MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

BHOPAL (M.P.)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MINOR PROJECT

ON

AUTOMATIC IMAGE CAPTION GENERATION USING CONVOLUTIONAL NEURAL NETWORK

SUBMITTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF BACHELOR OF TECHNOLOGY

SUBMITTED BY:

ROHIT PATEL (141112043)

SHUBHENDRA SINGH SISODIYA (141112027)

PUSHKAR RAJ SINDAL (141112047)

BHARAT SUNEL (141112014)

UNDER THE GUIDANCE OF:

PROF. SWETA JAIN

SESSION 2016-2017

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

BHOPAL (M.P.)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **Rohit Patel**, **Shubhendra Singh Sisodiya**, **Pushkar Raj Sindal** and **Bharat Sunel** students of B.Tech 3rd Year (Computer Science & Engineering), have successfully completed their project "**AUTOMATIC IMAGE CAPTION GENERATION USING CONVOLUTIONAL NEURAL NETWORK**" in partial fulfillment of their minor project in Computer Science & Engineering.

PROF. SWETA JAIN	**PROF. SANYAM SHUKLA**
(Project Guide)	(Project Coordinator)

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

BHOPAL (M.P.)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, hereby, declare that the following report which is being presented in the Minor Project Documentation entitled "AUTOMATIC IMAGE CAPTION GENERATION USING CONVOLUTIONAL NEURAL NETWORK" is the partial fulfillment of the requirements of the third year (sixth semester) Minor Project in the field of Computer Science and Engineering. It is an authentic documentation of our own original work carried out under the able guidance of Prof. Sweta Jain and the dedicated co-ordination of Prof.Sanyam Shukla. The work has been carried out entirely at Maulana Azad National Institute of Technology, Bhopal. The following project and its report, in part or whole, has not been presented or submitted by us for any purpose in any other institute or organization.

We, hereby, declare that the facts mentioned above are true to the best of our knowledge. In case of any unlikely discrepancy that may possibly occur, we will be the ones to take responsibility

ROHIT PATEL (141112043)

SHUBENDRA SINGH SISODIYA (141112027)

PUSHKAR RAJ SINDAL (141112047)

BHARAT SUNEL (141112014)

ACKNOWLEDGEMENT

With due respect, we express our deep sense of gratitude to our respected guide Prof. Sweta Jain, for her valuable help and guidance. We are thankful for the encouragement that he has given us in completing this project successfully. His rigorous evaluation and constructive criticism was of great assistance.

It is imperative for us to mention the fact that this minor project could not have been accomplished without the periodic suggestions and advice of our project coordinator Prof. Sanyam Shukla.

We are also grateful to our respected director Dr. N. S Choudhary for permitting us to utilize all the necessary facilities of the college.

Needless to mention is the additional help and support extended by our respected HOD, Dr. R. K. Pateriya, in allowing us to use the departmental laboratories and other services.

We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind co-operation and help.

We are also thankful to Ms. Dixa Saxena for providing us with the continuous guidance and support during the project.

Last but certainly not the least; we would like to express our deep appreciation towards our family members and batch mates for providing the much needed support and encouragement.

CONTENTS

ABSTRACT

A quick glance at an image is sufficient for a human to point out and describe an immense amount of details about the visual scene. Automatic image captioning (also known as automatic image tagging or linguistic indexing) is the process by which a computer system automatically assigns metadata in the form of captioning or keywords to a digital image.

This process is quite useful in the present scenario as the internet is replete with an incredible amount of images that, if not all useless, is useful to one's specific needs. It serves as a huge help for visually impaired people. Social media platforms like facebook can infer directly from the image, where you are (beach, cafe etc), what you wear (colour) and more importantly what you're doing also (in a way).. Google Photos also uses similar technique to classify our gallery photos into mountains, sea, papers etc.

Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and convolutional neural network. The image captioning is broadly divided into two parts. The first one is extracting the features of image and the second one is classification based on the features extracted. Feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps. Feature extraction is done using some constant no of filters which are initialized randomly and are learned during training period. Once features are extracted we are using Keras classifier to predict the class labels which are then mapped to sentences.

In this project, we have focused extensively on Convolutional Neural Network and how they are trained to predict captions. We have studied various methods that are used for Max Pooling, Relu in detail implemented them through code and have tried to answer the questions like how efficiency will improve with the number of hidden layers in the network.

