A Project Report

on

IMAGE CAPTIONING USING MACHINE LEARNING AND DEEP LEARNING

Submitted in partial fulfillment of the
requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

Under The Supervision of
Mr. V. Arul
Assistant Professor
Department of Computer Science and Engineering

Submitted By

18SCSE1010517 - ARYAN RAJ
18SCSE1010564 - HIMANSHU KUMAR

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
December , 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled "**IMAGE CAPTIONING USING MACHINE LEARNING AND DEEP LEARNING** "in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **July-2021 to December- 2021**, under the supervision of **Mr V. Arul ,Assistant Professor, Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida .

The matter presented in the project has not been submitted by me for the award of any other degree of this or any other places.

18SCSE1010517-ARYAN RAJ
18SCSE1010564 -HIMANSHU KUMAR

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Mr V. Arul, Assistant Professor)

CERTIFICATE

The Final Project Viva-Voce examination of **18SCSE1010517- ARYAN RAJ ,18SCSE1010564 – HIMANSHU KUMAR** has been held on _____ and their work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place: Greater Noida

ACKNOWLEDGEMENT

Primarily We might thank God for having the ability to finish this mission with success .Then we would love to thank my mission manual Mr. V. Arul ,whose treasured steerage has been those that helped me patch this mission and make it complete evidence success, his pointers and his commands has served because the foremost contributor in the direction of the finishing touch of the mission.

Then We would love to thank our mother and father and friends who've helped me with their treasured pointers and steerage has been beneficial in diverse levels of the finishing touch of the mission.

ABSTRACT

The project is about identifying suitable heading or caption for the image .The image captioning is a classical problem of image processing, computer vision , machine learning and deep learning fields. The main focus of the paper is towards identifying the different parts of the image and collectively give a suitable caption for it. We use the concept of deep learning with convolutional neural networks for this purpose. Different images are selected from the Flickr8k database for the classification purpose. While we are forming the image processing, we are seeing the image but at the same time, we are looking to create a meaningful sequence of words. The first part is handled by CNNs and the second is handled by RNNs. If we can obtain a suitable dataset with images and their corresponding human descriptions, we can train networks to automatically caption images. FLICKR 8K, FLICKR 30K, and MS-COCO are some most used datasets for the purpose. We have analysed the images from various portion areas and conducted experiments. The results shows the effectiveness of deep learning based image captioning in identifying the suitable title for the image.

Keywords: Convolutional neural Networks, Deep Learning, Image Classification, Machine Learning.

Contents

List of Table

List of Figures

Acronyms

ML	**Machine Learning**
DL	**Deep Learning**
CNN	**Convolutional Neural Networks**
RNN	**Recurrent Neural Networks**

CHAPTER-1

Introduction

1.1 Introduction of Project:

Artificial Intelligence(AI) is now at the heart of innovation economy and thus the base for this project is also the same. In the recent past a field of AI namely Deep Learning has turned a lot of heads due to its impressive results in terms of accuracy when compared to the already existing Machine learning algorithms. The task of being able to generate a meaningful sentence from an image is a difficult task but can have great impact, for instance helping the visually impaired to have a better understanding of images. The task of image captioning is significantly harder than that of image classification, which has been the main focus in the computer vision community. A description for an image must capture the relationship between the objects in the image. In addition to the visual understanding of the image, the above semantic knowledge has to be expressed in a natural language like English, which means that a language model is needed. The attempts made in the past have all been to stitch the two models together.

Caption generation is an interesting artificial intelligence problem where a descriptive sentence is generated for a given image. It involves the dual techniques 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. 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. Recently, deep learning methods have achieved state-of the-art results on examples of this problem. It has been demonstrated that

deep learning models are able to achieve optimum results in the field of caption generation problems. Instead of requiring complex data preparation or a pipeline of specifically designed models, a single end-to-end model can be defined to predict a caption, given a photo. In order to evaluate our model, we measure its performance on the Flickr8K dataset using the BLEU standard metric. These results show that our proposed model performs better than standard models regarding image captioning in performance evaluation.

The final phase of the model combines the input from the Image extractor phase and the sequence processor phase using an additional operation then fed to a 256 neuron layer and then to a final output Dense layer that produces a soft max prediction of the next word in the caption over the entire vocabulary which was formed from the text data that was processed in the sequence processor phase. The structure of the network to understand the flow of images and text is shown in the Figure 2.

During training phase we provide pair of input image and its appropriate captions to the image captioning model. The VGG model is trained to identify all possible objects in an image. While LSTM part of model is trained to predict every word in the sentence after it has seen image as well as all previous words. For each caption we add two additional symbols to denote the starting and ending of the sequence. Whenever stop word is encountered it stops generating sentence and it marks end of string. Loss function for model is calculated as, where I represents input image and S represents the generated caption. N is length of generated sentence. pt and St represent probability and predicted word at the time t respectively. During the process of training we have tried to minimize this loss function.

Generating a caption for a given image is a challenging problem in the deep learning domain. In this article, we will use different techniques of computer vision and NLP to recognize the context of an image and describe them in a

natural language like English. we will build a working model of the image caption generator by using CNN (Convolutional Neural Networks) and LSTM (Long short term memory) units.

For training our model I'm using Flickr8K dataset. It consists of 8000 unique images and each image will be mapped to five different sentences which will describe the image.

Step 1: Import the required libraries

Step 2: Load the descriptions

The format of our file is image and caption separated by a newline ("¥n") i.e, it consists of the name of the image followed by a space and the description of the image in CSV format. Here we need to map the image to its descriptions by storing them in a dictionary.

Step 3: Cleaning the text

One of the main steps in NLP is to remove noise so that the machine can detect the patterns easily in the text. Noise will be present in the form of special characters such as hashtags, punctuation and numbers. All of which are difficult for computers to understand if they are present in the text. So we need to remove these for better results. Additionally, you can also remove stop words and perform **Stemming** and **Lemmatization** by using NLTK library.

Step 4: Generate the Vocabulary

Vocabulary is a set of unique words which are present in our text corpus. When processing raw text for NLP, everything is done around the vocabulary.

Step 5: Load the images

Here we need to map the images in the training set to their corresponding descriptions which are present in our descriptions variable. Create a list of names of all training images and then create an empty dictionary and map the images to

their descriptions using image name as key and a list of descriptions as its value. while mapping the descriptions add unique words at the beginning and end to identify the start and end of the sentence.

Step 6: Extract the feature vector from all images

Now we will give an image as an input to our model but unlike humans, machines cannot understand the image by seeing them. So we need to convert the image into an encoding so that the machine can understand the patterns in it. For this task, I'm using transfer learning i.e, we use a pre-trained model that has been already trained on large datasets and extract the features from these models and use them for our work. Here I'm using the InceptionV3 model which has been trained on Imagenet dataset that had 1000 different classes to classify. We can directly import this model from Keras.applications module.

We need to remove the last classification layer to get the (2048,) dimensional feature vector from InceptionV3 model.

Step 7: Tokenizing the vocabulary

In this step, we need to tokenize all the words present in our vocabulary. Alternatively, we can use tokenizer in Keras to do this task.

Step 8: Glove vector embeddings

GloVe stands for global vectors for word representation. It is an unsupervised learning algorithm developed by Stanford for generating word embeddings by aggregating global word-word co-occurrence matrix from a corpus. Also, we have 8000 images and each image has 5 captions associated with it. It means we have 30000 examples for training our model. As there are more examples you can also use data generator for feeding input in the form of batches to our model rather than giving all at one time. For simplicity, I'm not using this here.

Also, we are going to use an embedding matrix to store the relations between words in our vocabulary. An embedding matrix is a linear mapping of the original space to a real-valued space where entities will have meaningful relationships.

Step 9: Define the model

For defining the structure of our model, we will be using the Keras Model from Functional API. It has three major steps:

- Processing the sequence from the text
- Extracting the feature vector from the image
- Decoding the output by concatenating the above two layers

- **Step 10: Training the model**
- For training our model I'm using Adam's optimizer and loss function as categorical cross-entropy. I'm training the model for 50 epochs which will be enough for predicting the output. In case you have more computational power (no. of GPU's) you can train it by decreasing batch size and increasing number of epochs.

The development of the image description system may help the visually impaired people "see" the world in the future. Recently, it has drawn increasing attention and become one of the most important topics in computer vision [1–11]. Early image description generation methods aggregate image information using static object class libraries in the image and modeled using statistical language models. Aker and Gaizauskas [12] use a dependency model to summarize multiple web documents containing information related to image locations and propose a method for automatically tagging geotagged images. Li et al. [13] propose a n-gram method based on network scale, collecting candidate phrases and merging them to form sentences describing images from zero. Yang et al. [14] propose a language model trained from the English Gigaword corpus to obtain the estimation of

motion in the image and the probability of colocated nouns, scenes, and prepositions and use these estimates as parameters of the hidden Markov model. The image description is obtained by predicting the most likely nouns, verbs, scenes, and prepositions that make up the sentence. Kulkarni et al. [15] propose using a detector to detect objects in an image, classifying each candidate region and processing it by a prepositional relationship function and finally applying a conditional random field (CRF) prediction image tag to generate a natural language description. Object detection is also performed on images. Lin et al. [16] used a 3D visual analysis system to infer objects, attributes, and relationships in an image and convert them into a series of semantic trees and then learn the grammar to generate text descriptions for these trees.

Some indirect methods have also been proposed for dealing with image description problems, such as the query expansion method proposed by Yagcioglu et al. [17], by retrieving similar images from a large dataset and using the distribution described in association with the retrieved images. The expression is used to create an extended query, and then the candidate descriptions are reordered by estimating the cosine between the distributed representation and the extended query vector, and finally, the closest description is taken as a description of the input image. In summary, the methods described are brainstorming and have their own characteristics, but all have the common disadvantage that they do not make intuitive feature observations on objects or actions in the image, nor do they give an end-to-end mature general model to solve this problem. The efficiency and popularization of neural networks have made breakthroughs in the field of image description and saw new hopes until the advent of the era of big data and the outbreak of deep learning methods.

1.2 Problem Definition:

In day to day life we have seen lots of images on internet and almost everywhere like news, articles. Sometimes images having some short amount of description about it but out of them some images are just images and nothing extra as we are human we can figure out what's in it. So we are trying to build a model that will take input image from user and then machine gives a suitable caption. So due to this our model can analyse thousands of images and then it will be used for test purposes to know what images says. It is very helpful in Artificial Intelligence to recognize images and gives responses to request sends comes from users.

