Image Captioning with CNN and LSTM using Python

**Dr. A. Suneetha1, Dr.K.Ratna Babu2**

1Associate professor, Department of CSE, KKR & KSR Institute of Technology & Sciences

2 Lecturer in Computer Engineering Department, Government Polytechnic, Addanki

**Abstract** Our vision is our most vital sense. Software developers have utilized the capability of vision as they build more interactive, intelligent, and accessible software through images. However, there are scenarios where an image might not be sufficient alone. There might be extra context needed, or alternative text is displayed to circumvent bandwidth restrictions and provide a more accessible experience. In an era where there are simply a huge number of images to be described, the manual description fails at such a scale. Through the help of deep learning, image processing and natural language processing can combine to give way to empower the computer to describe images on its own. This can be presented as a consumable service through the use of a web frontend where users can simply provide the images they wish to be described. This allows anyone to leverage the power of this deep learning approach with relative ease and the use of adaptive image descriptor functionality via an easy-to-use API while the computationally heavy tasks will be abstracted away.

# Introduction

With the development of the electronic computer and the stored program computer the conditions for research in intelligent systems started. While exploiting the power of the computer systems, the curiosity of humans led them to wonder, “Can a machine think and behave like a human do?” Thus, the development of the Intelligent System started with the intention of creating similar intelligence in machines that we find and regard high in humans. After decades of research, Artificial Intelligence (AI) has arrived. It started to take over the work which cannot be done by humans. According to Wikipedia, “Artificial Intelligence is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals. Computer science defines AI research as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals.”

# Related Work

They propose to follow a refined way by restoring the encoder Network which was earlier employed by a deep CNN. From the previous years, it has been compellingly seen that CNNs can produce a great presentation of the input picture by inserting it to a rigid-length array, so that presentation can be

used for various vision tasks. So, it’s common to use a Convolutional neural network as a picture “encoder”, by initially training it for an image classiﬁcation work and later on using the last hidden layer as an input to the RNN decoder which can generate sentences. Neural Image Caption (NIC) they called this model.

Presented a robust system for the problem. Firstly, by using stochastic gradient descent a neural net is completely trainable. Secondly, the model combines latest sub-networks for language and vision models which are often pre-trained on larger corporations and thus can cash in more data. Finally, it produces comparatively good performance to modern approaches.

A probabilistic and neural framework can produce descriptions for pictures. Advancements in statistical machine translation(MT) is evident that, when we give a robust sequence model, it is possible to realize good results by directly increasing the probability of the right translation by giving an input sentence in an “end-to-end” fashion–both for training and prediction. These Models use a RNN that encodes the changing length input into a rigid dimensional vector and uses this presentation to “decode” it into a specified output. Thus, it is common to use an equivalent way in which, we give an image (rather than an input sentence in the source language), we apply the equivalent process of “converting” to its description

The foremost usual issue in building and training RNNs is to deal with this challenge of sequence generation and translation, a specific sort of recurrent networks called LSTM was used and applied with great success. Encoding knowledge at every time step of inputs which was observed up to the present step is the key part of LSTM memory cells. The “gates” are used to control the behavior of the cells – layers that are put in repeatedly and this can either keep a worth from the gated layer if the gate is one(1) or zero(0). In particular, three gates are being used which control if it should read its input (input gate) , whether to forget the present cell value (forget gate), and to output the new cell value.

# Bottom-up and Top-Down

When combining language understanding and image like visual question answering (VQA) and image captioning kept on inspiring comparatively

large research at the edge of NLP and computer vision. In both of them, to get good quality results it is regularly required to perform some compact visual processing or even various steps of reasoning. Consider an image I, both visual question answering and image captioning model, take as input a possibly varying-sized set of n image features, X = {v1,...,vn}, vi ∈ RD, such that every picture feature encodes an important portion of the picture. The output of the bottom-up model is the spatial picture features X. For the top-down attention component, given models use simple one- pass attention mechanisms, rather than more complex schemes of recent models such as bidirectional, multi-headed or stacked which might also be applied.

The sense of spatial picture characteristics X is general. But, here they constructed bottom-up attention using Faster R-CNN and deﬁned spatial parts in the form of bounding boxes. They used Faster R-CNN as an object detection model to spot occurrences of objects corresponding to a specific class and localize them using bounding boxes. An attentive mechanism is employed to train different region proposals. Two different stages are required for a Faster R-CNN to detect objects. The initial stage, predicts object proposals, defined as a Region Proposal Network (RPN). Intermediate levels of a CNN are used to slide over features of a tiny network. Class-agnostic objectness score is predicted at various spatial locations, and a bounding box refinement for anchor boxes of various aspect ratios and scales. The highest box proposals are selected as input to the second stage using greedy non-maximum suppression with an intersection over union (IoU) threshold. In the second stage, A little feature map(e.g. 14×14) for every box proposal in the region of interest (ROI)is extracted using pooling. The feature maps obtained are clubbed together as input to the last layers of the Network. Softmax distribution over class labels and the class specific reﬁnements bounding boxes are the ﬁnal outputs of the model.