CHAPTER 1
INTRODUCTION

1.1 Overview

Artificial Neural Networks (ANNs) are a new approach that follows a different way from traditional computing methods to solve problems. Since conventional computers use algorithmic approach, if the specific steps that the computer needs to follow are not known, the computer cannot solve the problem. That means, traditional computing methods can only solve the problems that we have already understood and knew how to solve. However, ANNs are, in some way, much more powerful because they can solve problems that we do not exactly know how to solve. That's why, of late, their usage is spreading over a wide range of area including, virus detection, robot control, intrusion detection systems, pattern (image, fingerprint, noise) recognition and so on.

Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. Recent advances are starting to enable machines to describe image with sentences. This project uses neural networks to automatically generate the caption of images.First, features are extracted from the image using the convolutional neural network. These features act as the input to the classifier which classifies the images into class labels which are then mapped to sentences. In this way, short description about the image fed to the neural network is generated.

1.2 Convolutional Neural Network

A **ConvolutionalNeuralNetwork** (CNN, or ConvNet) is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex.
Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have

been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars.

Convolutional Neural Networks are very similar to ordinary Neural Networks. They are made up of neurons that have learnable weights and biases. Eachneuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.
ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture.These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.
A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks).

1.3 Classification

In machine learning and statistics, **Classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of pattern recognition.

In the terminology of machine learning, classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance. An algorithm that implements classification, especially in a concrete implementation, is known as a **classifier**. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm that maps input data to a category.

Terminology across fields is quite varied. In statistics, where classification is often done with logistic regression or a similar procedure, the properties of observations are termed explanatory variables (or independent variables, regressors etc.), and the categories to be predicted are known as outcomes, which are considered to be possible values of the dependent variable. In machine learning, the observations are often known as instances, the explanatory variables are termed features (grouped into a feature vector), and the possible categories to be predicted are classes.

CHAPTER 2

LITERATURE REVIEW

Much work has been done by several universities and many people have published their research papers on the topic of automatic image caption based approach. Here we present some of them:

1. Stanford Computer Science : NeuralTalk2 by Andrej Karpathy (Research Scientist) :

Recurrent Neural Network captions your images. Now much faster and better than the original NeuralTalk. Compared to the original NeuralTalk this implementation is batched, uses Torch, runs on a GPU, and supports CNN finetuning.All of these together result in quite a large increase in training speed for the Language Model (~100x), but overall not as much because we also have to forward a VGGNet. However, overall very good models can be trained in 2-3 days, and they show a much better performance.

2. Learning rich features from RGB-D images for object detection and segmentation (Saurabh Gupta, Ross Girshick, Pablo Arbeláez, Jitendra Malik) :

In this paper they studied the problem of object detection for RGB-D images using semantically rich image and depth features. They proposed a new geocentric embedding for depth images that encodes height above ground and angle with gravity for each pixel in addition to the horizontal disparity. We demonstrate that this geocentric embedding works better than using raw depth images for learning feature representations with convolutional neural networks.

3. Image captioning with semantic attention (Zhaowen Wang) :

In this paper, they proposed a new algorithm that combines both approaches through a model of semantic attention. Our algorithm learns to selectively attend to semantic concept proposals and fuse them into hidden states and outputs of recurrent neural networks.The selection and fusion form a feedback connecting the top-down and bottom-up computation. We evaluate our algorithm on two public benchmarks: Microsoft COCO and Flickr30K. Experimental results show that our algorithm significantly outperforms the state-of-the-art approaches consistently across different evaluation metrics.

CHAPTER 3

PROPOSED WORK

This Project implements the neural network based approach to generate the captions for the images. This process can be divided into two parts – Feature Extraction and Classification. Features extraction is done using a convolutional neural network. For this, we have taken 32 random filter matrices which learn the features by themselves. In this Convolutional Neural Network, we have three types of layers – convolutional layer, max pooling layer and Relu layer. These layers together extract features from the image.

These features now act as the input to the keras classifier which uses sequential model for the classification of images. The keras classifier uses softmax as activation function for each layer. The keras classifier generates class labels which are then mapped to sentences and the error is calculated as the difference between the generated sentence and the original sentence present in the dataset. In this way model is trained for the Flickr 8K dataset and weights are stored.

When we input an image into the model, features are extracted and the keras classifier predicts the sentence according to trained weights and the sentence generated is given as output to the input image.

3.1Flickr8k Dataset

The ability to associate images with natural language sentences that describe what is depicted in them is a hallmark of image understanding, and a prerequisite for applications such as sentence-based image search.

