**IBM Naan Mudhalvan**

**Cloud Application Development**

**(Image recognition and IBM**

**Cloud visual recognition)**

**Phase-5 Project**

**Introduction**

Artificial intelligence (AI) becomes the driving force behind evolving technologies such as IoT (Internet of Things) [1], sensor networks [2], robotics, and big data. Automatic image captioning is also an emerging interdisciplinary field of research in AI, related to NLP (natural language processing) and computer vision. The idea of an automatic caption generation system is to create a relevant and meaningful description for an image in an appropriate natural language format. The automatic caption generation system has an extensive array of applications such as intelligent human-machine interactions, supporting visually impaired people, and image indexing. Image caption generation is a tedious process of identifying various objects present in the image and the association between those objects, attributes, and behavior.

Recently developed deep learning-based caption generation techniques [7–9] considers only the factual and semantic content of the image. But, the emotional perspective of the image plays a vital role in generating high-quality descriptions more intelligently. To capture the emotional aspects of the image, the Face-Cap model is introduced by Nezami et al. [10]. The system can detect and derive the facial expression features and apply these features to generate captions for the image. The system extracted the features of facial expression from the image using a FER technique. Additionally, the system considers the features from the ImageNet dataset trained using the Oxford Visual Geometry Group network (VGGnet) [11] with a weighted attention mechanism. Finally, to generate the caption, the system utilized a long short-term memory (LSTM) network. Inspired by this work, the proposed system embeds the emotional analysis in the image caption generation model to automatically generate descriptions based on both the emotional features and salient image features. The proposed caption generation model employs CSPDenseNet [12] and the FER model [10] to extract image features and emotion features and introduced a self-attentive BiLSTM (bidirectional long short-term memory) network to describe the image more effectively. The contributions of the proposed work are as follows:(1)Extracts more extensive and salient image features using CSPDenseNet instead of the standard CNN model and generates an image representation vector.(2)Self-attentive BiLSTM model is introduced to generate captions. The BiLSTM model process the textual knowledge from both the forward direction and the backward direction. The self-attention mechanism is implemented for improving the quality of caption generation by focusing on the important text features as well as the contextual features.(3)The experimental outcomes exhibit that the newly designed caption generation system is capable of describing an image with better quality in comparison to the existing image captioning models.

The remainder of the article is organized in the following sections. Section [2](https://www.hindawi.com/journals/acisc/2022/2756396/#sec2) examines the related works. The detailed description of the proposed model is explored in section [3](https://www.hindawi.com/journals/acisc/2022/2756396/#sec3). The experimental settings, results, and discussion are illustrated in section [4](https://www.hindawi.com/journals/acisc/2022/2756396/#sec4). Finally, section [5](https://www.hindawi.com/journals/acisc/2022/2756396/#sec5) concludes the present article with a summary.

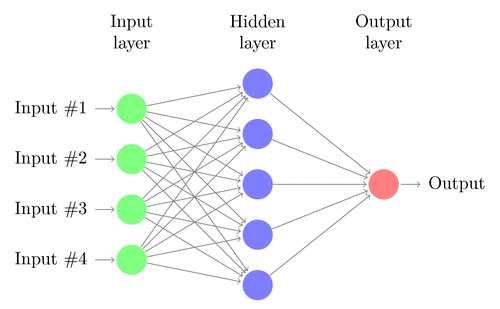
**objective:**

* + - * Install tool which used for image recognition process like cnn
      * Create a images(dataset) like human faces with emotions
      * Write a code which works with capturing humans emotion and generate a caption with respect to the image

A convolutional neural network(CNN) is a type of **Artificial Neural Network(ANN)** used in image recognition and processing which is specially designed for processing data(pixels).