This captioning model uses a ‘soft’ top-down attention mechanism to weight each feature during caption generation when given a set of picture features X, using the current partial output sequence as context. The captioning model is divided into two LSTM layers at a high level, using a standard implementation. There is the Long short term memory (LSTM) layer as a top-down visual attention model, within the captioning model. Whereas the second LSTM layer as a language model which indicates every layer with superscripts in the equations. The input array to the attention LSTM at each time step consists of the previous output of the language LSTM, added with the mean-pooled picture feature.

# Existing System

Humans are the first and best choice to describe an image in any circumstances, but in reality there are a huge number of images created and uploaded everyday, who will describe them. Here the ability and energy of humans reach its extreme limits. Whenever a human reaches his limit there arises a technology to overcome the problem, the solution comes in the form of Deep Learning (DL) technique, an important component of Artificial Intelligence (AI).

In the existing system like show and tell they leveraged the power of the Artificial Neural Network (ANN) which replicates the behavior of the human brain to produce good results. Convolution Neural Networks (CNN) are being used to replicate the work done by humans such as image description. CNNs are vastly used for image processing due to their ability to produce high accuracy, whereas Recurrent Neural Networks(RNN) are used for text and speech processing due to their ability to process sequence information with ease.

As observed in the Bottom-Up and Top-Down approach where we need to pose a question to the system to produce an appropriate description, this process of questioning and answer generation is called Visual Question Answering (VQA). But independence and self-reliance is expected from the system as humans can't keep posing questions to each and every image to produce good quality descriptions.

# Challenges to be implemented

One of the main challenge is Dumb Captions, the services use generic and vague descriptions as their base. The captions generated are hit-or-miss. Proprietary algorithms and code mean that we cannot be sure of how the brain of a descriptor works. This lessens its robustness. The system should be self-starting. It should automatically caption the image supplied to it without any external catalyst.The system should try multiple possible captions and decide upon the best one. We wanted to overcome these challenges in the proposed system.

# Proposed System

When we analyzed the problem and observed how the existing system tackled it, we realized that we could improvise and devise a new solution to the problem to yield better results. The earlier methods of solving the matter were revolutionary in their approach. When its contemporaries checked out the matter as a language problem, Show and Tell

presented the ingenious idea of leveraging the advancements brought by transfer learning in image processing. They were correct: Image captioning may be a task which belongs to 2 areas, Image Processing and tongue Processing. By focusing their attention separately on these two areas, they were ready to leverage the simplest approaches for both and combined them together to great effect. The hallmark of their approach was the use of a Convolutional Neural Network (CNN) for the image processing portion to yield the input for the latter Recurrent Neural Network (RNN) based text generation. CNNs underwent a big research process and analysis which made them best in school for tackling visual problems. As images are visual media too, the CNN they deployed in their approach performed exceedingly well and people improvements helped the RNN generate better captions.

It wasn’t without its flaws. Their image processing advancements cause better entity detection which improved the standard of the captions generated. But the caption generation still relied on the generic information. This meant that a lot of different captions would be generated with similar levels of accuracy and it made selection of an “appropriate” caption hard. But, their approach was so influential that future methods built on their core idea of utilizing a CNN and RNN combination model. Bottom-Up and Top-Down was one such new approach. They identified the matter of unfocused captions faced by Show and Tell and devised an innovative solution. The idea of their new approach was Visual Question Answering (VQA). because the system functioned only when an issue was posed thereto , it had priori knowledge of what the caption, or answer, would be. The scope of search was vastly narrowed down and this resulted in far better and succinct answers. This reliance on a context specific question gave it a foothold but also deterred its usage in large scale scenarios. The trouble spent constructing an issue to the model could instead wont to describe the image itself.

We decided to apply another deep learning to make our system self-starting whilst providing high quality captions at the same time. To do this, we opted for Generational Adversarial Networks (GANs). The idea behind GANs is that there are two components in the machine learning process: A generator and a discriminator. It is the job of the generator to process the provided input and produce the output which is expected of the system. Meanwhile, the discriminator acts as a preemptive judge of the output. Rather than providing the generated output straightaway to the user, the discriminator takes the rule of a user and acts as a pseudo user. It uses the information available to it via the problem to determine if the output generated

by the generator is satisfactory. If the output generated by the generator is deemed to be satisfactory by the discriminator, it is allowed to pass onto the user.

As the GAN augmented model is trained for a few epochs, the generator gets better at producing the output as the discriminator forces it to keep improving. At a similar instance of temporal time, the discriminator becomes smarter at identifying less satisfactory output. This dual improvement process is one the key reasons for the ability of GANs to produce high quality results in machine learning scenario. Hence, the gains of the previous methods can be merged and combined into a single system to propose a new system which uses the existing system as a base and undergoes an evolution to yield a proposed system which can yield better results as it is able to address the drawbacks of the existing systems due to the evolution it undergoes.

# Methodology

* 1. **Problem Scope**

The earlier methods of solving the problem were revolutionary in their approach. When its contemporaries looked at the problem as a language problem, Show and Tell presented the ingenious idea of leveraging the advancements brought by transfer learning in image processing. They were correct: Image captioning is a task which belongs to two areas, Image Processing and Natural Language Processing. By focusing their attention separately on these two areas, they were able to leverage the best approaches for both and combined them together to great effect. The hallmark of their approach was the utilization of a Convolutional Neural Network (CNN) for the image processing portion to yield the input for the latter Recurrent Neural Network (RNN) based text generation. CNNs underwent a significant research process and analysis which made them best in class for tackling visual problems. As images are visual media too, the CNN they deployed in their approach performed exceedingly well and those improvements helped the RNN generate better captions.