1.2.1 Tool and Technology Used:

In the report we first consider the task of image classification separately. We try to classify the images of the cifar-10 dataset using various classifiers. We first try to train the model using a K-Nearest Neighbour classifier. Then we try to apply some linear classifiers. The accuracy with these models was much less than expected since a high loss factor at the time of classification will amplify the loss even further at the time of caption generation. We then try to train a simple Convolutional Neural Network and achieve decent results within few hours of training. Thus, by the end of this section we conclude that CNN are a good fit to be used as the image encoder for the captioning model .

Python:

Python is an interpreted , high-level , general-purpose programming language .

Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace . Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and largescale projects.

CHAPTER-2

Literature Survey

Image caption generation is a core part of scene understanding, which is important because of its use in a variety of applications (eg. - image search, telling stories from albums, helping visually impaired people understand the web etc.). Over the years, many different image captioning approaches have been developed.

The architectures used by the winners of ILSVRC have contributed a lot to this field. One such architecture used by us was the VGG16 proposed by He et. al. in 2014 . Apart from that the research in the tasks of machine translation have consistently helped in improving the state of the art performance in language generation. In 2015, researchers at Microsoft's AI Lab used a pipeline approach to image captioning . They used a CNN to generate high-level features for each potential object in the image. Then they used Multiple Instance Learning (MIL) to figure out which region best matches each word. The approach yielded 21.9% BLEU score on MSCOCO. After the pipeline approach, researchers at Google came up with the first endto- end trainable model. They were inspired by the RNN model used in machine translation.

Vinyals et al. [1] replaced this encoder RNN with CNN features of the image as the CNN features are widely used in all computer vision tasks. They called this model as Neural Image Caption(NIC). Following this, two researchers at Stanford modified the NIC. They used an approach that leverages datasets of images and their sentence descriptions to learn about the inter-modal correspondences between language and visual data. Their alignment model was based on a novel combination of Convolutional Neural Networks over image regions, bidirectional Recurrent Neural Networks over sentences, and a structured objective to align the two modalities through a multimodal embedding. They used the Flickr8K,

Flickr30K and MSCOCO datasets and achieved state-of-the-art results in the same [4]. Their model was further modified by Jonathan et. al. [5] in 2015 when they proposed a dense captioning task in which each region of an image was detected and a set of descriptions generated. Another model which used a deep convolutional neural network (CNN) and two separate LSTM networks was proposed by Wang et. al. [6] in the year 2016.

One of the most recent work was inspired by the NIC model and was proposed by Xu et. al. in 2016 [7]. They were inspired by the advancements in the field of machine translation and object detection and introduced an attention based model that automatically learned to describe the content of images.
In the past few years, progress has been made not only in image captioning models but also in various evaluation metrics. The accuracy metric used by us was the BLEU score [8]. BLEU - which was a standard evaluation metric adopted by many of the groups - is slowly being replaced by CIDEr proposed by Vedantam et. al. in 2015 [9].

The image captioning problem and its proposed solutions have existed since the advent of the Internet and its widespread adoption as a medium to share images. Numerous algorithms and techniques have been put forward by researchers from different perspectives. Krizhevsky et al. [1] implemented a neural network using non-saturating neurons and a very efficient a unique method GPU implementation of the convolution function. By employing a regularization method called dropout, they succeeded in reducing overfitting. Their neural network consisted of maxpooling layers and a final 1000-way softmax. Deng et al. [2] introduced a new database which they called ImageNet, an extensive collection of images built using the core of the WordNet structure. ImageNet organized the different classes of

images in a densely populated semantic hierarchy. Karpathy and FeiFei [3] made use of datasets of images and their sentence descriptions to learn about the inner correspondences visual data and language. Their work described a Multimodal Recurrent Neural Network architecture that utilises the inferred co-linear arrangement of features in order to learn how to generate novel descriptions of images. Yang et al. [4] proposed a system for the automatic generation of a natural language description of an image, which will help immensely in furthering image understanding. The proposed multimodel neural network method, consisting of object detection and localization modules, is very similar to the human visual system which is able to learns how to describe the content of images automatically. In order to address the problem of LSTM units being complex and inherently sequential across time, Aneja et al. [5] proposed a convolutional network model for machine translation and conditional image generation. Pan et. al [6] experimented extensively with multiple network architectures on large datasets consisting of varying content styles, and proposed a unique model showing noteworthy improvement on captioning accuracy over the previously proposed models. Vinyals et al. [7] presented a generative model consisting of a deep recurrent architecture that leverages machine translation and computer vision, used to generate natural descriptions of an image by ensuring highest probability of the generated sentence to accurately describe the target image. Xu et al. [8] introduced an attention based model that learned to describe the image regions automatically. The model was trained using standard backpropagation techniques by maximizing a variable lower bound. The model was able to automatically learn identify object boundaries while at the same time generate an accurate descriptive sentence.

Image caption generation is a core part of scene understanding, which is important because of its use in a variety of applications (eg. - image search, telling stories

from albums, helping visually impaired people understand the web etc.). Over the years, many different image captioning approaches have been developed.

The architectures used by the winners of ILSVRC have contributed a lot to this field. One such architecture used by us was the VGG16 proposed by He et. al. in 2014 [2]. Apart from that the research in the tasks of machine translation have consistently helped in improving the state of the art performance in language generation.

In 2015, researchers at Microsoft's AI Lab used a pipeline approach to image captioning [3]. They used a CNN to generate high-level features for each potential object in the image. Then they used Multiple Instance Learning (MIL) to figure out which region best matches each word. The approach yielded 21.9% BLEU score on MSCOCO. After the pipeline approach, researchers at Google came up with the first end-to-end trainable model. They were inspired by the RNN model used in machine translation.

Vinyals et al. [1] replaced this encoder RNN with CNN features of the image as the CNN features are widely used in all computer vision tasks. They called this model as Neural

Image Caption(NIC). Following this, two researchers at Stanford modified the NIC. They used an approach that leverages datasets of images and their sentence descriptions to learn about the inter-modal correspondences between language and visual data. Their alignment model was based on a novel combination of Convolutional Neural Networks over image regions, bidirectional Recurrent Neural Networks over sentences, and a structured objective to align the two modalities through a multimodal embedding. They used the Flickr8K, Flickr30K and MSCOCO datasets and achieved state-of-the-art results in the same [4]. Their model was further modified by Jonathan et. al. [5] in 2015 when they proposed a

dense captioning task in which each region of an image was detected and a set of descriptions generated. Another model which used a deep convolutional neural network (CNN) and two separate LSTM networks was proposed by Wang et. al. [6] in the year 2016.

One of the most recent work was inspired by the NIC model and was proposed by Xu et. al. in 2016 [7]. They were inspired by the advancements in the field of machine translation and object detection and introduced an attention based model that automatically learned to describe the content of images.

In the past few years, progress has been made not only in image captioning models but also in various evaluation metrics. The accuracy metric used by us was the BLEU score [8]. BLEU - which was a standard evaluation metric adopted by many of the groups - is slowly being replaced by CIDEr proposed by Vedantam et. al. in 2015 [9]. Data are the basis of artificial intelligence. People are increasingly discovering that many laws that are difficult to find can be found from a large amount of data. In the image description generation task, there are currently rich and colorful datasets, such as MSCOCO, Flickr8k, Flickr30k, PASCAL 1K, AI Challenger Dataset, and STAIR Captions, and gradually become a trend of contention. In the dataset, each image has five reference descriptions, and Table 2 summarizes the number of images in each dataset. In order to have multiple independent descriptions of each image, the dataset uses different syntax to describe the same image. As illustrated in the example in Figure 10, different descriptions of the same image focus on different aspects of the scene or are constructed using different grammars.

CHAPTER-3

Working of Project

3.1 Image Classification

For the task of image captioning we first have to determine a fit model for the task of encoding the image. We discuss three models in the following section.

Unsupervised classification method is a fully automated process without the use of training data. Using a suitable algorithm, the specified characteristics of an image is detected systematically during the image processing stage. The classification methods used in here are 'image clustering' or 'pattern recognition'. Two frequent algorithms used are called 'ISODATA' and 'K-mean'.

Supervised classification method is the process of visually selecting samples (training data) within the image and assigning them to pre-selected categories (i.e., roads, buildings, water body, vegetation, etc.) in order to create statistical measures to be applied to the entire image. 'maximum likelihood' and 'minimum distance' are two common methods to categorize the entire image using the training data. For example, 'maximum likelihood' classification uses the statistical characteristics of the data where the mean and standard deviation values of each spectral and textural indices of the image are computed first. Then, considering a normal distribution for the pixels in each class and using some classical statistics and probabilistic relationships, the likelihood of each pixel to belong to individual classes is computed. Finally, the pixels are labeled to a class of features that show the highest likelihood.

3.1.1 K-nearest neighbour classifier

As our first approach, we will explore the concept of a K-nearest neighbour (KNN) classifier. Such classifiers have nothing to do with Convolutional Neural Networks

and are very rarely used in practice, but they will allow us to get an idea about the basic approach to an image classification problem.

Suppose we have a training set of 50,000 images divided into 10 different 'labels' and we wish to label the remaining 10,000. The k - nearest neighbour classifier will take a test image, compare it to every single one of the training images, and predict the label of the test image based on majority decision of the 'k' closest images. A very simple method is used to compare the images - they are compared pixel by pixel and the difference in values is summed up. In other words, given two images and representing them as vectors I1 and I2, the L1 (or Manhattan) distance between them can be calculated as :

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

Test image -

10	23	45	22
41	100	34	52
36	49	90	150
33	20	200	110

Training image =

19	17	23	167
23	56	49	112
45	34	155	109
67	22	44	12

Pixel-wise difference

9	6	22	145
18	44	15	60
9	15	65	41
34	2	156	98

$d_1 = 739$

Alternatively, the L2 (or Euler) distance can be used to compare images:

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

Fine-tuning of hyperparameters:

The K-nearest neighbour classifier requires a setting for k. Additionally, there are many different distance functions we could have used (L1 norm/L2 norm etc.). Our goal is to find the best such values of the hyperparameters so as to maximize the accuracy of our classifier. One would think that an easy way to achieve this would

be to try out all possible values of k and pick the one that gives us maximum accuracy on our test data. However, this method should never be used to pick hyperparameters. Using test data to pick hyperparameters results in overfitting i.e our model's results will be too optimistic with respect to what we might actually observe when we deploy your model. In other words, it might fail to generalize to other data.

