Image Classification using Deep Learning

A report submitted to

RAMAIAH INSTITUTE OF TECHNOLOGY

Bengaluru

ISE741- Deep Learning (Miniproject)

as partial fulfillment of the requirement for

Bachelor of Engineering

by

Chethan M (USN- 1MS19IS401)
Ganesh N Hotti (USN- 1MS19IS404)
Shobha D (USN- 1MS19IS410)

under the guidance of

Dr. Sumana Maradithaya

Associate Professor

IS&E department

RAMAIAH
Institute of Technology

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

RAMAIAH INSTITUTE OF TECHNOLOGY

January 2022

Department of Information Science and Engineering

Ramaiah Institute of Technology

Bengaluru – 54

Ganesha N Hotti, 1MS19IS404, has given the presentation and submitted the report on 18/01/2022

Title: Image Classification using Deep Learning.

Signature of Examiner

Name :

Signature of Internal Guide

Name:

Abstract

Classification of objects into respective insightful groups have always been a very popular task in the area of machine learning and statistics. Yet another popular area of research these days is image processing and graphical learning. Most of the tasks in these domains of machine learning are always complex, seems conundrums in terms of classification objectives. There is a popular solution to these hard problems called artificial neural networks (ANNs). ANNs are always in discussions because of their complex structure and vast usefulness. In this project I will make use of convolutional neural network study to predict and classify different objects with the help of state of the art technology called Tenser Flow. Convolutional neural network (CNNs) is the competitive image recognition method which has been deeply exploited in the last years. It essentially mimics the local receptive capacity of the neurons as seen in the human brain, with sophisticated weight sharing and the linkage methods, significantly decreases the training constraints in comparison with traditional neural network approach.

Introduction

Classification is a systematic arrangement in groups and categories based on its features. Image classification came into existence for decreasing the gap between the computer vision and human vision by training the computer with the data. The image classification is achieved by differentiating the image into the prescribed category based on the content of the vision. Motivation by, in this paper, we explore the study of image classification using deep learning.

The conventional methods used for image classifying is part and piece of the field of artificial intelligence (AI) formally called as machine learning. The machine learning consists of feature extraction module that extracts the important features such as edges, textures etc and a classification module that classify based on the features extracted. The main limitation of machine learning is, while separating, it can only extract certain set of features on images and unable to extract differentiating features from the training set of data. This disadvantage is rectified by using the deep learning. Deep learning (DL) is a sub field to the machine learning, capable of learning through its own method of computing. A deep learning model is introduced to persistently break down information with a homogeneous structure like how a human would make determinations. To accomplish this, deep learning utilizes a layered structure of several algorithms expressed as an artificial neural system (ANN). The architecture of an ANN is simulated with the help of the biological neural network of the human brain. This makes the deep learning most capable than the standard machine learning models. In deep learning, we consider the neural networks that identify the image based on its features. This is accomplished for the building of a complete feature extraction model which is capable of solving the difficulties faced due to the conventional methods. The extractor of the integrated model should be able to learn extracting the differentiating features from the training set of images accurately.

Literature survey

Deep Learning for Image Classification

Image classification is a well-known classical problem in multimedia content analysis. This paper proposes a novel deep learning model called bilinear deep belief network (BDBN) for image classification. Unlike previous image classification models, BDBN aims to provide human-like judgment by referencing the architecture of the human visual system and the procedure of intelligent perception. Therefore, the multi-layer structure of the cortex and the propagation of information in the visual areas of the brain are realized faithfully. Unlike most existing deep models, BDBN utilizes a bilinear discriminant strategy to simulate the "initial guess" in human object recognition, and at the same time to avoid falling into a bad local optimum. To preserve the natural tensor structure of the image data, a novel deep architecture with greedy layer-wise reconstruction and global fine tuning is proposed. To adapt real-world image classification tasks, we develop BDBN under a semi-supervised learning framework, which makes the deep model work well when labelled images are insufficient. Comparative experiments on three standard datasets show that the proposed algorithm outperforms both representative classification models and existing deep learning techniques.