Flickr8k is a new benchmark collection for sentence-based image description and search, consisting of 8,000 images that are each paired with five different captions which provide clear descriptions of the salient entities and events.

Here we are using a python script to convert the flickr8k dataset into the format which is required for pre-processing and maintaining the image details in a JSON file.

JSON Schema:

1. **List**: We are maintaining a single list to store all the image details in the dataset.

2.**Dictionary**: Each list item is a dictionary containing the details of image like name, sentence and image id.

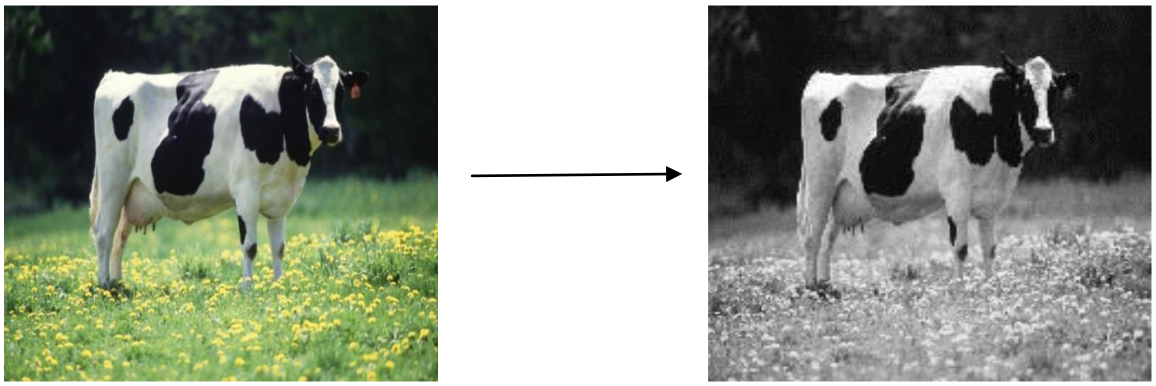
```
[
{ “name”:”Image_name.jpg”,
“sentence”:”Image sentence comes here .”,
“img_id”:0
}
,
.
.
]
```

3.2 Pre-processing of the Image

Image pre-processing is the term for operations on images at the lowest level of abstraction. These operations do not increase image information content but they decrease it if entropy is an information measure. The aim of pre-processing is an improvement of the image data that suppresses undesired distortions or enhances some image features relevant for further processing and analysis task.

Image pre-processing reduces the redundancy in images. Neighbouring pixels corresponding to one real object have the same or similar brightness value. If a distorted pixel can be picked out from the image, it can be restored as an average value of neighbouring pixels.

Here, we are converting the RGB image into gray scale in order to reduce itsdimensions. The converted image is then resized into 200X200 for feature extraction.



Original Image **Resized and Gray Scaled**

Fig 3.1 Resizing Image and Converting to Gray Scale

3.3 Feature Extraction

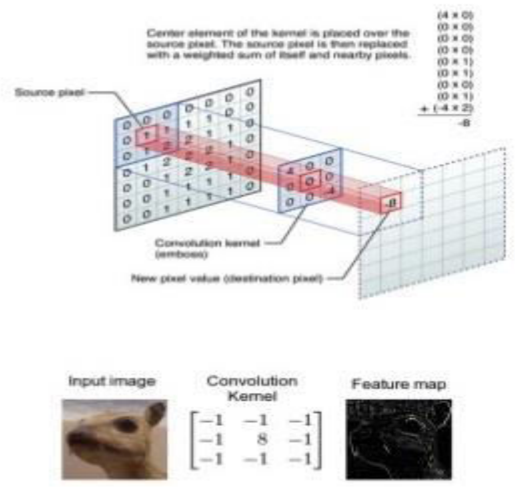
Feature extraction a type of dimensionality reduction that efficiently represents interesting parts of an image as a compact feature vector. This approach is useful when image sizes are large and a reduced feature representation is required to quickly complete tasks such as image matching and retrieval.

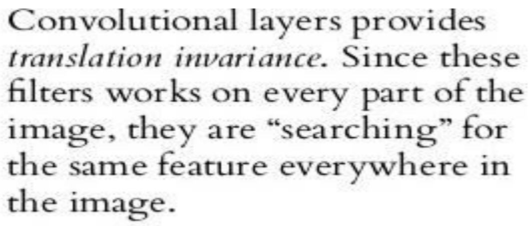
Following methodologies are used in extracting features from an image:

- **Convolution Layer**

Convolution layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. The CONV layer’s parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer.