To overcome this, we used 'cross-validation'. The training data is divided into several 'folds' , one fold is used as the 'test fold' and the other folds are used as training folds. Example, in 5-fold cross-validation, we would split the training data into 5 equal folds, use 4 of them for training, and 1 for validation. We would then iterate over which fold is the validation fold, evaluate the performance, and finally average the performance across the different folds.

Knn in practice - it's pros and cons:

One advantage of KNN is that it is very easy to implement. However, what we save on implementation time, we lose in computation time later on. KNN classifier takes no time to train, since all that is required is to store and possibly index the training data. However, we pay that computational cost at test time, since classifying a test example requires a comparison to every single training example. This is backwards, since in practice we often care about the test time efficiency much more than the efficiency at training time. Also, the use of L1 or L2 distances on raw pixel values is not adequate since the distances correlate more strongly with backgrounds and color distributions of images than with their semantic content. For example, the L2 distance between the following images is the same:

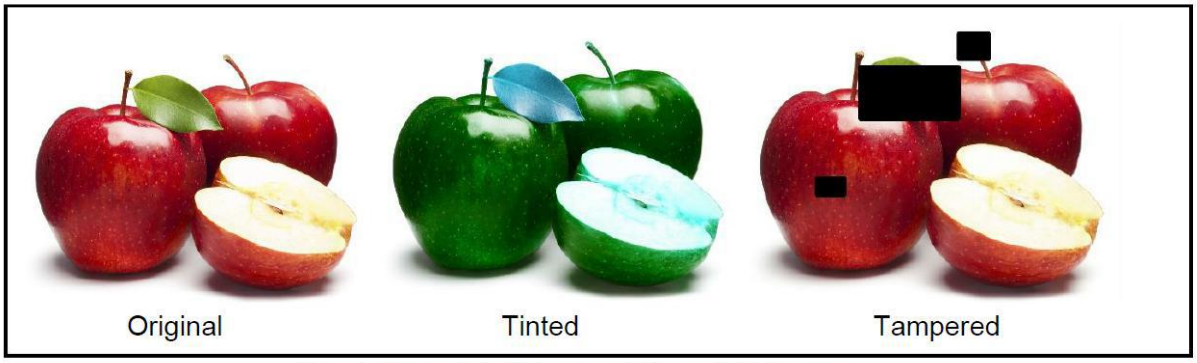


Fig. 1: Same L1 distance of three diff. images

This tells us that same pixel differences don't necessarily translate to same semantic difference.

In conclusion, the KNN Classifier may sometimes be a good choice in some settings (especially if the data is low-dimensional), but it is rarely appropriate for use in practical image classification settings.

3.2 Linear classifiers

In this section, we explore a more powerful approach to image classification that eventually extends to entire Neural Networks and Convolutional Neural Networks. Linear classifiers are an example of parametric classifiers, since we are optimizing some type of numerical values. They have two major components - a score function that maps the raw data to class scores, and a loss function that quantifies the agreement between the predicted scores and the ground truth labels. Then we treat this as an optimization problem in which we minimize the loss function with respect to the parameters of the score function .

Score function

Let's assume a training dataset of images $x_i \in R^D$, each associated with a label y_i. Here i=1…N and $y_i \in$ 1….K. That is, we have **N** examples (each with a

dimensionality **D**) and **K** distinct categories. For example, in CIFAR-10 dataset we have a training set of **N** = 50,000 images, each with **D** = 32 x 32 x 3 = 3072 pixels, and **K** = 10, since there are 10 distinct labels (dog, cat, car, etc).

We will now define the score function f : $R^D \rightarrow R^K$. that maps the raw image pixels to class scores.

$$f(x_i, W, b) = Wx_i + b$$

Where W is the weight matrix and B is the bias vector. W has dimensions 10x3072, xi has dimensions 3072x1 and b has dimensions 10x1.

Essentially, each row of W acts as a classifier for one class of y.

Example- say we have a 4-pixel image that needs to be classified into one of 3 classes. The image will be stretched out into a 12x1 column vector, multiplied with a 3x12 weight matrix and added to a bias vector of dimension 3x1. The result will be a 3x1 column vector where each row represents the image score for that class .

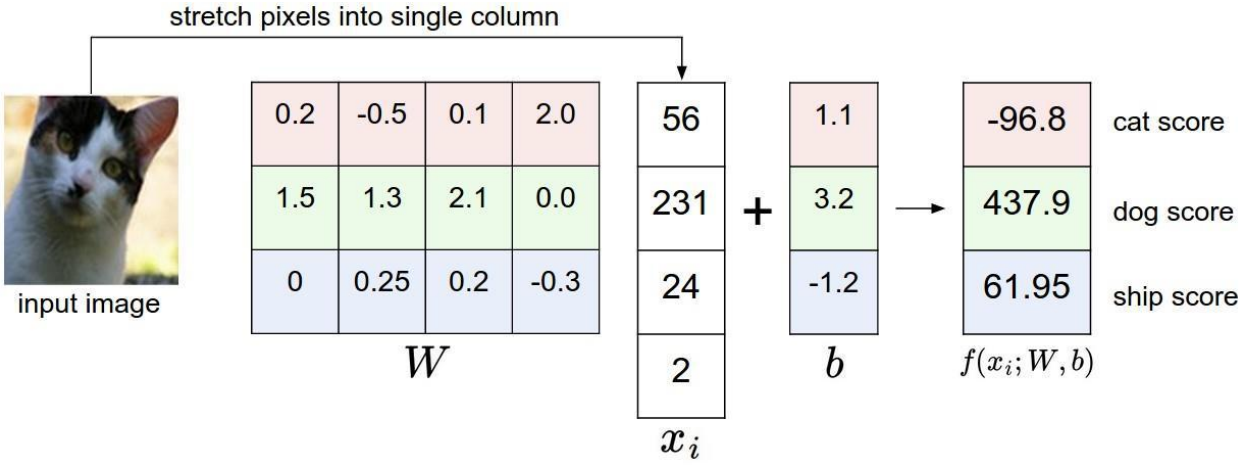


Fig.2 : Visualization of score function with 4 pixels

Another way to interpret a linear classifier is looking at each image as a point in a high-dimensional space. For instance, each image in CIFAR-10 could be thought

of as a point in 3072-dimensional space of 32x32x3 pixels. Analogously, we could say that the entire dataset is a (labeled) set of points, and the linear classifier is drawing gradients in the direction of decreasing similarity with a given class of objects.

Yet another way to interpret linear classifiers is is that each row of the weight matrix W corresponds to a template for one of the classes. The score of each class for an image is then obtained by comparing each template with the image using a dot product one by one to find the one that "fits" best. With this terminology, the linear classifier is doing template matching, where the templates are learned. This is quite similar to nearest neighbour search, except for the fact that one 'template' is being constructed per class instead of comparing the test image to thousands of images in each class .

Fig.3 : Templates of weights generated by linear classifier

Loss function

Loss functions are used to measure the discrepancy between the model's prediction and the desired output. In other words, the loss function quantifies our unhappiness with predictions on the training set. Intuitively, the value of the loss function will be high if we're doing a poor job of classifying the training data, and it will be low if we're doing well.

3.2.1 Hinge Loss (SVM Classifier)

Hinge loss is set up so that it "wants" the correct class for each image to a have a score higher than the incorrect classes by some fixed margin Δ.

The score function takes the pixels and computes the vector (,) of class scores. For example, the score for the j-th class is the j-th element: = (,) .

3.2.2 Cross-entropy loss (SoftMax classifier)

The SoftMax classifier uses a different loss function- namely, the cross-entropy loss. Unlike the SVM which treats the outputs $f(x_i,W)$ as (uncalibrated and possibly difficult to interpret) scores for each class, the Softmax classifier gives a slightly more intuitive output (normalized class probabilities) and also has a probabilistic interpretation. Instead of interpreting each row of $f(x_i,W)$ as score for that particular class, we interpret it as the probability of the image belonging to that class. The cross-entropy loss has the form –

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j f_j}\right),$$

$$or\ L_i = -f_{y_i} + \log\sum_j f_j$$

where we are using the notation f_i to mean the j-th element of the vector of class scores f. As before, the full loss for the dataset is the mean of L_i over all training examples together with a regularization term R(W) (please see next section). The function $fj(z) = ezj/\Sigma_k e^{zk}$ is called the SoftMax function.

3.2.3 SVM vs SoftMax

Unlike the SVM which computes uncalibrated and not easy to interpret scores for all classes, the SoftMax classifier allows us to compute “probabilities” for all labels. For example, given an image the SVM classifier might give us scores [12.5, 0.6, -23.0] for the classes “cat”, “house” and “dog”. The SoftMax classifier can instead compute the probabilities of the three labels as [0.9, 0.09, 0.01], which allows us to interpret its confidence in each class. In practice, the performance difference between the SVM and SoftMax are usually very small.

The SVM does not care about the details of the individual scores: if they were instead [10, -100, -100] or [10, 9, 9] the SVM would be indifferent since the margin of delta = 1 is satisfied and hence the loss is zero. However, these scenarios are not equivalent to a SoftMax classifier, which would accumulate a much higher loss for the scores [10, 9, 9] than for [10, -100, -100]. In other words, the Softmax classifier is never fully happy with the scores it produces: the correct class could always have a higher probability and the incorrect classes always a lower probability and the loss would always get better. However, the SVM is happy once the margins are satisfied and it does not micromanage the exact scores beyond this constraint. This can be thought of as a feature: For example, a ‘dog classifier’ should be spending most of its “effort” on the difficult problem of classifying different breeds of dogs, and should completely ignore the cat examples, which it already assigns very low scores to, and which likely cluster around a completely different side of the data cloud.