It wasn’t without its flaws. Their image processing advancements lead to better entity detection which improved the qualitative aspects of the captions generated. But the caption generation still relied on the generic information. This meant that many, different captions would be generated with similar levels of accuracy and it made selection of an “appropriate” caption hard. But, their approach was so influential that future methods built on their core idea of utilizing a CNN and RNN combination model. Bottom-Up and Top-Down was one such new approach. They identified the problem of

unfocused captions faced by Show and Tell and devised an innovative solution. The basis of their new approach was Visual Question Answering (VQA). As the system functioned only when a question was posed to it, it had priori knowledge of what the caption, or answer, would be. The scope of search was vastly narrowed down and this resulted in much better and succinct answers. This reliance on a context specific question gave it an edge but also deterred its usage in large scale scenarios. The effort spent constructing a question to the model could have been instead used to describe the image itself.

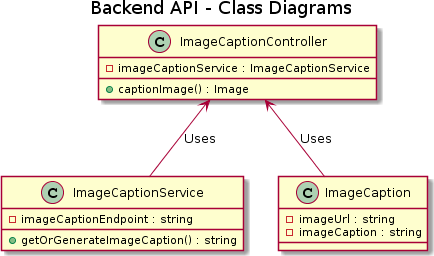


Figure 1. Backend API

The backend API is built with dependency injection (DI) in mind. Going for a DI based implementation allows for loose coupling of the components involved. As long as the different components hold true to the interface or contract they specify, its dependents can function without worrying about its internal dependencies. The usage of DI also means that unit testing can be carried out by mocking the dependencies of components in order to isolate the component and test the component under observation.

# ML Module

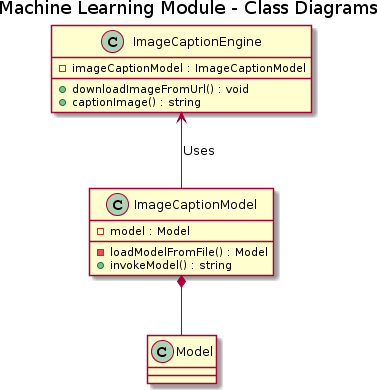


Figure 2. ML module

The machine learning module is the component which generates captions for images. At the core of this module lies the machine learning model we developed to generate caption to a given image. Part of the machine learning model’s productionization, the training of the model is done for many epochs or iterations until satisfactory results have been obtained. After the training is

done, the model is saved to disk in order to skip the whole training process whenever an image has to be captioned. When an image caption request is received, the model is loaded from the disk and inference is performed using the saved model. By leveraging this technique, the model runs significantly faster. The pre-trained and saved model is denoted by Model.

# Frontend

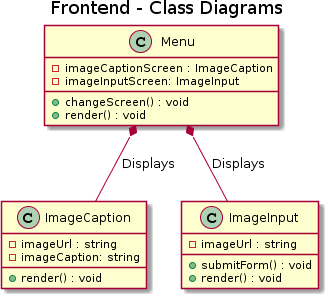


Figure 3. Front End

The classes present in the frontend are primarily React functional components which handle the presentation of the UI to the user and communicating with the Backend API via HTTP requests. The Menu class is the origin of the interaction of the user with the system. It controls the user interaction flow and is responsible for changing the screen being displayed to reflect the user’s chosen option. To do this, it holds references to ImageCaption and ImageInput screens and injects data into them and calls their render method when the screen is changed. All these classes have the render method present as that is how React knows to display them on the screen. The changeScreen method handles the task of displaying the appropriate screen dependent on the user’s choices.

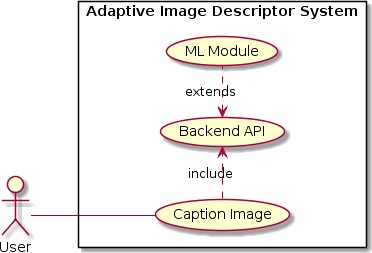


Figure 4. Captioning Image process

# Implementation

* 1. **Architecture**

The hallmark of an asynchronous architecture is the non-linear fashion in which the components of the system interact with one other. The components in this architecture are also known as services.

Services communicate with each other through asynchronous means, most of the time they don’t receive an immediate response but rather are notified that their request has been accepted. After the request has been completely processed, the result is put into a shared database or message queue and it is retrieved. Message queues are one of the most common means of communication in this architecture. A message queue stores and transfers the messages. Services can subscribe to a channel on the message queue and they can then push messages to this channel or retrieve messages which were pushed to the channel. A channel forms a point of contact for different services to communicate with one other.