Now, we will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map.


Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)
Input image
Convolution Kernel
Feature map


Convolutional layers provides *translation invariance*. Since these filters works on every part of the image, they are "searching" for the same feature everywhere in the image.

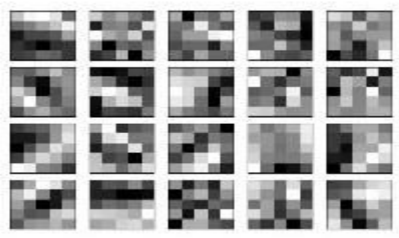


Figure 3.2 Convolution Layer

- **Pooling Layer**

It is common to periodically insert a Pooling layer in-between successive Convolution layers in ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control over-fitting. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations.

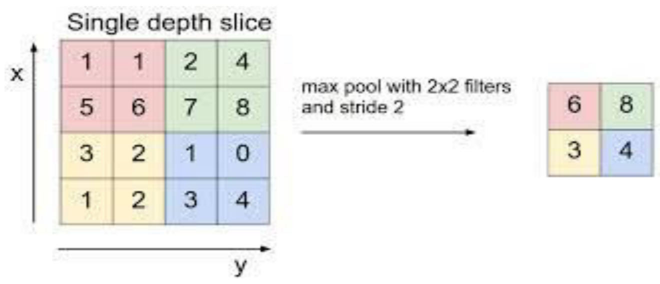


Figure 3.3 Pooling Layer

- **RELU Layer**

RELU layer will apply an element wise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged.

- **Fully-connected layer**

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. It is worth noting that the only difference between FC and CONV layers is that the neurons in the CONV layer are connected only to a local region in the input, and that many of the neurons in a CONV volume share parameters. However, the neurons in both layers still compute dot products, so their functional form is identical.

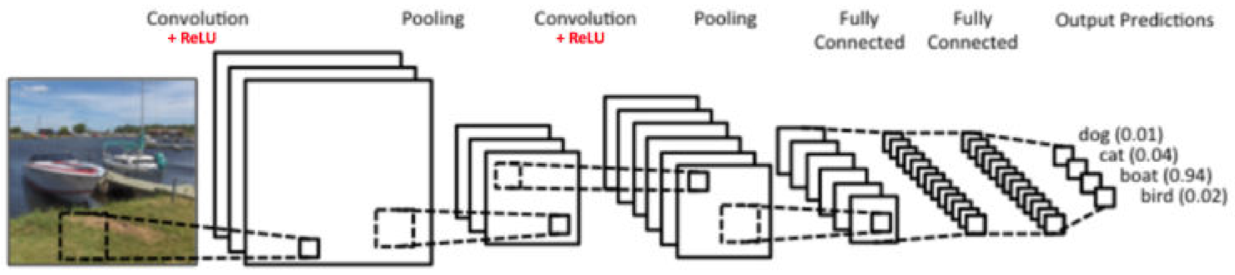


Fig 3.4 An Example of how Convolutional Neural Network Works.

CHAPTER 4

IMPLEMENTATION AND RESULT

4.1 Software and Hardware Requirements

Software:

The Following Software were used for this project:

1. Operating System: Kali GNU/Linux Rolling

2. Python 2.7.13 version (GCC 6.3.0)

3. Keras Library with TensorFlow and Theano backend Support

4. Python Library: Numpy, Scipy, PIL, sklearn, json, h5py

5. hdf5 (Hierarchical Data Format) for storing weights

6. Tomcat Server (tomcat7 for JSPs)

Hardware:

The Following hardware configuration is required to run the various softwares for this project.

1. Processor: Intel@ Core^{TM}i3 CPU

2. Memory: 4 GB RAM

3. Graphics Card: not needed

4.2 Feature Extraction

4.2.1 Convolution

ConvNets derive their name from the "convolution" operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

Following terminologies are used:

- **Depth**: Depth corresponds to the number of filters we use for the convolution operation. In our network , we are performing convolution of the original image using 32 distinct filters, thus producing 32 different feature maps . We can think of these three feature maps as stacked 2d matrices, so, the 'depth' of the feature map would be 32.

- **Stride**: Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.

- **Zero-padding**: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A nice feature of zero padding is that it allows us to control the size of the feature maps. Adding zero-padding is also called wide convolution, and not using zero-padding would be a narrow convolution.