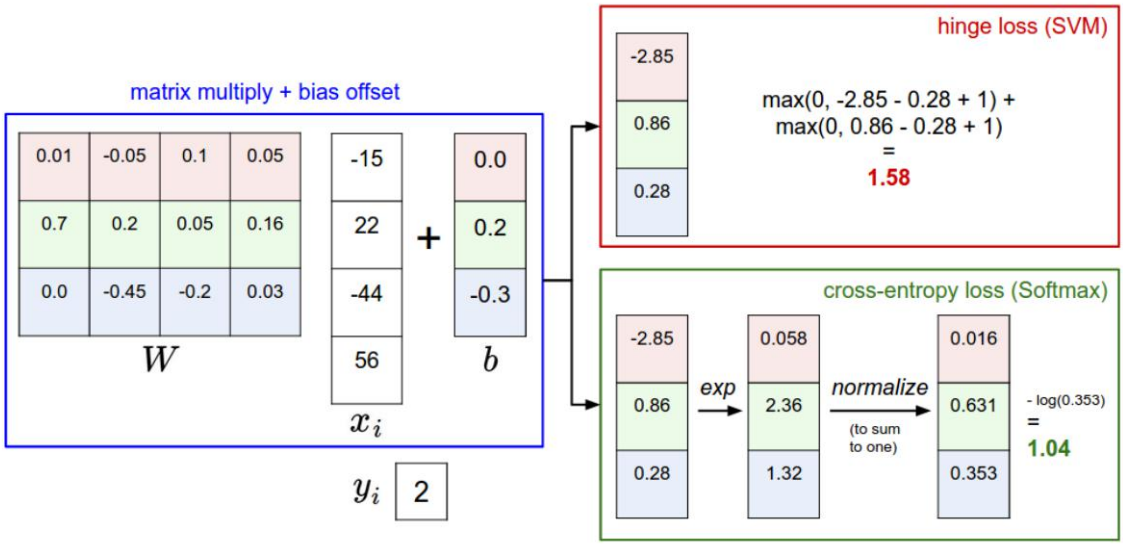


Fig.4 : Score function comparison of SoftMax and SVM

Regularization

Suppose that we have a dataset and a set of parameters W that correctly classify every example. The issue is that this set of W is not necessarily unique: there might be many similar W that correctly classify the examples. Different values of W don't affect the loss function - the loss function yields a score of zero for all W. In such situations, we must have some criteria to choose from the given set of W. This can be achieved by extending the loss function with a regularization penalty R(W). The most common regularization penalty is the L2 norm that discourages large weights through an element-wise quadratic penalty over all parameters:

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

In the expression above, we are summing up all the squared elements of W.

For example, suppose that we have some input image vector x=[1,1,1,1] and two weight vectors w1=[1,0,0,0] and w2=[0.25,0.25,0.25,0.25]. Let's ignore the bias for the sake of simplicity.

Then $w1Tx$=$w2Tx$=1 so both weight vectors lead to the same dot product, but the $L2$ penalty of $w1$ is 1.0 while the $L2$ penalty of $w2$ is only 0.25. Therefore, according to the $L2$ penalty the weight vector $w2$ would be preferred since it achieves a lower regularization loss. Intuitively, this is because the weights in $w2$ are smaller and more diffuse. Since the $L2$ penalty prefers smaller and more diffuse weight vectors, the final classifier is encouraged to take into account all input dimensions to small amounts rather than a few input dimensions and very strongly. As we will see later in the class, this effect can improve the generalization performance of the classifiers on test images and lead to less overfitting.

Optimization (Stochastic Gradient Descent)

SVM loss function for a single data point:

$$Li=\Sigma\max(0,j\neq yi\ wjTxi-wyiTxi+\Delta)$$

We can differentiate the function with respect to the weights. For example, taking the gradient with respect to wyi we obtain:

$$\nabla wyiLi=-(\Sigma(wjTxi-wyiTxi+\Delta>0))xij\neq yi$$

where 1 is the indicator function that is one if the condition inside is true or zero otherwise. This means that we simply count the number of classes that didn't meet the desired margin (and hence contributed to the loss function) and then scale the data vector xi by this number. This gives us a way to calculate the gradient analytically. But this is the gradient only with respect to the row of W that corresponds to the correct class. For the other rows where $j\neq yi$ the gradient is:
$\nabla wjLi=1(wjTxi-wyiTxi+\Delta>0))xi$
Once we have calculated the gradient, it is straight-forward to implement the expressions and use them to perform the gradient update: $W=W-mu*gradient$,

where mu is the step size.

3.3 Convolutional Neural Network

Convolutional Neural Networks (**ConvNets** or **CNNs**) are a category of Artificial Neural Networks which have proven to be very effective in the field of image recognition and classification. They have been used extensively for the task of object detection, self driving cars, image captioning etc. First convnet was discovered in the year 1990 by Yann Lecun and the architecture of the model was called as the LeNet architecture. A basic convnet is shown in the fig. below

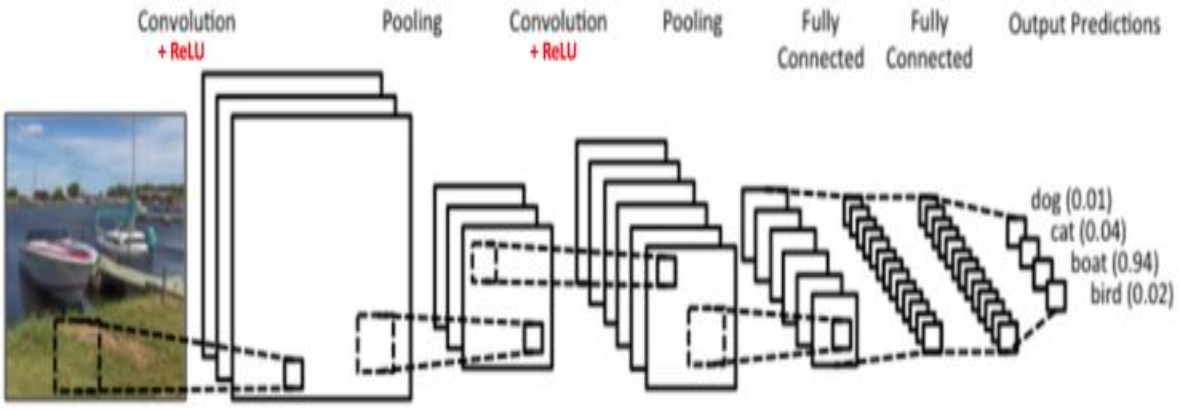


Fig. 5: A simple convnet architecture

The entire architecture of a convnet can be explained using four main operations namely,

1. Convolution
2. Non- Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

These operations are the basic building blocks of *every* Convolutional Neural Network, so understanding how these work is an important step to developing a

sound understanding of ConvNets. We will discuss each of these operations in detail below.

Essentially, every image can be represented as a matrix of pixel values. An image from a standard digital camera will have three channels – red, green and blue – you can imagine those as three 2d-matrices stacked over each other (one for each color), each having pixel values in the range 0 to 255.

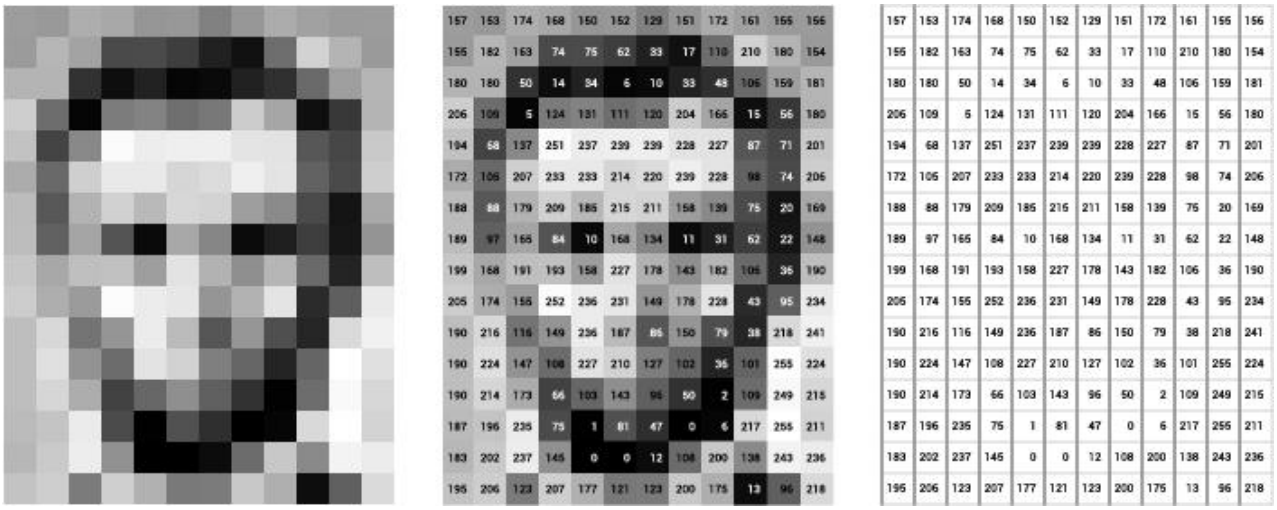


Fig. 6: A grayscale image as matrix of numbers

Convolution Operator

The purpose of convolution operation is to extract features from an image. We consider filters of size smaller than the dimensions of image. The entire operation of convolution can be understood with the example below.

Consider a small 2-dimensional 5*5 image with binary pixel values. Consider another 3*3 matrix shown in Fig. 7.

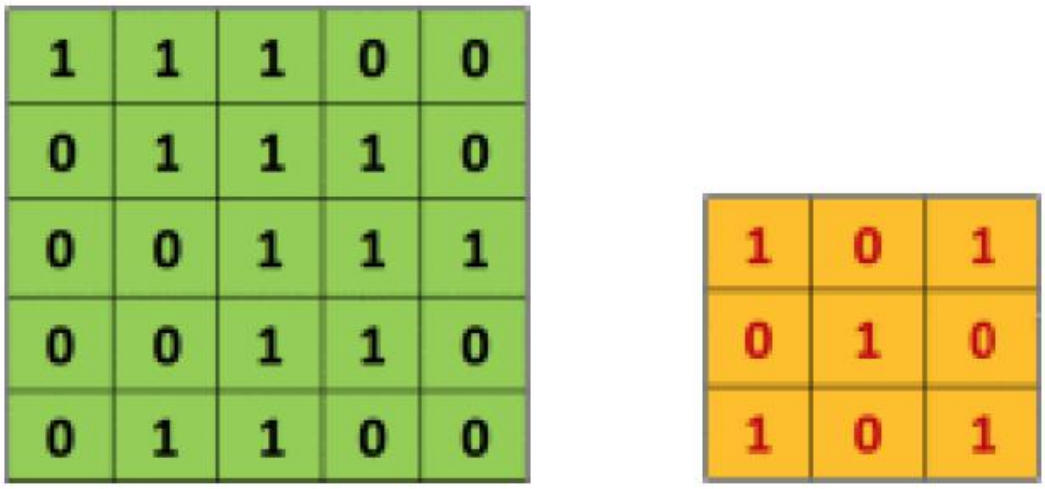


Fig. 7: Image (in green) and Filter (in orange)

We slide this orange 3*3 matrix over the original image by 1 pixel and calculate element-wise multiplication of the orange matrix with the sub-matrix of the original image and add the final multiplication outputs to get the final integer which forms a single element of the output matrix which is shown in the Fig. 8 by the pink matrix.