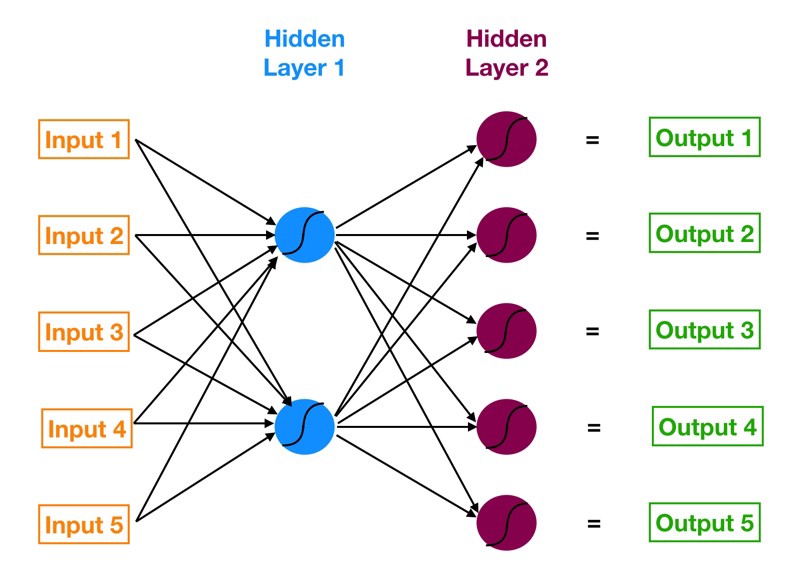
## Neural Network:

A neural network is constructed from several interconnected nodes called **“neurons”**.  Neurons are arranged into the**input layer, hidden layer, and output layer.** The input layer corresponds to our predictors/features and the Output layer to our response variable/s.



**Multi-Layer Perceptron(MLP):**

The neural network with an input layer, one or more hidden layers, and one output layer is called a **multi-layer perceptron (MLP).** MLP is Invented by **Frank Rosenblatt** in the year of 1957. MLP given below has 5 input nodes, 5 hidden nodes with two hidden layers, and one output node



**How does this Neural Network work?**

– Input layer neurons receive incoming information from the data which they process and distribute to the **hidden layers**.

– That information, in turn, is processed by hidden layers and is passed to the output **neurons**.

– The information in this artificial neural network(ANN) is processed in terms of one **activation function**. This function actually imitates the brain neurons.

– Each neuron contains a value of **activation functions** and a **threshold value.**

– The **threshold value** is the minimum value that must be possessed by the input so that it can be activated.

– The task of the neuron is to perform a weighted sum of all the input signals and apply the activation function on the sum before passing it to the next(hidden or output) layer.

# image recognition

Image recognition, in the context of [machine vision](https://www.techtarget.com/searchenterpriseai/definition/machine-vision-computer-vision), is the ability of software to [identify objects](https://www.techtarget.com/whatis/definition/object-recognition), places, people, writing and actions in digital images. Computers can use machine vision technologies in combination with a camera and artificial intelligence ([AI](https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence)) software to achieve image recognition.

Steps to achieve Image recognition:

* Collecting the Dataset
* Importing Libraries and Splitting the Dataset
* Building the CNN
* Full Connection
* Data Augmentation
* Training our Network
* Testing

# **Step 1 — Collecting the Dataset**

In order to train our machine, we need a huuuuggge amount of data so that our model can learn from them by identifying out certain relations and common features related to the objects.

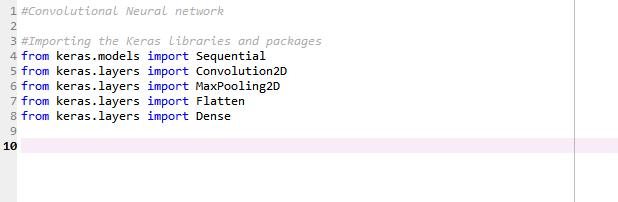
For example consider:



In above we have images of apples and oranges these set of images are required for the image recognition program to be identify as one of them

# **Step 2 — Importing Libraries and Splitting the Dataset**

To use the powers of the libraries, we first need to import them.



After importing the libraries, we need to split our data into two parts- taining\_set and test\_set.

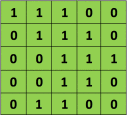
# **Step 3 — Buliding the CNN**

This is most important step for our network. It consists of three parts

* Convolution
* Polling
* Flattening

The primary purpose of Convolution 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.