Here we are using 32 random filters which are trained using backpropogation technique.

Let:

No of filters =32,

Size of pooling area for max pooling=2,

Convolutional Kernel size=3,

Now, we will use Conv2D which is method of keras.layers,taking the above parameters as argument and adding convolutional layer to the network.

```
model = Sequential()

model.add(Conv2D(nb_filters, (nb_conv, nb_conv) , padding='valid',input_shape=(
img_rows, img_cols,1)))
```

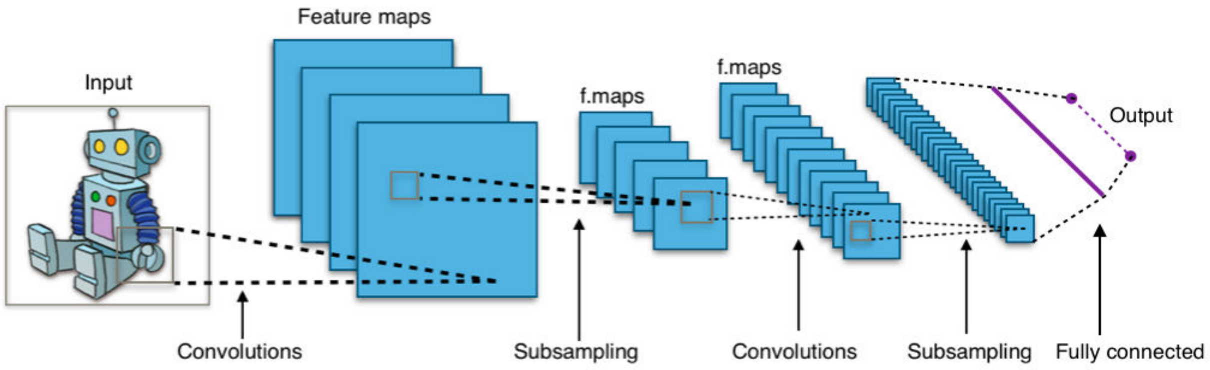


Figure 4.1 Convolutional Neural Network

4.2.2 Non Linearity (ReLU)

ReLU stands for Rectified Linear Unit and is a non-linear operation.ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.

The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

Its output is given by:

Output = Max(zero, Input)

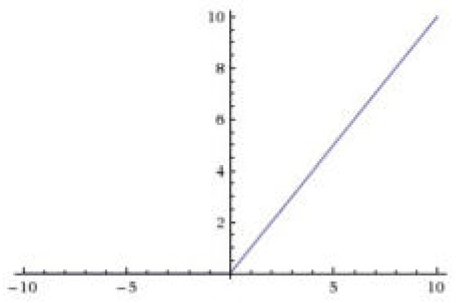


Figure 4.2 Output of Relu

4.2.3 The Pooling Step

Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

In particular, pooling:

1. Makes the input representations (feature dimension) smaller and more manageable.

2.Reduces the number of parameters and computations in the network, therefore controlling over fitting.

3. Makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in input will not change the output of Pooling – since we take the maximum / average value in a local neighbourhood).

4. Helps us arrive at an almost scale invariant representation of our image (the exact term is “equivariant”). This is very powerful since we can detect objects in an image no matter where they are located.

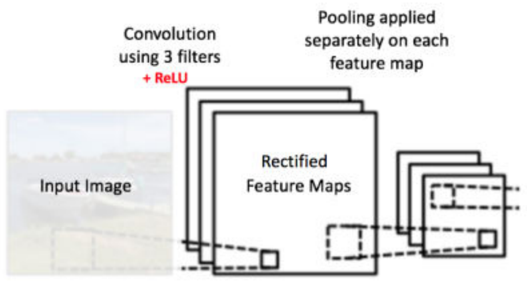

Convolution using 3 filters + ReLU
Pooling applied separately on each feature map
Input Image
Rectified Feature Maps

Figure 4.3 Pooling Layer

4.3 Training Process.

The overall training process of the Convolution Network is as below:

Step1: We initialize all filters and parameters / weights with random values

Step2: The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class.

Let's say the output probabilities for the images are [0.2, 0.4, 0.1, 0.3]

Since weights are randomly assigned for the first training example, output probabilities are also random.

Step3: Calculate the total error at the output layer (summation over all the defined classes)

Total Error = ∑ ½ (target probability – output probability) ^{2}

Step4: Use Backpropagation to calculate the gradients of the error with respect to all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output error.