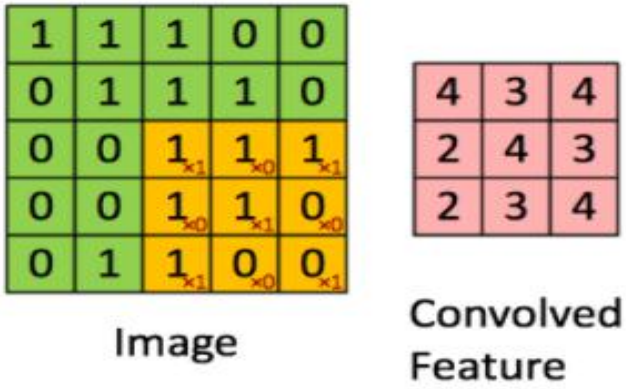


Fig. 8: Convolution operation

The 3*3 matrix is called a filter or kernel or feature detector and the matrix formed by sliding the filter over the image and computing the dot product is called the

Convolved Feature or Activation Map or the Feature Map. The number of pixels by which we slide the filter over the original image is known as stride.

Introducing Non-Linearity

An additional operation is applied after every convolution operation. The most commonly used non-linear function for images is the ReLU which stands for Rectified Linear Unit. The ReLU operation is an element-wise operation which replaces the negative pixels in the image with a zero.

Since most of the operations in real-life relate to non-linear data but the output of convolution operation is linear because the operation applied is elementwise multiplication and addition. The output of the ReLU operation is shown in the figure below.

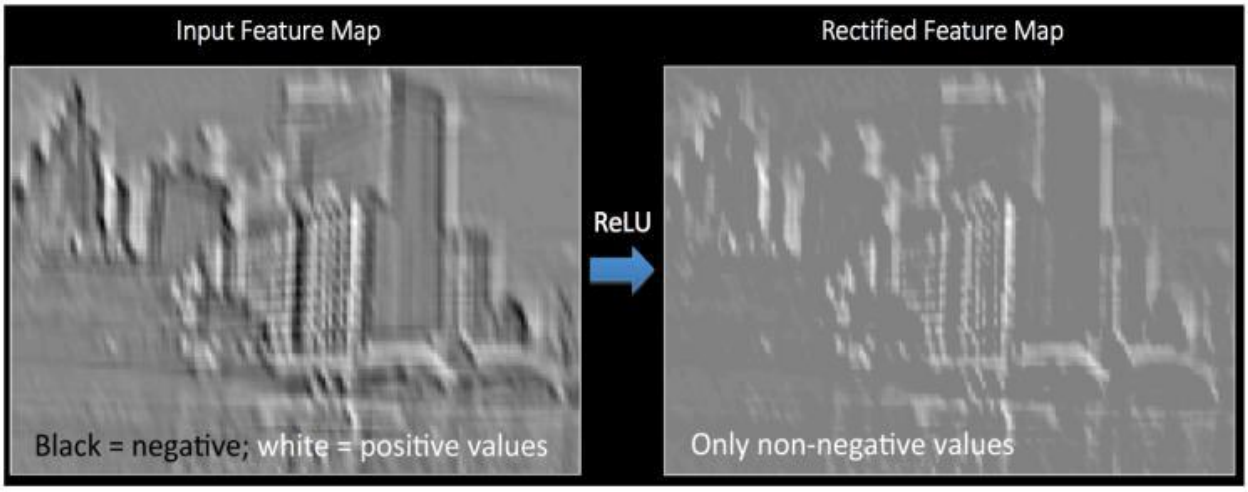


Fig. 9: Output after a ReLU operation

Some other commonly used non-linearity functions are sigmoid and tanh.

Spatial Pooling

The pooling operation reduces the dimensionality of the image but preserves the important features in the image. The most common type of pooling technique used

is max pooling. In max pooling you slide a window of n*n where n is less than the side of the image and determine the maximum in that window and then shift the window with the given stride length. The complete process is specified by the fig.

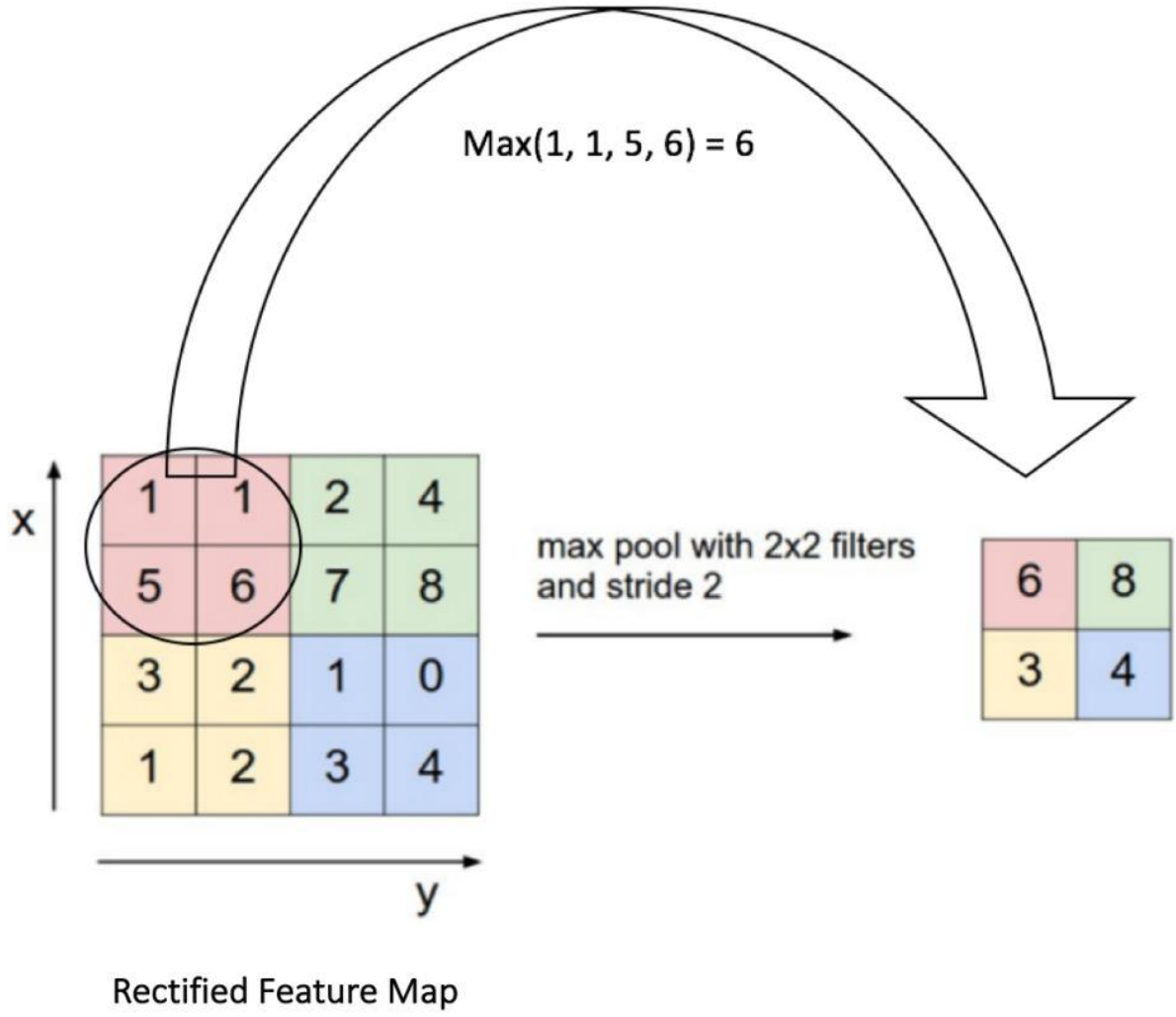


Fig. 10: Max pooling operation

Fully-Connected layer

The fully connected layer is the multi-layer perceptron that uses the SoftMax activation function in the output layer. The term “fully-connected” refers to the fact that all the neurons in the previous layer are connected to all the neurons of the next layer. The convolution and pooling operation generate features of an image.

The task of the fully connected layer is to map these feature vectors to the classes in the training data.

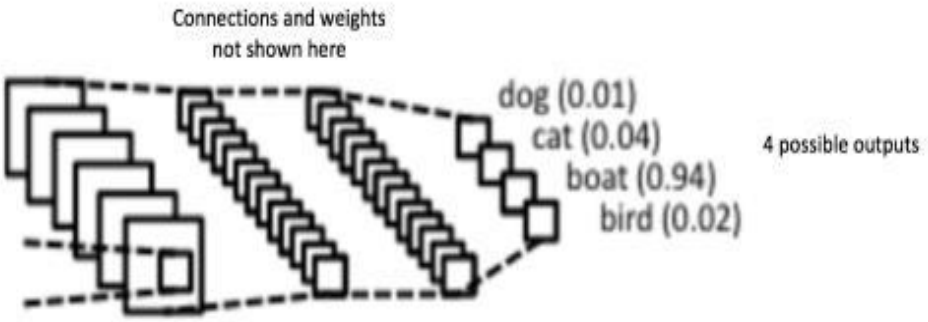


Fig. 11: An example of fully connected layer of data with 4 classes

The task of image classification on cifar-10 has shown state of the art results with the use of convnets. We use the alex net architecture proposed by Alex krizhevsky with a few tweaks. Alexnet is trained for images having 224*224 dimensions and hence need to be modified to be used for cifar-10 since the images in cifar-10 are 32*32. The model used by us has alternate layers of convolution and non-linearities. We use a fully connected layer at the end which uses softmax activation to give the scores of the 10 classes present in the cifar-10 dataset.

The dataset on these convnets yield an accuracy of 85% within around 1.5 hrs of training on gpus. The plots of loss and accuracy on test and validation set are shown in the figures below.