In the conventional image recognition methods, CNNs uses the multilayer convolution to do the feature engineering and combine these features internally. It also uses the pooling layer, and the fully connected layer along with softmax. Google's Tenser Flow an open source library can be used for the arithmetic calculations, specialized for machine learning computations. Recently launched second generation of the Google's artificial intelligence learning system got more views and appreciation in the scientific community in the field of machine learning from all over the world. Tenser Flow has many advantages such as high

flexibility, high accessibility, and the great provision of coders and Tenser Flow practitioners on the GitHub makes this implementation task smoother.

One of the traditional methods of image recognition is the use of elastic graphs, in which the grid structure is used to create templates with which different images are compared within the bound of grids. In this approach specifically the recognition performance is better, compromising the execution speed and execution time. There has been several other investigations on object recognition using arithmetic mean based Hidden Markov Model (HMM) techniques where the probabilistic estimation of the single value feature of objects are transformed into vectors and then HMM is used on the top to predict the image. There is an another method as well in the scope of machine learning optimization called the Eigen face method, which is based on Principal Component Analysis (PCA), using PCA transform to reduce the original image processing, and then to classify and identify. Also the Fisher face method based on Linear Discriminant Analysis

(LDA), the features of the image are reduced and then LDA is used to transform and extract the features of the principal components after the dimension reduction. It is expected that large inter-class divergence and small intra-class divergence. Talking about the Mobile Nets is one of the already trained models on the Tensor Flow. It is regarded as routine development to the initial structure of computer graphics learning after Inception-v1, Inception-v2, and Inception-v3 in the year 2015. The Mobile Net model is trained on the popular ImageNet data sources, consisting of the 1000s of categories in ImageNet, the fault percentage of top-5 is up to 3.5% leaving behind average accuracy portion of ~90% in desired cases, the fault percentage of top-1 dropped to 17.3%. Following section we will be discussing the different state of the art convolutional neural learning approaches

Proposed model

This article describes how to establish the neural network technique for various image groupings in a convolution neural network (CNN) training. Image classification is a well-known classical problem in multimedia content analysis. In addition, it also suggests initial classification results using CNN learning characteristics and classification of images from different categories. To determine the correct architecture, we explore a transfer learning technique, called Fine-Tuning of Deep Learning Technology, a dataset used to provide solutions for individually classified image-classes

Detailed Design

All the neurons in the preceding layers as well as all neurons in the layer directly after it. These connections have a specific weight, which will be manipulated later on in the training process, attributed to them [2]. The value in a neuron will be the sum of all of the values in the signal multiplied by their individual weights, all summed together.

Information flows from the input layers towards the output layers. At first, random weights are used but errors are minimized using a process called backpropagation. In backpropagation, the error is computed at the output and then the neural network's weights in the connections are manipulated to minimize the error. This process starts from the outputs, moves backwards through the hidden layers, and ends with the input layer. This process minimizes error and successfully trains the system.