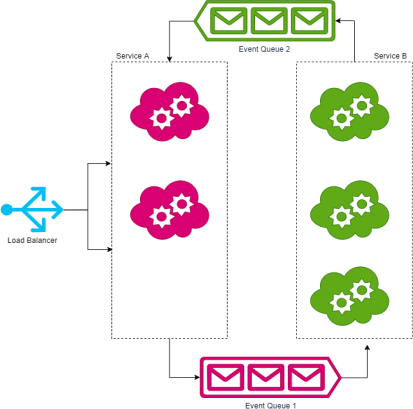


Figure 5. Process Architecture

# Specific Architecture

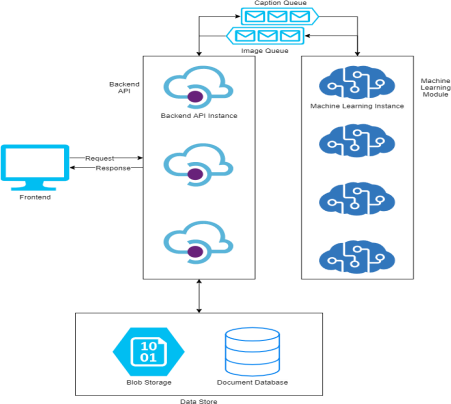


Figure 6. Specific Architecture

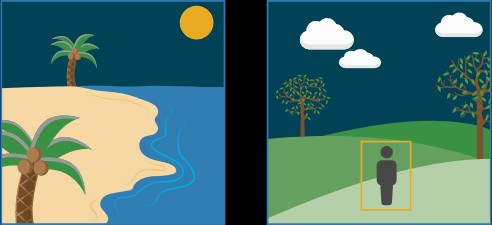


Figure 7. Object recognition (left) and object detection (right).

Object Recognition Using Deep Learning- Deep learning (DL) techniques have become a well-liked method for doing object recognition. DL models like convolutional neural networks (CNN), are used to automatically learn an object’s inherent characteristic so as to spot that object. For instance, CNN can learn to spot contrast between cats and dogs by analyzing thousands of training pictures and learning the features that make them different. To train a deep network from scratch, one should

gather a awfully large labeled dataset and style a network architecture that may learn the features and construct the model. The results may be impressive, but this approach requires a huge amount of training data, and one needs to line up the layers and weights in the CNN. DL provides high accuracy but requires a huge amount of data to make precise predictions.

Neural network we used is the Convolutional neural network to train the model. It can be considered as few of the powerful supervised deep learning techniques. The concluding form of a CNN is quite similar to Regular Nets (a regnet is a Regular Neural Network), which contains neurons with weights and biases. In inclusion, just like in Regular Nets, we have to use a loss function like cross entropy or softmax and an optimizer like adam optimizer for CNNs. In CNNs, there are Pooling Layers, Flatten Layers and Convolutional Layers. CNNs are mainly used for image classification although you may find other application areas such as natural language processing, digital signal processing (Audio processing). In Regular Nets all the neurons are connected to each other which is the main structural feature. For instance, most images have a large number of pixels and not even grey-scaled. So, assuming that we have a set of images in Ultra HD 4K, we will have 26,542,080 (4096 x 2160 x 3), as different neurons are connected to one another in the very first layer and this becomes unmanageable. So, we can conclude that Regular Nets don’t work as required for image classification. Nevertheless, when we come to image, there looks like to be a small correlation or relation between two different pixels unless they are very close to one another. This pilot to the idea of Pooling Layers and Convolutional Layers.

We can make use of many different layers in CNN. But, pooling, convolution, ReLU layer, Loss layer and fully connected layers are some of the layers.

Feature extraction is done in the very first layer which is Convolutional Layer from the pictures in our dataset. It's because of the certainty that pixels are closely linked with the adjacent ones, convolution helps to store the relation between different segments of a picture. Convolution is filtering the image with a smaller pixel filter to reduce the size of the image without losing the relationship between the pixels. When we perform convolution to a 5x5 image with a 3x3 filter with 1x1 stride (we can shift 1 pixel at each step). We can reduce it to a 3x3 output (65% reduction in complexity).

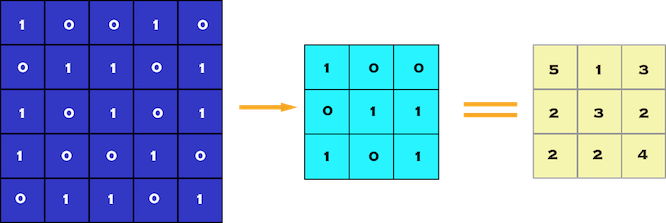


Figure 8(a). Convolution of 5 x 5 pixel image with 3 x 3 pixel filter (stride = 1 x 1 pixel)

In CNNs, it is quite common to insert pooling layers after every convolution layer to decrease the spatial size of the presentation to decrease the parameter numbers which in turn decreases the computational complexity. And more, over fitting problems can be solved with the help of pooling layers. Here to reduce the amount of the parameters we select a pooling size by selecting the average, maximum, or sum values inside pixels. One of the most common pooling techniques, Max Pooling**.**

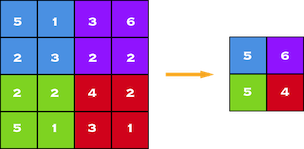


Figure 8(b). Polling of 5x5 to 3x3

Images are non-linear by nature, we apply RELU to increase the non-linearity which consists of a rectifier function.