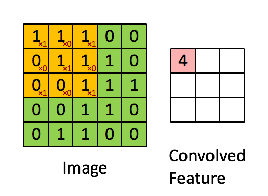
Since every image can be considered as a matrix of pixel values. Consider a 5 x 5 image whose pixel values are only 0 and 1 (note that for a grayscale image, pixel values range from 0 to 255, the green matrix below is a special case where pixel values are only 0 and 1):



Also, consider another 3 x 3 matrix as shown below:

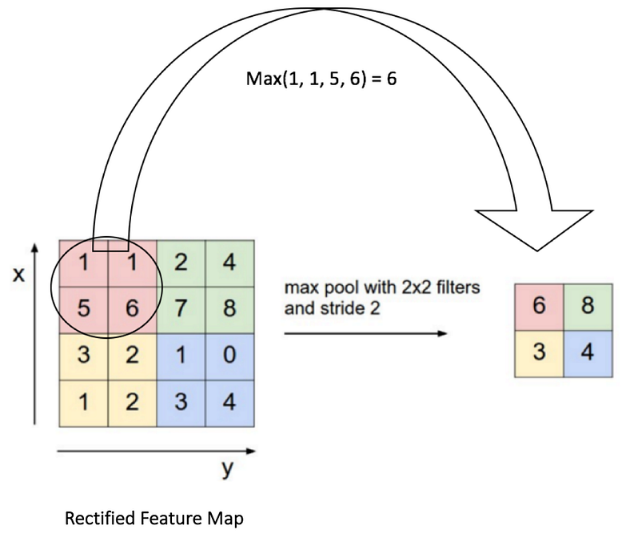


Then, the Convolution of the 5 x 5 image and the 3 x 3 matrix can be computed as shown in the animation in **Figure 5** below:



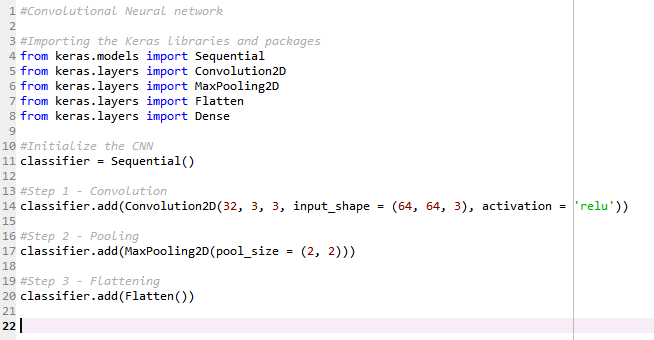
The obtained matrix is also known as the feature map. An additional operation called ReLU is used after every Convolution operation. The next step is of pooling.

Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. In case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.

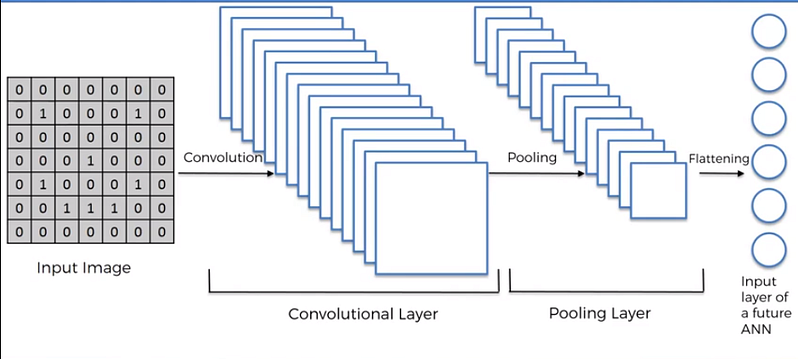


After pooling comes flattening. Here the matrix is converted into a linear array so that to input it into the nodes of our neural network.

Let’s come to the code.