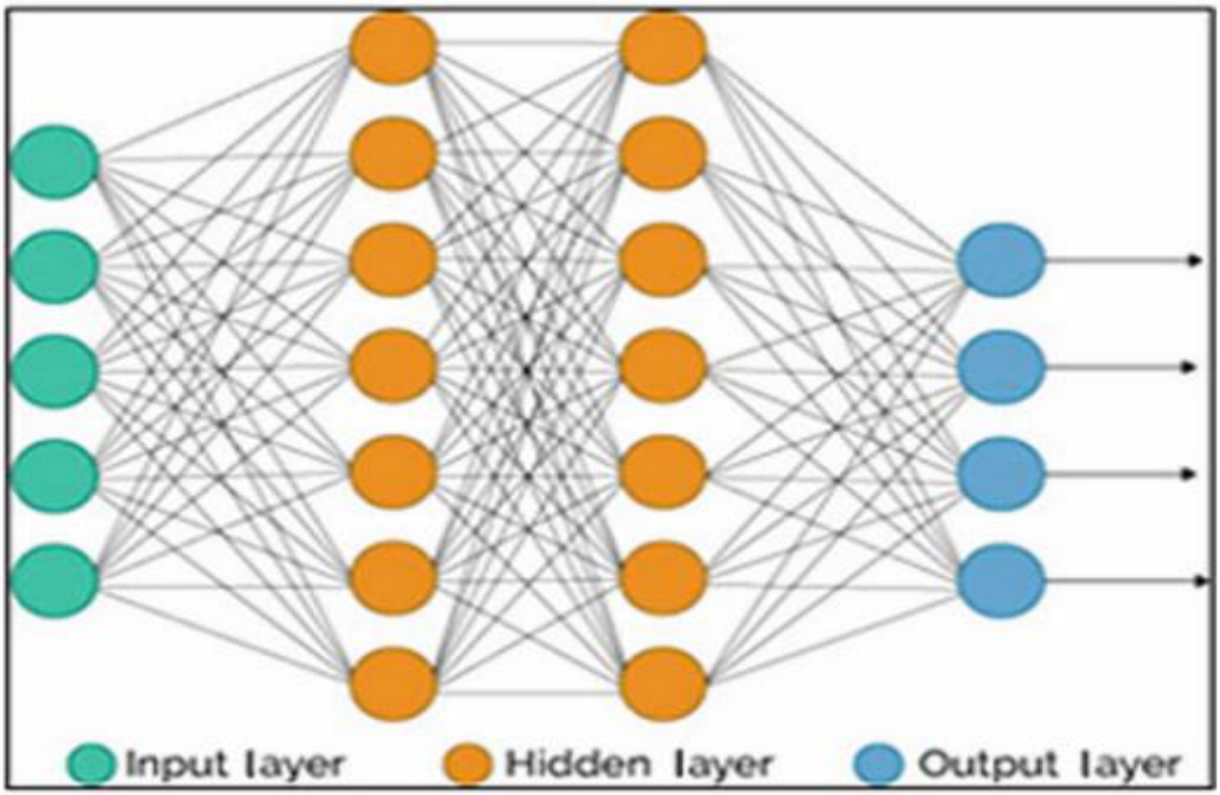


Fig A.Artificial Neural Networks with two hidden layers

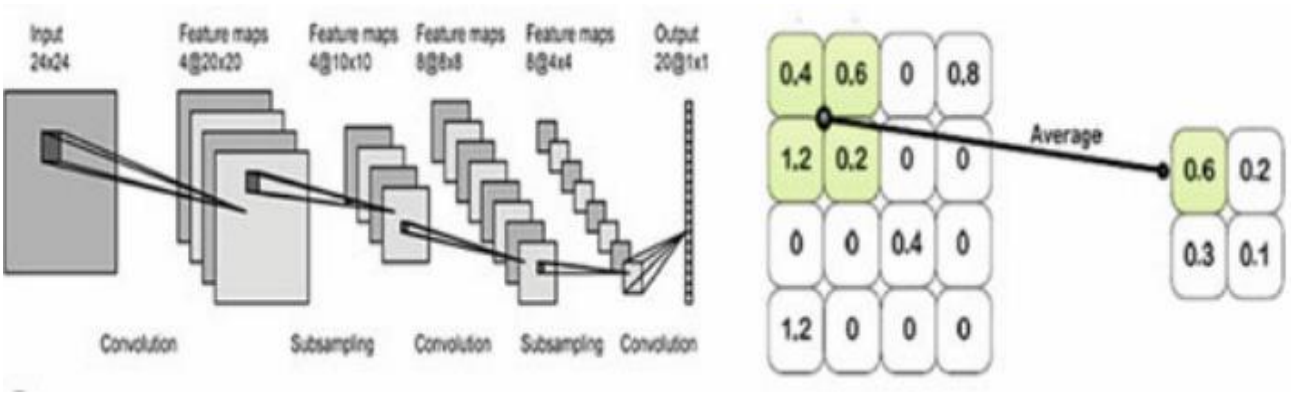


Fig B. CNN architecture

INPUT -> FC Implements a linear classifier

INPUT -> CONV -> ReLU -> FC

INPUT -> [CONV -> ReLU -> POOL] * 2 -> FC -> ReLU -< FC

(Single CONV layer between every pool)