Loss Layer-This layer shows how training results in the deviation between the true labels and predicted (output) which is generally the last layer. For different tasks various loss functions may be used. Flattering is done after pooling. The

reason we do this is that we're going to need to insert this data into an artificial neural network later on

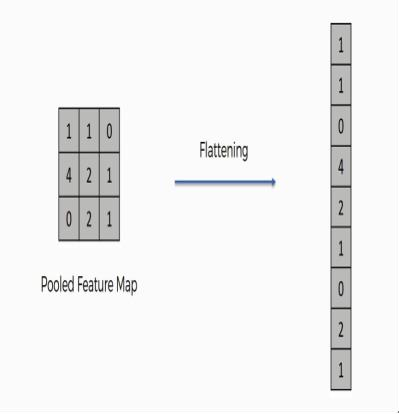


Figure 9. Flattening of feature map

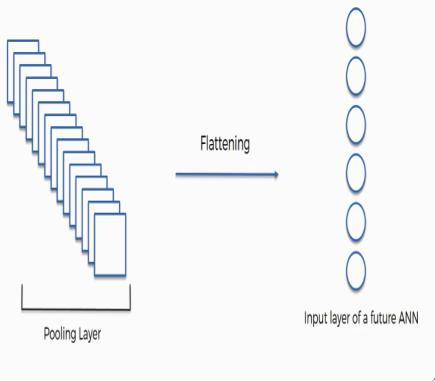


Figure 10. Flattening of pooling Layer

As you can see from the above figure, after the flattening step the result obtained is the long vector that can then be passed through the network to have it further processed.

The very last layer of a neural network is the Softmax activation layer rather than others like tanh, ReLu or sigmoid activation functions. As this layer possesses the capability to convert the output into essentially the probability distribution, this layer is placed at last.

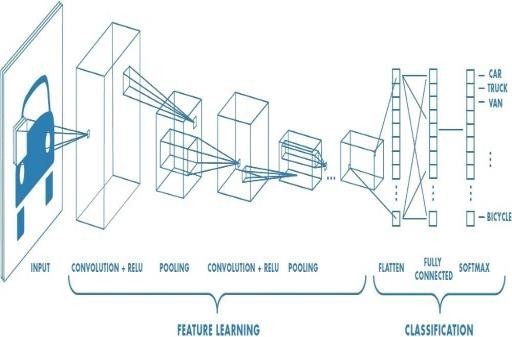


Figure 11. Overview of CNN

Natural language Processing-NLP can be defined as the automatic handling of the language we generally use, like speech and text, by a computer. In other ways NLP could be a collective term

bearing on automatic computational processing of language we generally use. Which consists of both algorithms that take text as input (training data set), and algorithms (models) that produce genuine looking text data as outputs (result).

Natural language means the way we communicate with one another. Like speech and text. We are associated by text. Give some thought to what proportion of text you find every day: Signs, Menus, Email, SMS, websites and the list is endless…

Now give some thought to speech. We communicate with each other, as humans, over what we write. It is easier to be told to talk than to put in writing. Text and voice are how we communicate with one another. As this sort of data is important, we must have ways to know, understand and reason about it, the same which we do for all other data.

Recurrent Neural Network (RNN)-RNNs are a awfully critical version of artificial neural networks (ANN’s) hugely used in NLP. RNNs are a kind of ANN that adds extra weights to the network to make a cycle within the network graph to take care of the inside state. This adding of state to NNs promises that they will be able to externally learn and exploit context in linear prediction problems, like problems with a sequence or a temporal component. They differ a bit from a customary neural network because the input in an RNN is a word rather than the sample which is in the general way of a standard NN. This offers the elasticity for the network to work with differing sizes of words and sentences, whereas this can’t be achieved in a standard NN due to its rigid structure. It also provides an added edge of splitting features learned from different positions which can’t be seen in a customary NN.

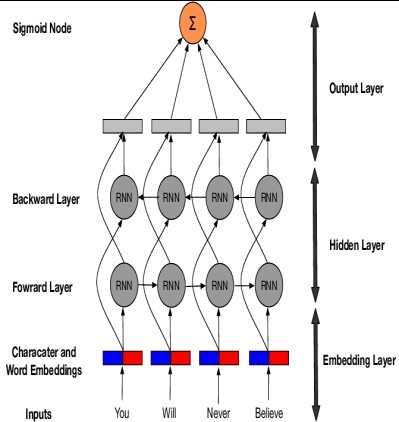


Figure 12. RNN process

**Long Short-Term Memory Cell (LSTM) -** LSTM comes under recurrent neural networks which are having the capacity of learning order dependence in sequence prediction problems. This is required in complicated problem domains like Natural

Language Processing (NLP), Machine Translation (MT), speech recognition technology, and more.

Though an RNN can learn dependencies but, it can only find out about most recent data. LSTM can counter the problem, because it can understand content together with contemporary dependency. Hence, Long Short-Term Memory Cells are a special kind of recurrent neural networks where understanding comes handy. Long Short-Term Memory Cell networks are similar to recurrent neural networks with one critical contrast, that is, hidden layer updates are restored with memory cells. This provides better performance at recognizing and exposing long range dependencies in which data is vitally important for sentence. Image below shows an LSTM sequence tagging model which has memory cells in position of hidden layer.