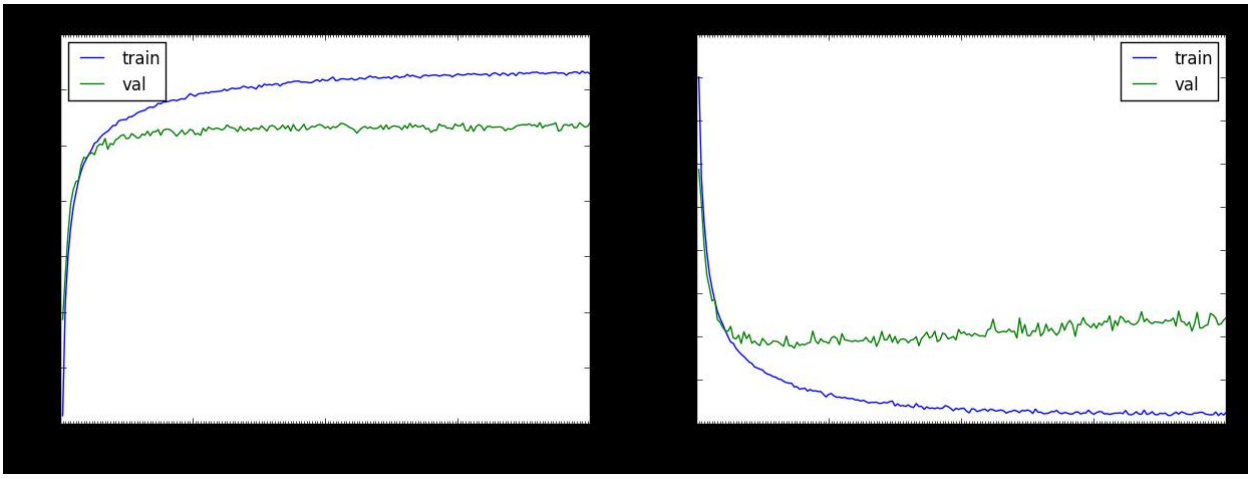
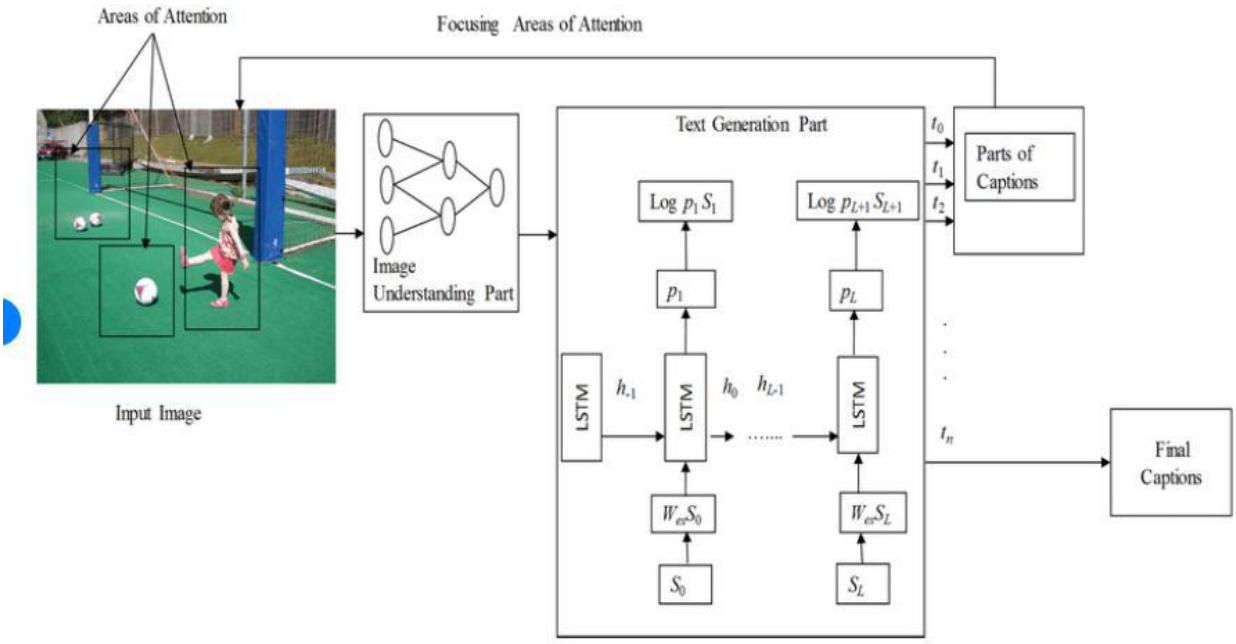


Fig. 12: Accuracy and loss plot on training and validation set

The model is built using tensorflow. Tensorflow is an open source library developed by Google brain team for machine learning. Though being a python api, most of the code of tensorflow is written in C++ and CUDA which is nvidia's programming language for gpus. This helps tensorflow in faster execution of code since python is slower than CPP. Also, the use of gpu enhances the performance of the code significantly.

3.4 Image Captioning Model



A block diagram of a typical attention-based image captioning technique.

3.4.1 Model Overview

The model proposed takes an image *I* as input and is trained to maximize the probability of *p(S|I)* [1] where *S* is the sequence of words generated from the model and each word is generated from a dictionary built from the training dataset. The input image *I* is fed into a deep vision Convolutional Neural Network (CNN) which helps in detecting the objects present in the image.

The image encodings are passed on to the Language Generating Recurrent Neural Network (RNN) which helps in generating a meaningful sentence for the image as shown in the fig. 13. An analogy to the model can be given with a language translation RNN model where we try to maximize the *p(T|S)* where T is the translation to the sentence S. However, in our model the encoder RNN which helps in transforming an input sentence to a fixed length vector is replaced by a CNN encoder. Recent research has shown that the CNN can easily transform an input image to a vector.

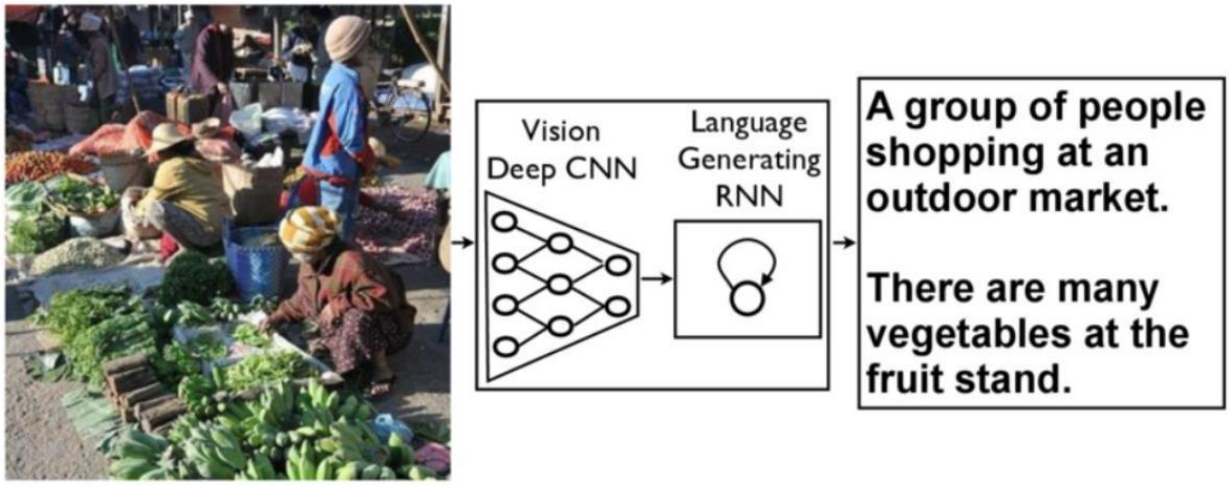


Fig.13 : An overview of the image captioning model

3.4.2 Dataset

For the task of image captioning we use Flickr8k dataset. The dataset contains 8000 images with 5 captions per image. The dataset by default is split into image and text folders. Each image has a unique id and the caption for each of these images is stored corresponding to the respective id. The dataset contains 6000 training images, 1000 development images and 1000 test images. A sample from the data is given in fig.

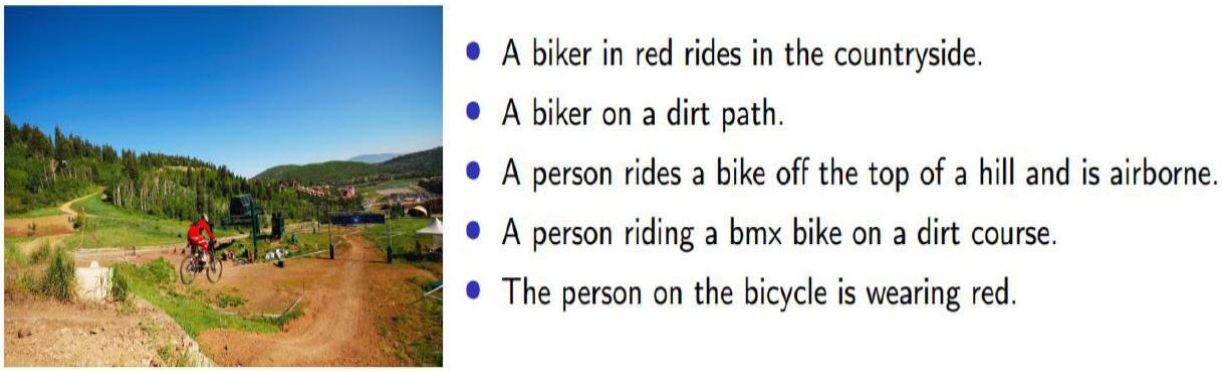


Fig.14 : Sample image and corresponding captions from the Flickr8k dataset

Other datasets like Flickr30k and MSCOCO for image captioning exist but both these datasets have more than 30,000 images thus processing them becomes computationally very expensive. Captions generated using these datasets may

prove to be better than the ones generated after training on Flickr8k because the dictionary of words used by RNN decoder would be larger in case of Flickr30k and MSCOCO.

3.4.3 Deep CNN Architecture :-

The details of the CNN were discussed in section 3.3. Convolutional Neural Network (CNN) have improved the task of image classification significantly. Imagenet Large Scale Visual Recognition competition(ILSVRC) have provided various opensource deep learning frameworks like ZFnet, Alexnet, Vgg16, Resnet etc have shown great potential in the field of image classification. For the task of image encoding in our model we use Vgg16 which is a 16-layered network proposed in ILSVRC 2014 [2]. VGG16 significantly decreased the top-5 error rate in the year 2014 to 7.3%. The image taken for classification needs to be a 224*224 image. The only preprocessing done is by subtracting the mean RGB values from each pixel determined from the training images.