INPUT -> [CONV -> ReLU -> CONV -> ReLU -> Pool] * 3 -> [FC -< ReLU] * 2 -> FC

(Here there are two CONV layers stacked before every pool)

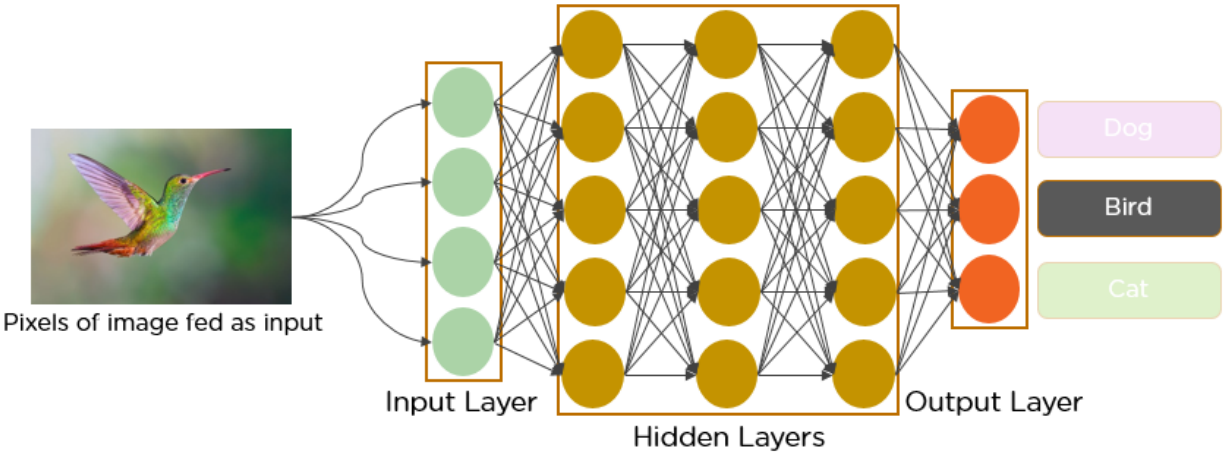
Fig C. CNN architecture Convolutional layers stacked before each pool

A fully connected neural network transforms inputs to outputs. We call it fully connected because the input affects the output. The layers possess many learnable parameters and assume no structure in the input. In convolutional neural networks (CNN), the inputs are closer and semantically related. Therefore, CNN is more suitable for image classification. The deep neural network (DNN) architecture (a CNN model) requires a significant number of hidden layers, input parameters, and a sufficient number of images for training. DNN provides the breakthrough architecture in image classification, speech recognition, and natural language processing. Fundamentally, DNN possesses series of layers between input and output for feature identification, similar to

our brain (process in series of stages). The existing algorithms are good for learning weights in neural networks with one or two layers. Current algorithms are not suitable for more hidden layers since they need to make thousands of adjustments to make the network slightly better. CNN works in such situations. In the image analysis, the relative position is significant. In CNN, each image is represented as numbers (each number is a pixel). We apply a series of operations to conclude the probability of closeness to original object. The network contains multiple layers, including: convolution, rectified linear units (ReLU), pooling, fully connected and loss. Suppose the input is a 28x28 area. The input area and other areas consist of grids and resulting grids. The grid has several neurons and takes input from all grids of the previous layer. The weights for each neuron in the current section are the same. After each CNN layer, there is a pooling layer (filter) to produce a resolution of the future map. In pooling, each grid creates a value. The value may be average or maximum or a linear combination of grid values. Figure B shows the CNN architecture and the example of maximum pooling. The pooling layer controls overfitting of the number of parameters and the amount of computation. The pooling layer operates independently on every depth slice of input and sizes spatially. Recent trends tried to discard the pooling filter due to the control of the size of the CNN architecture and used many other filters. The Rectified Linear Units (ReLU) increase the nonlinear properties of the decision function and overall network, without affecting the receptive fields of the convolution layer.