**LSTM (Long Short Term Memory)**: Long Short Term Memory (LSTM), popularly referred to as LSTMs, is special variant of RNNs that are capable of understanding distant dependencies in a word sentence. The architecture of long-term memory (LSTM) is anything but different from that of conventional RNNs. The working of an LSTM is shown in below figure [14].

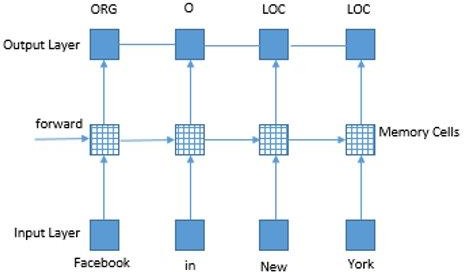


Figure 13. LSTM process

The essential building blocks and functions of an LSTM are as follows: The latest feature in LSTMs is the addition of three gates that control the cell state Ct, and the gates are composed of sigmoid functions that give output values from 0 to 1.

At sequence step t the hidden state of the input xt and the preceding step ht-1 determine the information to be forgotten via the forget-gate layer. The forgotten gate looks at xt and ht-1 and assigns 0 to 1 in the Ct-1 cell state vector for each variable. A0 output means that the state is completely missing whereas A1 output means that the state is complete. The forget gate performance is determined in the following manner:



Then, the input gate determines which cell units should be modified with new details. To this end,

like the forget-output, the following logic helps determine a value between 0 and 1 for each part of the cell state:

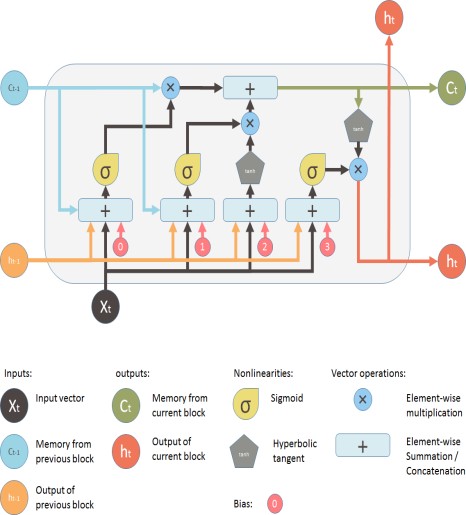


Figure 14. LSTM Architecture

Instead, it generates a candidate set of new values using xt and ht-1 as input for the cell state. Nominated cell state Ct shows the calculations as follows:



The next task is to decide which cell states to emit, because the cell state has a lot of information. For this reason, an output gate which produces a value between 0 and 1 for each part in the Ct is passed through xt and ht-1. The output of that gate is calculated according to:



The modified hidden h, is determined by passing each of its elements through an activation function tanh from the cell state Ctand then using the output gate values to do an element-wise product.



Keep in mind that the sign \* in the previous equations indicates multiplication in element-wise terms. It's done so we can allocate weights based on the gate outputs to every dimension of the vector on which it is being worked. Also note that whatever values are obtained from the output of the gate, they are multiplied as is. Do not give discrete values of 0 and 1 to the gate outputs.

A Continuous values between 0 and 1 by an activation function sigmoid are given smooth gradients for back propagation.

In the LSTM, the forgotten gate plays a crucial role. When the forgotten gate units output zero then the

repeated gradients get null and the corresponding old cell state units are discarded. That way, the LSTM throws away information that it thinks won't be valuable in the future. Also, when the forgotten gate units are output 1 the error flows unsupervised through the cell units, and the model can learn to associate long distances between temporally distant terms.

Some other essential element of the LSTM is that of having to add gates for output. The output gate unit helps to ensure that not all of the Ct cell state units ' info is revealed to the rest of the network and that only the necessary information is released in the form of ht. This means that the unused data does not affect the rest of the network, as cell-state data is still retained to help in future decisions [14].

# Results and Discussion

The .csproj file is the file which declares all the libraries and files required and used by the project. The backend utilizes the NuGet packages and utilizes their provided functionality. The NewtonsoftJson packages provide advanced JSON functionality. It is used in the project to provide serialization and deserialization of the data in JSON format. The JSON data is deserialized into POCO (Plain Old C# Object) classes and the POCO classes are serialized into JSON data. This is because the backend provides a JSON based API. The ObjectPool extension is used to prevent a large number of allocations. Although ASP .NET Core is backed by the CLR which is efficient at performing garbage collection, one of the best ways to improve the application’s performance is to prevent the allocation of extra and unneeded objects. The ObjectPool package allows us to define a pool of

objects which are created once. Whenever a particular object is required, it is taken from the

pool. After the object is finished bei(n5g) utilized, it is

returned back to the pool. By doing this, we avoid extra allocations. The MongoDB Driver is the package which allows us to communicate with MongoDB, our database of choice. By using the driver, we are able to establish a connection with the database. After establishing the connection, the driver helps us obtain a handle to the collections of the database. This handle allows operations which manipulate the collection.