The weights are adjusted in proportion to their contribution to the total error.

When if the same image is input again, output probabilities might now be [0.1, 0.1, 0.7, 0.1], which is closer to the target vector [0, 0, 1, 0].

This means that the network has learnt to classify this particular image correctly by adjusting its weights / filters such that the output error is reduced.

Parameters like number of filters, filter sizes, architecture of the network etc. have all been fixed before Step 1 and do not change during training process – only the values of the filter matrix and connection weights get updated.

Step5: Repeat steps 2-4 with all images in the training set.

4.4 Implementing Backpropagation

The backward propagation of errors or backpropogation is a common method of training artificial neural networks and used in conjunction with an optimization method such as gradient descent. The algorithm repeats a two phase cycle, propagation and weight update. When an input vector is presented to the network, it is propagated forward through the network, layer by layer, until it reaches the output layer. The output of the network is then compared to the desired output, using a loss function, and an error value is calculated for each of the neurons in the output layer. The error values are then propagated backwards, starting from the output, until each neuron has an associated error value which roughly represents its contribution to the original output.

Backward pass through the network, is determining which weights contributed most to the loss and finding ways to adjust them so that the loss decreases. Once we compute this derivative, we then go to the last step which is the **weight update**. This is where we take all the weights of the filters and update them so that they change in the direction of the gradient.

$$w = w_i - \eta \frac{dL}{dW}$$

w = Weight
w_i = Initial Weight
η = Learning Rate

Figure 4.4 Updating Weights

The **learning rate** is a parameter that is chosen by the programmer. A high learning rate means that bigger steps are taken in the weight updates and thus, it may take less time for the model to converge on an optimal set of weights. However, a learning rate that is too high could result in jumps that are too large and not precise enough to reach the optimal point.

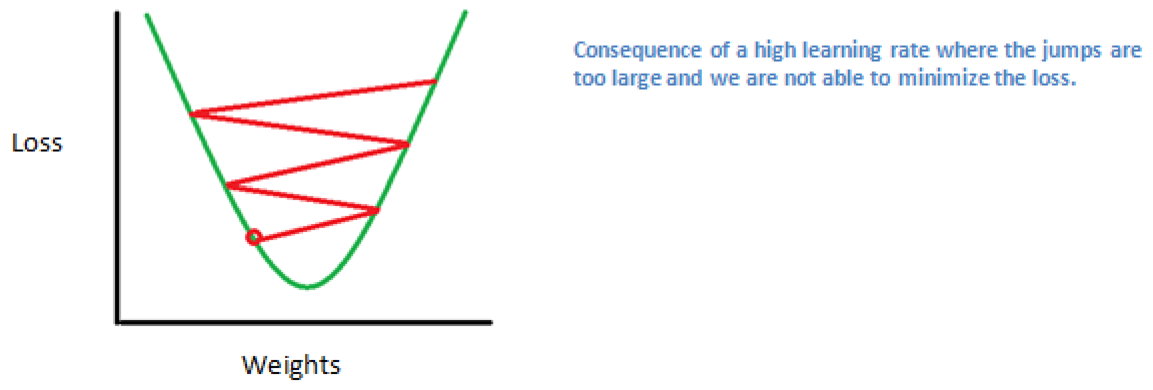

Loss
Weights
Consequence of a high learning rate where the jumps are too large and we are not able to minimize the loss.

Figure 4.5 Loss variation according to weights

The process of forward pass, loss function, backward pass, and parameter update is generally called one **epoch**. The program will repeat this process for a fixed number of epochs for each set of training images (commonly called a batch). Once you finish the parameter update on the last training example, hopefully the network should be trained well enough so that the weights of the layers are tuned correctly.

- **Saving Weights**

  ```
  fname = "weights-Test-CNN.hdf5"
  model.save_weights(fname,overwrite=True)
  ```

- **Loading Weights**

  ```
  fname = "weights-Test-CNN.hdf5"
  model.load_weights(fname)
  ```

4.5 Classification

A classifier is a supervised function (machine learning tool) where the learned (target) attribute is categorical. It is used after the learning process to classify new records (data) by giving them the best target attributes (prediction). In this project, we are using Keras classifier, which is provided by the Keras library. The keras classifier uses the sequential model. Here we are using different layers like convolution,Relu, Max Pooling and in the final layer, we are using a softmax function as activation function. Layers are added into the network as explained below:

1. **Convolution Layer :**

```
model.add(Convolution2D(nb_filters, nb_conv, nb_conv, border_mode='valid', input_shape=(1, img_rows, img_cols)))
```

2. **Relu Layer :**

```
convout1 = Activation('relu')
model.add(convout1)
```

3. **Max Pooling Layer :**

```
model.add(MaxPooling2D(pool_size=(nb_pool, nb_pool)))
```

4. **Softmax Layer :**Softmax function, or normalized exponential function is a generalization of the logistic function that "squashes" a K-dimensional vector **z**of arbitrary real values to a K-dimensional vector **σ(z)** of real values in the range (0, 1) that add up to 1.

```
model.add(Activation('softmax'))
```

After adding the layers the probability generated by this classifier is then used for predicting class labels which are then mapped to sentences.

```
model=sequential()
model.predict_classes(self, x, batch_size=32, verbose=1)
```

4.6 Performance Parameters

- **Accuracy:**

Accuracy of the network is defined as follows:

Accuracy = (No. of labels predicted correctly) / (Total no. of test images)*100

- **Loss function**

The loss function is a function that maps values of one or more variables onto a real number intuitively representing some "cost" associated with the event. For backpropagation, the loss function calculates the difference between the input training example and its expected output, after the example has been propagated through the network.

Let y, y' be vectors in R ^n.

Select an error function E(y,y') measuring the difference between two outputs.

The standard choice is:

$$E(y,y')= 0.5*(y-y')\ \hat{}2,$$

the square of the Euclidean distance between the vectors y and y'.

The factor of 0.5 conveniently cancels the exponent when the error function is subsequently differentiated.

The error function over n training examples can be written as an average:

$$\mathbf{E = 0.5 * \sum(y(x) - y'(x))^2}$$

And the partial derivative with respect to the output is :

$$\mathbf{dE/dy' = (y' - y)}$$

- **Factors affecting accuracy:**

1. **Number of Images in the Dataset:** Accuracy increases significantly with the increase in number of images in the training phase.

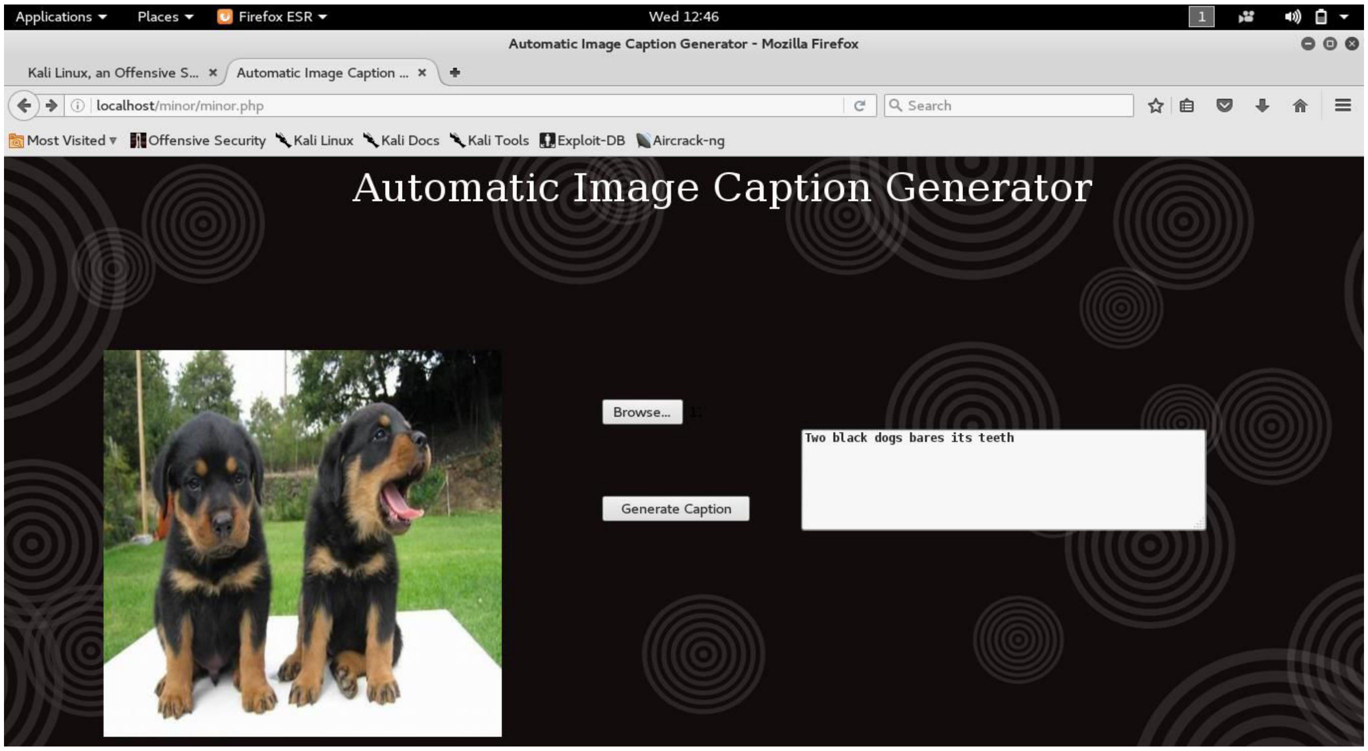
   Ex: For 200 images Accuracy was 28.2, and for 800 images accuracy was found to be 35.6%