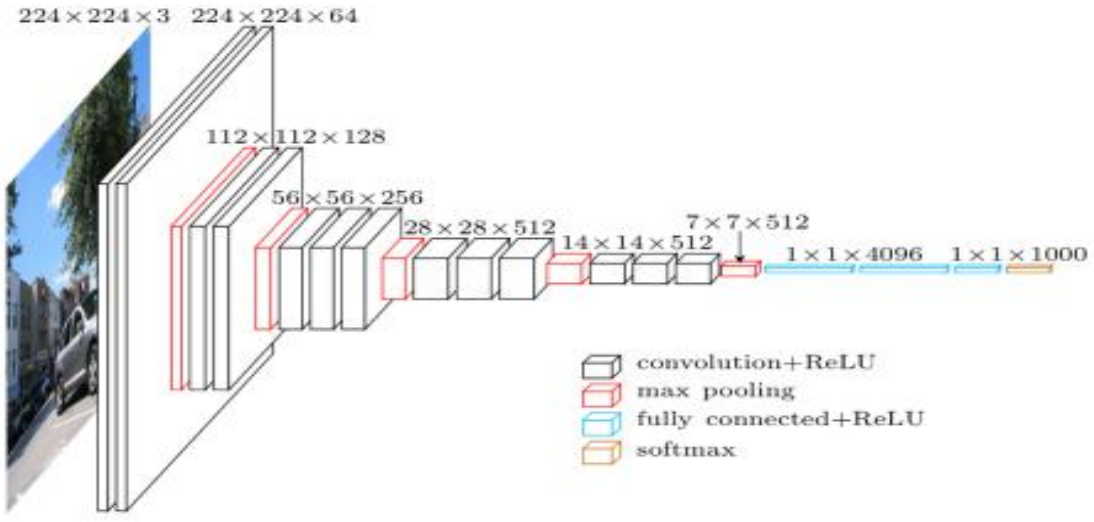


Fig. 15: VGG16 architecture

The convolution layer consists of 3*3 filters and the stride length is fixed at 1. Max pooling is done using 2*2-pixel window with a stride length of 2. All the images need to be converted into 224*224-dimensional image. A Rectified Linear Unit

(ReLU) activation function is follows every convolution layer. A ReLU computes the function ( )=max (0,). The output of the ReLU function is given below:

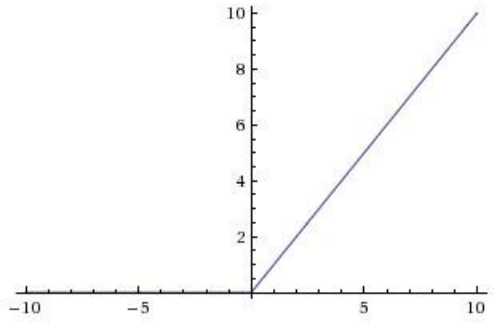


Fig. 16: Rectified linear unit activation function

The advantage of using a ReLU layer over sigmoid and tanh is that it accelerates the stochastic gradient descent. Also unlike the extensive operations (exponential etc.) the ReLU operation can be easily implemented by thresholding a matrix of activations at zero. For our purpose however, we need not classify the image and hence we remove the last 1*1*1000 classification layer.

The output of our CNN encoder would thus be a 1*1*4096 encoded which is then passed to the language generating RNN. There have been more successful CNN frameworks like Resnet but they are computationally very expensive since the number of layers in Resnet was 152 as compared to vgg16 which is only a 16-layered network. A comparison between the layers vs top-5 error rate in the ILSVRC challenge is given below.

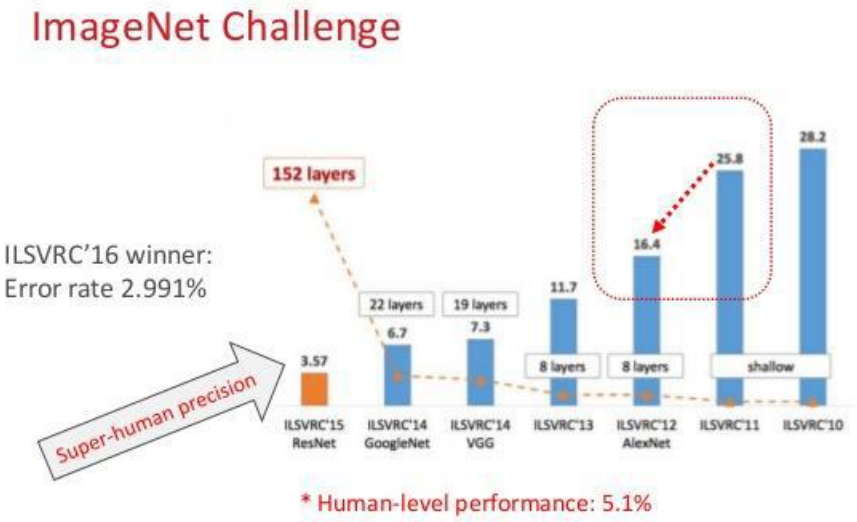


Fig. 17: Top-5 error rate vs the no. of layers

3.4.4 Recurrent Neural Net (RNN) Decoder Architecture

Recurrent neural nets are a type of artificial neural network in which connection between units form a directed cycle. The advantage of using RNN over conventional feed forward net is that the RNN can process arbitrary set of inputs using its memory. RNNs were discovered in the year 1980 by John Hopfield who gave the famous Hopfield model. Recurrent neural nets in simple terms can be considered as networks with loops which allows the information to persist in the network.

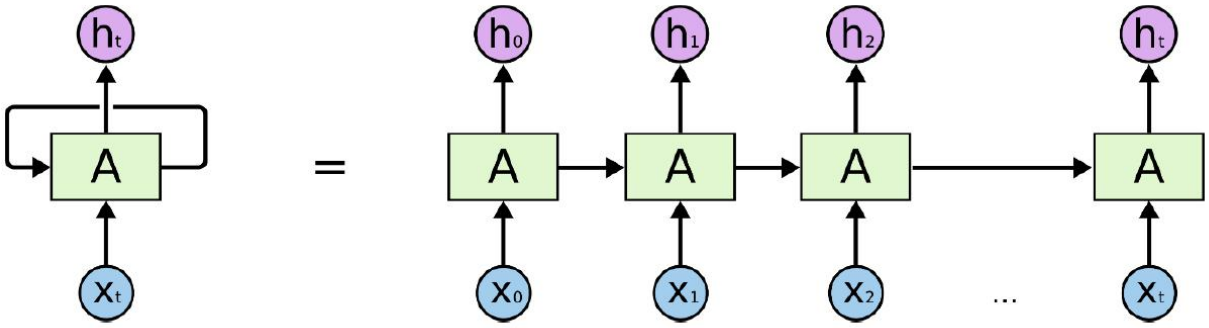


Fig. 18: A simple neural network unrolled into simple neural net

As shown in the figure above a recurrent neural network can be considered as multiple copies of same network with each network passing the message to its successor.

One of the problems with RNNs is that they do not take long-term dependencies into account. Consider a machine that tries to generate sentences on its own. For instance, the sentence is “I grew up in England, I speak fluent English”, if the machine is trying to predict the last word in the sentence i.e. *English*, the machine needs to know that the language name to be followed by fluent is dependent on the context of the word England. It is possible that the gap between the relevant information and the point where it is needed becomes very large in which case the conventional RNNs fail.

To overcome the above-mentioned problem of “long term dependencies”, Hochreiter and Schmidhuber proposed the Long Short-Term Memory (LSTM) networks in the year 1997. Since then LSTM networks have revolutionized the fields of speech recognition, machine translation etc. Like the conventional RNNs, LSTMs also have a chain like structure, but the repeating modules have a different structure in case of a LSTM network. A simple LSTM network is shown in Fig. 19.

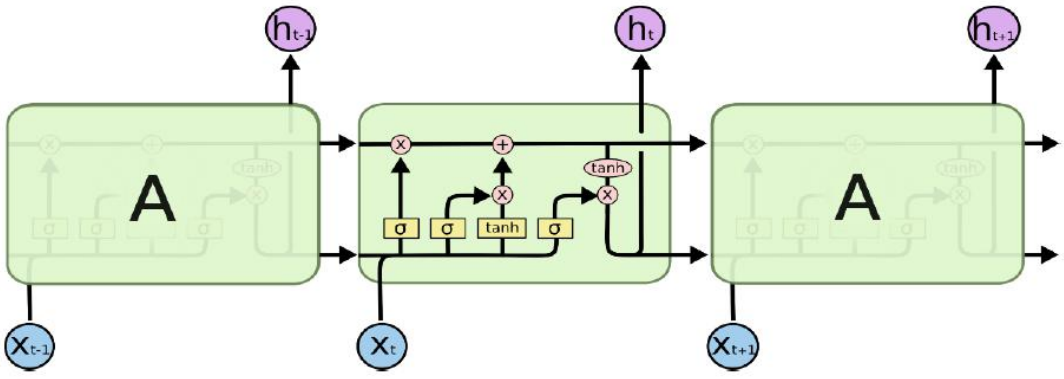


Fig. 19: Four interacting layers in a LSTM layer

The key behind the LSTM network is the horizontal line running on the top which is known as the cell state. The cell state runs through all the repeating modules and

is modified at every module with the help of gates. This causes the information in a LSTM network to persist.

We use this LSTM network with a slight variation. The architecture of the LSTM network used is given below.

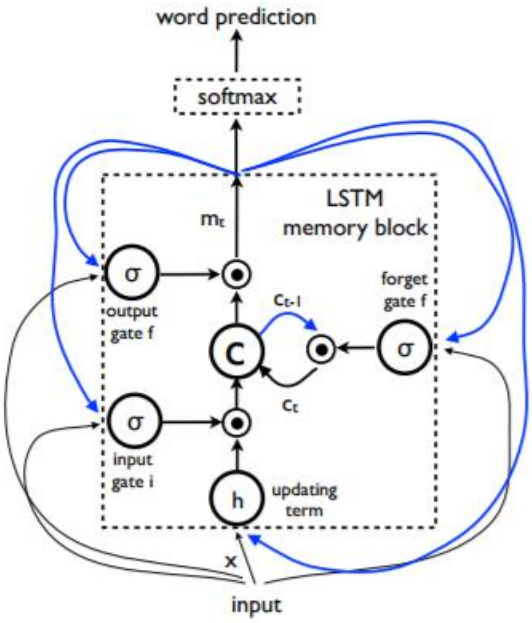


Fig. 20: LSTM architecture for language generation

The entire network is governed by the following equations $it= \sigma(Wixxt+Wimmt-1)$,

where it is the input gate at time t, W represents the trained parameters. The variable $mt-1$ denotes the output of the module at time t-1 and σ represents the sigmoid operation which outputs numbers between zero and one, describing how much of each component should be let through. $ft= \sigma(Wfxxt+Wfmmt-1)$,

where ft represents the forget gate which control whether to forget the current cell value. $ot= \sigma(Woxxt+Wommt-1)$,

where ot represents the output gate which determines whether to output the new cell value or not.