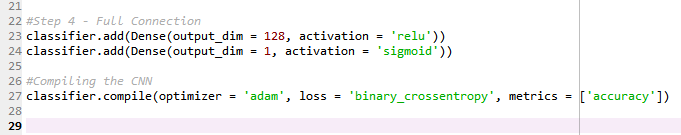


So now our CNN network looks like this



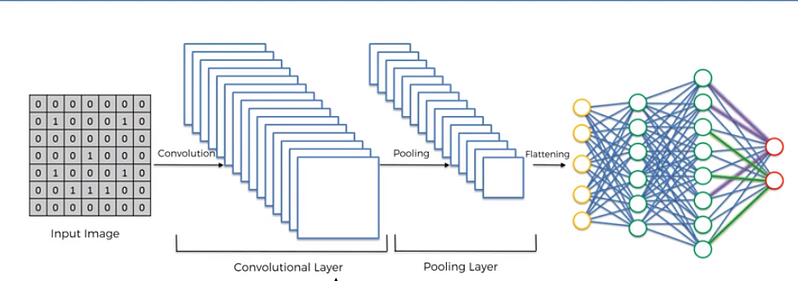
# **Step 4 — Full Connection**

Full connection is connecting our convolutional network to a neural network and then compiling our network.



Here we have made 2 layer neural network with a sigmoid function as an activation function for the last layer as we need to find the probability of the object being a cat or a dog.

So now the final network looks something like this -



# **Step 5 — Data Augmentation**

While training your data, you need a lot of data to train upon. Suppose we have a limited number of images for our network. What to do now??

You don’t need to hunt for novel new images that can be added to your dataset. Why? Because, neural networks aren’t smart to begin with. For instance, a poorly trained neural network would think that these three tennis balls shown below, are distinct, unique images.



The same tennis ball, but translated.

So, to get more data, we just need to make minor alterations to our existing dataset. Minor changes such as flips or translations or rotations. Our neural network would think these are distinct images anyway.

Data augmentation is a way we can reduce overfitting on models, where we increase the amount of trainingdata using information only in our training data. The field of data augmentation is not new, and in fact, various data augmentation techniques have been applied to specific problems.

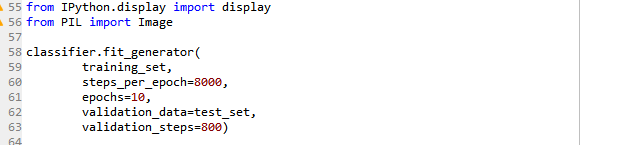
And here goes the code



Now we have a huge amount of data and its time for the training.

# **Step 6 — Training our Network**

So, we completed all the steps of construction and its time to train our model.



If you are training with a good video card with enough RAM (like an Nvidia GeForce GTX 980 Ti or better), this will be done in less than an hour. If you are training with a normal cpu, it might take a lot longer.

With increasing number of epochs, the accuracy will increase.

# Step 7 — Testing

Now lets test a random image.



And the code will be

import urllib

import requests

import os

 # retrieving using image url

urllib.request.urlretrieve("https://i.ibb.co/xY4DJJ5/img1.jpg", "img1.jpg")

urllib.request.urlretrieve("https://i.ibb.co/Gnd1Y1L/img2.jpg", "img2.jpg")

urllib.request.urlretrieve("https://i.ibb.co/Z6JgS1L/img3.jpg", "img3.jpg")

 print('Images downloaded')

 # get current working directory path

path = os.getcwd()

captionarr = [

    "This is the first caption",

    "This is the second caption",

    "This is the third caption"

    ]

# importing necessary functions from PIL

from PIL import Image

from PIL import ImageFont

from PIL import ImageDraw

 # print(os.getcwd())

 # checking the file mime types if

# it is jpg, png or jpeg

def ext(file):

    index = file.find(".jpg")

    current\_file = ""

    current\_file = file[index:]

    return current\_file

 def ext2(file):

    index = file.find(".jpeg")

    current\_file = ""

    current\_file = file[index:]

    return current\_file

def ext3(file):

    index = file.find(".png")

    current\_file = ""

    current\_file = file[index:]

    return current\_file

 # converting text from lowercase to uppercase

def convert(words):

    s = ""

    for word in words:

        s += word.upper()

    return s

 caption\_first = convert(captionarr[0])

caption\_second = convert(captionarr[1])

caption\_third = convert(captionarr[2])

print(caption\_first)

print(caption\_second)

print(caption\_third)

count = 0

 for f in os.listdir('.'):

    try:

        # Checking for file types if jpg, png

        # or jpeg excluding other files

        if (ext(f) == '.jpg' or ext2(f) == '.jpeg' or ext3(f) == '.png'):

            img = Image.open(f)

            width, height = img.size

            basewidth = 1200

            # print(height)

 # Resizing images to same width height

            wpercent = (basewidth / float(img.size[0]))

            hsize = int((float(img.size[1])\*float(wpercent)))

            img = img.resize((basewidth, hsize), Image.ANTIALIAS)

            new\_width, new\_height = img.size

            # print(new\_height)

            # changing image mode if not in RGB

            if not img.mode == 'RGB':

                img = img.convert('RGB')

            draw = ImageDraw.Draw(img)

            # font = ImageFont.truetype(<font-file>, <font-size>)

            # initializing which font will be chosen by us

            font = ImageFont.truetype("Arial Bold.ttf", 35)

             # First Caption on First image

            if count == 0:

                draw.text((new\_width / 15 + 25, new\_height - 100),

                           caption\_first, (255, 0, 0), font = font,

                           align ="center")

            # Second Caption on Second image

            elif count == 1:

                draw.text((new\_width / 15 + 25, new\_height - 100),

                          caption\_second, (255, 0, 0), font = font,

                          align ="center")

            # Third Caption on Third image

            else:

                draw.text(( new\_width / 15 + 25, new\_height - 100),

                            caption\_third, (255, 0, 0), font = font,

                            align ="center")

img.save("CaptionedImges/{}".format(f))

            print('done')

            count = count + 1

    except OSError:

        pass

import os

import glob

import shutil

# changing directory to CaptionedImages

os.chdir(".\\CaptionedImges")

fnames = []

for file in os.listdir('.'):

    # appending files in directory to the frames arr

    fnames.append(file)

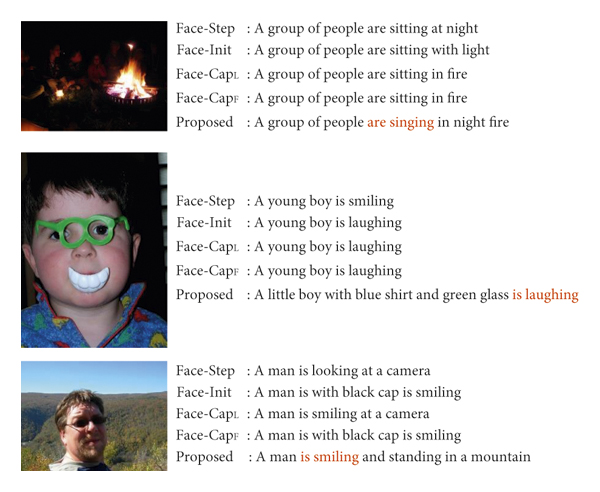
# sorting the files in frames array

# on the basis of last modified time

# reverse = True means ascending order sorting

fnames.sort(key = lambda x: os.stat(x).st\_ctime, reverse = True)

[[](https://www.hindawi.com/journals/acisc/2022/2756396/fig9/)](https://www.hindawi.com/journals/acisc/2022/2756396/fig9/" \t "_blank)

[[](https://www.hindawi.com/journals/acisc/2022/2756396/fig10/)](https://www.hindawi.com/journals/acisc/2022/2756396/fig10/" \t "_blank)

**Conclusion**

Caption generation with emotions paved the way to explore a wider scope of image captioning applications. In this work, a novel encoder-decoder architecture for the automatic caption generation process is introduced by incorporating human emotions extracted from facial features. Human facial emotions are captured and emotion feature vectors are created using a FER model trained on the FER-2013 dataset. CSPDenseNet, a new variant of CNN is adopted for encoding the image to capture the more intense feature vectors. The cross-stage feature fusion strategy is employed in CSPDenseNet which makes it lightweight to run on CPUs. A Word2vec model is developed and trained using the human-annotated captions to extract the word feature vectors. Finally, the extracted emotion, image, and word feature vectors are fused and fed into the language decoder. The language decoder employed BiLSTM with self-attention to produce the captions. The BiLSTM network is implemented to extract the semantic knowledge and self-attention is incorporated to focus on the salient contextual features in the text. The experimental outcomes demonstrated that the model proposed in this article generates image captions more effectively when compared to the state-of-art models in terms of BLEU, METEOR, CIDEr, ROUGE-L, and SPICE. The future work aims to develop novel facial expression recognition models that can capture a wider range of emotions.