The Startup class is the main entrypoint of the backend. Here, all the services used are declared and configured. The ConfigureServices method is in-charge of configuring all the services of the backend. The Configure method handles the actions to be carried depending on the configuration of the system. This class also initializes the singletons used by other classes of the system. It carries out the dependency injection and thus it takes the DependencyInjection package. After the Startup

class has initialized, the backend is ready to service requests.

As the project makes extensive use of dependency injection, one of the omnipresent features is the use of interfaces to separate the concerns of the project from its implementation. THe interface based approach works as long as the implementation being used fulfills the contract specified by the interface. This approach makes it easy to swap out implementations.. It also helps in unit testing as the interface can be easily mocked compared to an entire class. Here, the ICaptionImageService declares two methods. The first method is GetCaptionImageByIdAsync which is declared by ICaptionImageService. It takes a string parameter of the image URL as input and returns a CaptionImage object as the output. The CaptionImage object is a model which encapsulates the data about a captioned image. The method name is suffixed with “Async” and this indicates that the method should have an asynchronous implementation. This makes sense because the process would involve communicating with a database which is an IO process and hence should be non-blocking by making it asynchronous. The asynchronous nature of the method is further exemplified by the fact it returns a CaptionImage object within a Task object. A Task object encapsulates information for asynchronous operations in .NET. The second method declared by ICaptionImageService is SaveImageToDatabase. It takes a string parameter of the image URL as input and the caption associated with the corresponding image and returns a Task result. A Task object encapsulates information for asynchronous operations in .NET.

The CaptionImageService is a service class which facilitates interaction with the database database and the captioning service. It implements the ICaptionImageService interface and is thus associated with this interface as a singleton in the Startup class. Any class which wants to utilize the CaptionImageService class can take a dependency of the ICaptionImageService interface in its constructor and the dependency injection component will provide the appropriate instance to the interface which is taking ICaptionImageService as a dependency. The responsibilities of the CaptionImageService class include the task of facilitating interactions with the MongoDB database and fulfilling the contract specified by the ICaptionImageService interface. The CaptionImageService class maintains a reference to the MongoDB collection of the images and their captions in the \_CaptionImages field. The interactions with the database are facilitated by

\_CaptionImages which is of type MongoCollection over CaptionImage class. MongoCollection is an

interface provided by the MongoDB driver which enables typed, generic, collections of objects to be created. The methods provided by the MongoCollection have asynchronous variants and this makes sense as most of these methods involve IO communication with the database. As such, CaptionImageService class too provides asynchronous methods.

The CaptionImage class is a model class which encapsulates the response of the API supported by the backend. By encapsulating all the data the client receives into a simple, coherent class, a unified and resilient response can be provided to the client. As the database in the system is MongoDB, the class uses the data serialization features provided by the MongoDB driver. The data format used by MongoDB is BSON, a binary, JSON-like format. The fact that the data is stored in a binary format means that less space is used.

# Preprocessing the data for training RNN model

If the image is successfully captioned, the backend API returns a response to the frontend. The user can go to my captions screen to view all their captions requests. It is expected that the user’s previous caption request will appear here if the image was successfully captioned.

# Test cases and Result Analysis

The level of the system’s correctness was assessed as part of the end-to-end testing phase. We wanted to assess how the system performed on the input of a real user. The system’s performance meant how the system generated captions for the images provided to it. We took a few images and designed a template for assessing the system’s performance. The system was assessed based on the captions it generated for the images we supplied to it. The following results were obtained.

Image #1

Expected: A caption which signifies that a game of football is being played.

Actual: “A group of young men playing a game of football”

Result: Successful

Observations: The system was able to successfully generate an approximate caption for the image. Moreover, it was able to make out the group of men involved were young and it included this descriptive term in the generated caption.

Image #2

Expected: A caption which describes the horse. Actual: “A brown horse standing on a grass covered field”

Result: Successful

Observations: The system was able to successfully generate an approximate caption for the image. Moreover, it was able to discern the secondary details present in the image such as the field of grass. It added this information into the generated caption to make it more informational.



Image #3

Expected: A caption which describes the cat. Actual: “An orange cat lying down.”

Result: Successful

Observations: The system was able to successfully generate an approximate caption for the image. The lack of particular details in the image meant that the system couldn’t generate a more descriptive caption like it was able to do for the previous images. One improvement which can be identified here is that we could teach the system about the different breeds of cats so that it describes the cat’s breed as part of the caption.



Image #4

Expected: A caption which describes the group of people.

Actual: “A group of people posing for a photo” Result: Successful

Observations: The system was able to successfully generate an approximate caption for the image. Moreover, it was able to discern the contextual details present in the image such as the fact that the group of people here were posing for a photo. It added this information into the generated caption to make it more informational.



Image #5

Expected: A caption which describes the traffic. Actual: “A busy street filled with lots of traffic” Result: Successful

Observations: The system was able to successfully generate an approximate caption for the image. Moreover, it was able to discern the secondary details present in the image such as the amount of vehicular traffic present. It added this information into the generated caption to make it more informational.