The output pt+1 of a module gives the word prediction. The same LSTM network is repeated until an end token (.) is encountered by the network. The series of these word prediction generate the caption for a given image. The complete training process for the combined model (CNN encoder + RNN language generator) and the LSTM network in unravelled form is given below in Fig. 21.

The LSTM model is trained to predict each word of the sentence after it has seen the image as well as all preceding words as defined by $p(St|I, S0, \ldots, St-1)$.

Following are the briefly described important functions in the above mentioned code.

1. Conv2D

This operation creates a layer which is a convolution kernel that is convolved with the layer input to produce a tensor of outputs. The important arguments to this function are-

Filters: The first parameter to the function is filters i.e. an Integer that determines the dimensionality of the output space (the number output of filters in the convolution).

Strides: The second argument to the function is an integer or tuple/list of 2 integers, specifying the strides of the convolution along the width and height. Can be a single integer to specify the same value for all spatial dimensions.

Activation: The third argument to the function is which Activation function to use which in this case is defined as ReLU. If you don't specify anything, linear activation is applied.

2. MaxPooling2D

This operation performs maximum spatial pooling. The important arguments are-

Pool_size: First argument to the function is an integer or tuple of 2 integers, factors by which to downscale (vertical, horizontal). If only one integer is specified, the same window length will be used for both dimensions.

Strides: Second argument to the function is an Integer, tuple of 2 integers, or None. The argument governs the same operation as discussed in Conv2D layer. If None, it will default to pool_size.

3. Flatten

The flatten() function in keras is used to flatten the input. It does not effect the batch size. For instance, the output of a particular layer is of dimension (32, 32, 64) then after applying the flatten function the dimension of the feature vector would become 65536 i.e. 32*32*64. The operation of the flatten function is similar to that of numpy.reshape() function.

4. Dense

The dense layer is fully connected layer, so all the neurons in a layer are connected to those in a next layer. One important parameter to the Dense function is –

Activation: It specifies the activation function to be used. The softmax activation function in the last line maps the encoded vector in the previous step to all possible output classes.

The language generating RNN is initialized with the following block of code

```
from keras.models import Sequential
from keras.layers import LSTM, Embedding, TimeDistributed, Dense,
RepeatVector, Merge, Activation
image_model = Sequential()
image_model.add(Dense(128, input_dim = 4096, activation='relu'))
image_model.add(RepeatVector(self.max_length))
```

```
lang_model = Sequential()
lang_model.add(Embedding(self.vocab_size, 256, input_length=self.max_length))
lang_model.add(LSTM(256,return_sequences=True))
lang_model.add(TimeDistributed(Dense(EMBEDDING_DIM)))
model = Sequential()
model.add(Merge([image_model, lang_model], mode='concat'))
model.add(LSTM(1000,return_sequences=False))
model.add(Dense(self.vocab_size))
model.add(Activation('softmax'))
```

3.4.5 Code

The code for the model was built on keras. Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. We use tensorflow as the backend to build the code.

Image Captioning

- Generating Captions for Images

Steps

- Data collection
- Understanding the data
- Data Cleaning
- Loading the training set
- Data Preprocessing — Images
- Data Preprocessing — Captions
- Data Preparation using Generator Function
- Word Embeddings
- Model Architecture
- Inference

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import keras
import re
import nltk
from nltk.corpus import stopwords
import string
import json
from time import time
import pickle
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.models import Model, load_model
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from keras.layers import Input, Dense, Dropout, Embedding, LSTM
from keras.layers.merge import add

import keras.applications.resnet
```

In [2]:
```
# Read Text Captions

def readTextFile(path):
    with open(path) as f:
        captions = f.read()
    return captions
```

In [3]:
```
captions  = readTextFile("C:/Users/aryan/Downloads/Compressed/Data/Flickr8k_text/Flickr8k.token.txt")
captions = captions.split('\n')
```

In [6]:
```
print(len(captions))
```

```
40461
```

In [7]:
```
first,second  = captions[0].split('\t')
print(first.split(".")[0])
print(second)
```

```
1000268201_693b08cb0e
A child in a pink dress is climbing up a set of stairs in an entry way .
```

In [ ]:
```
# Dictionary to Map each Image with the list of captions it has
descriptions={}
for x in captions:
    first,second,*rest = x.split('\t')
    img_name=first.split(".")[0]
    #if the image is already p
```

In [ ]:
```
# Dictionary to Map each Image with the list of captions it has
descriptions={}
for x in captions:
    first,second,*rest = x.split('\t')
    img_name=first.split(".")[0]
    #if the image is already present or not
    if descriptions.get(img_name) is None:
        descriptions[img_name]=[]
    descriptions[img_name].append(second)
```

In [12]:
```
descriptions["1000268201_693b08cb0e"]
```

Out[12]:
```
['A child in a pink dress is climbing up a set of stairs in an entry way .',
 'A girl going into a wooden building .',
 'A little girl climbing into a wooden playhouse .',
 'A little girl climbing the stairs to her playhouse .',
 'A little girl in a pink dress going into a wooden cabin .']
```

In [13]:
```
IMG_PATH = "C:/Users/aryan/Downloads/Compressed/Data/Flickr8k_Dataset/Flicker8k_Dataset/"
import cv2
import matplotlib.pyplot as plt


img = cv2.imread(IMG_PATH+"1000268201_693b08cb0e.jpg")
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img)
plt.axis("off")
plt.show()
```

Data Cleaning

In [14]:
```
def clean_text(sentence):
    sentence = sentence.lower()
    sentence = re.sub("[^a-z]+"," ",sentence)
    sentence = sentence.split()

    sentence  = [s for s in sentence if len(s)>1]
    sentence = " ".join(sentence)
    return sentence
```

In [15]:
```
clean_text("A cat is sitting over the house # 64")
```

Out[15]:
```
'cat is sitting over the house'
```

In [16]:
```
# Clean all Captions
for key,caption_list in descriptions.items():
    for i in range(len(caption_list)):
        caption_list[i] = clean_text(caption_list[i])
```

In [17]:
```
descriptions["1000268201_693b08cb0e"]
```

Out[17]:
```
['child in pink dress is climbing up set of stairs in an entry way',
 'girl going into wooden building',
 'little girl climbing into wooden playhouse',
 'little girl climbing the stairs to her playhouse',
 'little girl in pink dress going into wooden cabin']
```

In [18]:
```
# Write the data to text file
with open("descriptions_1.txt","w") as f:
    f.write(str(descriptions))
```

Vocabulary

In [19]:
```
descriptions = None
with open("descriptions_1.txt",'r') as f:
    descriptions= f.read()

json_acceptable_string = descriptions.replace("'","\"")
descriptions = json.loads(json_acceptable_string)
```

In [20]:
```
print(type(descriptions))
```

```
<class 'dict'>
```

In [21]:
```
# Vocab

vocab = set()
for key in descriptions.keys():
    [vocab.update(sentence.split()) for sentence in descriptions[key]]

print("Vocab Size : %d"% len(vocab))
```

```
Vocab Size : 8424
```

CHAPTER- 4

Results and Discussion

4.1 Result

We define the accuracy of the model by BLEU score. Bilingual evaluation understudy (BLEU) is an algorithm that evaluated the quality of text which has been translated by a machine. It was one of the first metrics to achieve high correlation with human judgement.

Blue score is always defined between 0 and 1, 0 being the machine translation is not at all related to the reference sentence. BLEU's evaluation system requires two inputs:

i (i) a numerical translation closeness metric, which is then assigned and measured against

ii (ii) a corpus of human reference translations.

For example,

Candidate	the	the	the	the	the	the	the
Reference 1	the	cat	is	on	the	mat	
Reference 2	there	is	a	cat	on	the	mat

The candidate in this example has all its words contained in the reference thus giving a unigram precision score of 1. A unigram precision score of 1 means the candidate and reference sentences are highly correlated. However, as we can see that the two are very different from each other. The modification that BLEU makes is straightforward. For each word in the candidate translation, the algorithm takes its maximum total count, *max*, in any of the reference translations. In the

example above, the word "the" appears twice in reference 1, and once in reference 2. Thus *max= 2.*

For the candidate translation, the count mw of each word is clipped to a maximum of *max* for that word. In this case, "the" has *mw =7* and *mmax= 2*, thus *mw* is clipped to 2. These clipped counts *mw* is then summed over all distinct words in the candidate. This sum is then divided by the total number of words in the candidate translation. In the above example, the modified unigram precision score would be: p=2/7

For calculating BLUE score we first generate captions for all the test images and then use theses machine generated captions as candidate sentences. We compare this candidate sentences with 5 of the captions given by humans and average the BLEU score of candidate corresponding to each of the references. Thus for 1000 test images we calculate 1000 BLEU scores using Natural Language Toolkit (NLTK), a python package.

We averaged out these BLEU scores over the 1000 test images. The net BLEU score of the model after training for 70 epochs with a batch size of 512 was found to be 0.562 or 56.2% while the state of the art on Flickr8k is around 66%. On increasing the number of epochs, we may reach near state of the art results but that would require higher computation. The net BLEU score can also be improved by decreasing the batch size.

CHAPTER-5

Conclusion and Future Scope

5.1 Conclusion:-

Our end-to-end system neural network system is capable of viewing an image and generating a reasonable description in English depending on the words in its dictionary generated on the basis of tokens in the captions of train images. The model has a convolutional neural network encoder and a LSTM decoder that helps in generation of sentences. The purpose of the model is to maximize the likelihood of the sentence given the image.

Experimenting the model with Flickr8K dataset show decent results. We evaluate the accuracy of the model on the basis of BLEU score. The accuracy can be increased if the same model is worked upon a bigger dataset. Furthermore, it will be interesting to see how one can use unsupervised data, both from images alone and text alone, to improve image description approaches.

5.2 Future Prospects

The task of image captioning can be put to great use for the visually impaired. The model proposed can be integrated with an android or ios application to work as a real-time scene descriptor. The accuracy of the model can be improved to achieve state of the art results by hyper tuning the parameters.

The model's accuracy can be boosted by deploying it on a larger dataset so that the words in the vocabulary of the model increase significantly. The use of relatively newer architecture, like ResNet and GoogleNet can also increase the accuracy in the classification task thus reducing the error rate in the language geneneration.

Reference

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

[2] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[3] Fang, Hao, et al. "From captions to visual concepts and back." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[4] Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[5] Johnson, Justin, Andrej Karpathy, and Li Fei-Fei. "Densecap: Fully convolutional localization networks for dense captioning." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[6] Wang, Cheng, et al. "Image captioning with deep bidirectional LSTMs." *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016.

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