we have tried to bring an ease to the work of content creators by creating this image caption generator that would prove to be a substantial tool now and always.

REFERENCES

[1] Bernardi, Raffaella, et al. "Automatic description generation from images: A survey of models,datasets, and evaluation measures." *Journal of Artificial Intelligence Research* 55 (2016): 409-442.

[2] Tanti, Marc, Albert Gatt, and Kenneth P. Camilleri. "What is the role of recurrent neural networks(rnns) in an image caption generator?." *arXiv preprint arXiv:1708.02043* (2017).

[3] Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." *Proceedings of the IEEEconference on computer vision and pattern recognition*. 2015.

[4] Sharma, Grishma, et al. "Visual image caption generator using deep learning." *2nd InternationalConference on Advances in Science & Technology (ICAST)*. 2019.

[5] Mathur, Pranay, et al. "Camera2Caption: a real-time image caption generator." *2017 International Conference on Computational Intelligence in Data Science (ICCIDS)*. IEEE, 2017.

[6] Chen, Jianhui, Wenqiang Dong, and Minchen Li. "Image Caption Generator Based On Deep NeuralNetworks." *2015 Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2015.