Image #6

Expected: A caption which describes Xi Jinping. Actual: “A man in a tuxedo standing for a photo.” Result: Unsuccessful

Observations: Here, we can observe one of the shortcomings of the system. The system is able to correctly guess to an extent that the image is that of a man in a tuxedo. However, one of the most important details of the image is that the man in the photo is the president of China, Xi Jinping. This is an area of improvement for the system. If we are able to teach the system about specific individuals, it would be able to identify these individuals in the images and generate captions which more accurately capture the intent of the image.



# Conclusion

While working on this algorithm, we implemented an image captioning model from scratch, and how profound learning can be useful to produce correct captions in natural language such as English.

We implemented our algorithm on Flickr8k dataset and eventually we were able to generate captions with moderate accuracy. We also concluded that larger the dataset more will be the accuracy and less will be the losses.

Automatic captioning of images is a fairly new job, and great progress has been made thanks to the efforts of researchers in this area. In our opinion there is still plenty of space for improving image captioning efficiency. Firstly, the rapid development of deep neural networks would certainly boost the efficiency of image description

generation by employing more efficient network architectures as language models and/or visual models. Secondly, since images consist of objects distributed in space while image captions are sequences of words, it is necessary for image captioning to examine the existence and order of visual concepts in image captions. It would also be interesting to work into solving image captioning problems in different special cases.

We have performed our research using Flickr8k dataset. Whereas, for further enhancement in future we can use more and more larger datasets such as Flickr30k or MSCOCO so as to get more accurate captions.

# References

1. Wu Y, et al. (2016) Google’s neural machine translation system: Bridging the gap between human and machine translation CoRR abs/ 1609.08144.
2. Meltzoff A, Kuhl P, Movellan J, Sejnowski T (2009) Foundations for a new science of learning.325:284-8.
3. Hinton G, et al (2012) Deep neural networks for acoustic modeling in speech recognition: The shared view of four research groups. *IEEE Signal Processing Magazine* 29(6):82-97.
4. Mishra J, SahaI (2010) Artificial neural networks in hardware: A survey of two decades ofprogress. *Neurocomputing* 74(1):239-255. Artificial Brains.
5. Maier HR, Dandy GC (2000) Neural networks for the prediction and forecasting of waterresources variables: a review of modelling issues and applications. *Environmental Modellingand Software* 15(1):101-124.
6. Bhute AN, Meshram BB (2014) Text based approach for indexing and retrieval of image andvideo: A review. *CoRR* abs/1404.1514.
7. Karpathy A, Fei-Fei L (2017) Deep visual- semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell*. 39(4):664-676.
8. Vinyals O, Toshev A, Bengio S, Erhan D (2014) Show and tell: A neural image caption generator. *CoRR* abs/1411.4555.
9. Lipton ZC, Kale DC, Elkan C, Wetzel RC (2015) Learning to diagnose with LSTM recurrent neural networks. *CoRR* abs/1511.03677.
10. Sharma, Grishma, PriyankaKalena, Nishi Malde, Aromal Nair, and SaurabhParkar. "Visual Image Caption Generator Using Deep Learning." *Available at SSRN 3368837* (2019).
11. Tan, Ying Hua, and CheeSeng Chan. "Phrase- based image caption generator with

hierarchical LSTM network."

*Neurocomputing* 333 (2019): 86-100.

1. Alahmadi, Rehab, Chung Hyuk Park, and James Hahn. "Sequence-to-sequence image caption generator." In *Eleventh International Conference on Machine Vision (ICMV 2018)*, vol. 11041, p. 110410C. International Society for Optics and Photonics, 2019.
2. Hani, Ansar, NajibaTagougui, and MonjiKherallah. "Image Caption Generation Using A Deep Architecture." In *2019 International ArabConference on Information Technology (ACIT)*, pp. 246-

251. IEEE, 2019.

1. Ballal, Noopur, and Sri KhetwatSaritha. "A Study of Deep Learning in Text Analytics." In *Social Networking and Computational Intelligence*, pp. 197- 205. Springer, Singapore, 2020.

Dr. Akula Suneetha received her B.Tech (CSE) from Sir C R Reddy College of Engineering, Andhra University in the year 2005 and M.Tech (CSE) Degree from Jawaharlal Nehru Technological University

Hyderabad, Telangana, India in the year 2009 .She is received Ph.D. Degree in Acharya Nagarjuna University in 2022. She is currently working as Associate professor in the Department of CSE, KKR & KSR Institute of Technology & Sciences, Guntur, Andhra Pradesh affiliated to JNTUK University. Her research area is Image Processing. She is having 19 years of Teaching Experience.

**Dr K.RatnaBabu** received his Ph.D (Computer Science and Engineering) from Jawaharlal Nehru Technological University Hyderabad, Telangana, India in the year 2015, M.Tech (Software Engineering) Degree from

Jawaharlal Nehru Technological University, Hyderabad, Telangana, India in the year 2003 and B.Tech (Computer Science and Engineering) from Bapatla Engineering College, Nagarjuna University, Andhra Pradesh in the year 2000. He is currently working as Lecturer in Computer Engineering Department, Government Polytechnic, Addanki, Andhra Pradesh. His research interests are image processing and pattern Recognition. He has published more than 15 papers in International & National Journals and conferences. He is having 20 years of Teaching Experience. He is a life member of Technical Association ISTE.