2. Number of Epoch: Accuracy increases as we increase the number of epoch up to a certain value after which accuracy starts decreasing.

3. Number of Hidden Layers: Accuracy increases as we increase the no. of hidden layers in the network but training time also increases with it.

4.7 Result:

Following is the screenshot of the GUI of the project. It takes the image as input and generates the caption.


Applications
Places
Firefox ESR
Wed 12:46
Automatic Image Caption Generator - Mozilla Firefox
Kali Linux, an Offensive S...
Automatic Image Caption ...
localhost/minor/minor.php
Search
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng
Automatic Image Caption Generator
Browse...
Generate Caption

Two black dogs bares its teeth

Figure 4.6 Screenshot of the GUI

Table 4.1 Accuracy and Loss Values

S. no.	Total no. of Images	No. of Training Images	No. of Testing Images	No. of Epochs	Batch Size	Accuracy (approx.)	Loss
1.	80	64	16	20	5	12%	7.56%
2.	850	680	70	20	5	28%	5.23%
3.	2250	1800	450	20	5	36%	2.19%
4.	8000	6400	1600	20	5	52%	1.08%

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

We have studied about generating the captions from image and came across three methods to do this – Template based, Retrieval based and Neural Network based. Out of this, we have applied the Neural Network based approach to generate the captions. We have used Flickr 8k dataset for training the neural network which has 5 captions for each image present in the dataset. In the training phase, we have extracted the features for the image present in the dataset using convolutional neural network. The extracted features are then given as input to the keras classifier which generates a class labels which are mapped to the sentences. Error is then calculated and weights are adjusted accordingly. These trained weights are used to predict the caption for the input images. This model gives the caption for images given as input to it. The current model used is able to generate sentences for the simple images quite accurately and has given quite good results during testing phase. We can further improve the accuracy of the model by training it with the Flickr 30K dataset.

5.2 Future Work

Possible future applications of automatic image caption generator are:

- **Smart Goggles:** Smart Goggles can be developed for visually impaired people so that they will get real time updates of surroundings.

- **Pre knowledge:** A pre-knowledge about the contents of the image can be gained before image loaded in WebPages.

- **Skin Vision:** Lets you confirm whether a skin condition can be skin cancer or not.

- **Text driven fields:** Probably, will be useful in cases/fields where text is most used and with the use of this, you can infer/generate text from images. As in, use the information directly from any particular image in a textual format automatically

- **Video captioning:** can be used for explaining what happens in a video, frame by frame.

REFERENCES

- Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3128-3137, 2015.

- Oriol Vinyals, Alexander Toshev, SamyBengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3156-3164, 2015.

- Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and YoshuaBengio. Show, attend and tell: Neural image caption generation with visual attention.arXiv preprint arXiv:1502.03044, 2015.

- Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. arXiv preprint arXiv:1511.07571, 2015.

- **Keras documentation:**

  URL address - https://keras.io/#keras-deep-learning-library-for-theano-and-tensorflow

- **Flickr 8K Dataset** :

  The Flickr 8K dataset includes images obtained from the "flickr" website. Use of such images must respect the flickr terms of use.

  We do not own the copyright of the images. We solely provide the link below for researchers and educators who wish to use the dataset for non-commercial research and/or educational purposes.

  URL address - https://illinois.edu/fb/sec/1713398

- A Neural Network in 11 lines of Python by iamtrask

  URL address - http://iamtrask.github.io/2015/07/12/basic-python-network