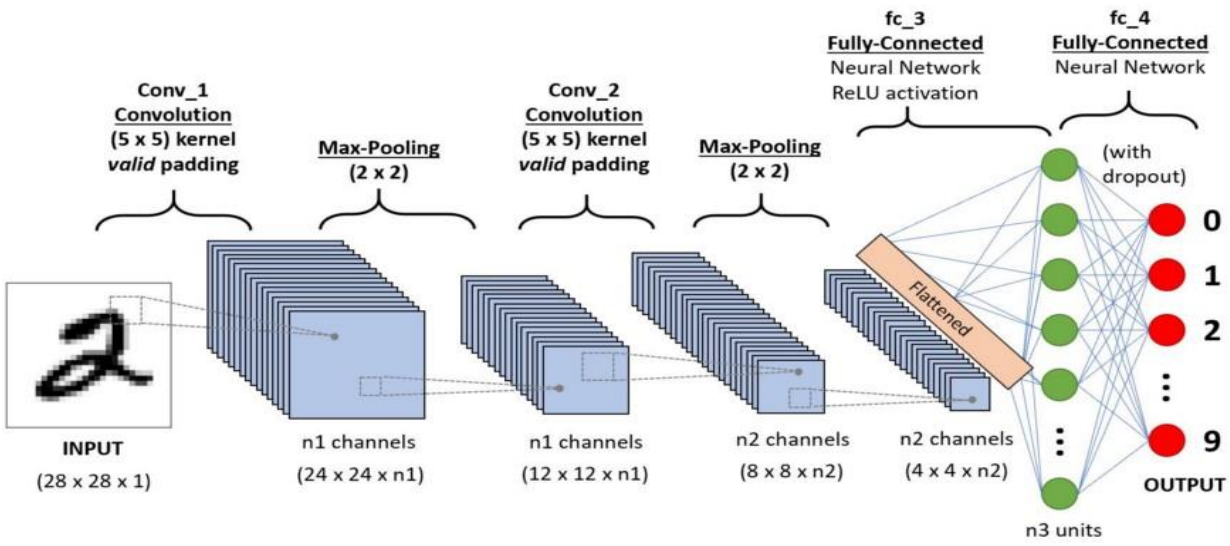
The process of input to fully connected (FC) is shown in Fig C.

Coding and Implementation



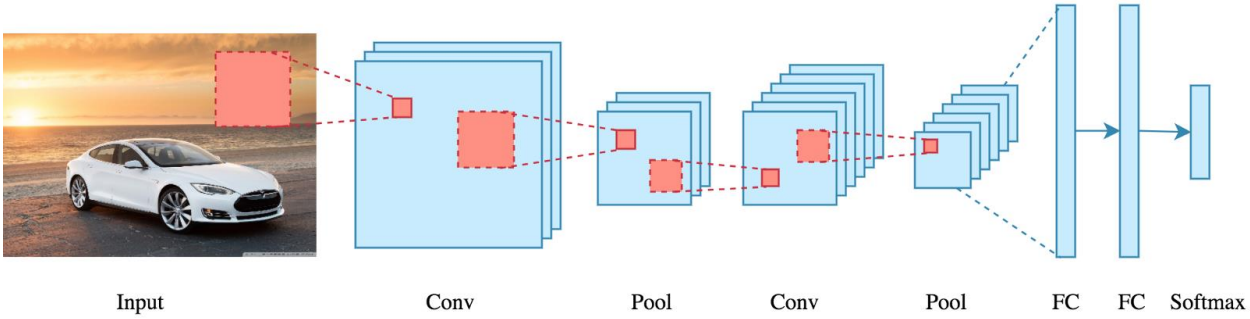
Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks.

Since the 1950s, the early days of AI, researchers have struggled to make a system that can understand visual data. In the following years, this field came to be known as Computer Vision. In 2012, computer vision took a quantum leap when a group of researchers from the University of Toronto developed an AI model that surpassed the best image recognition algorithms and that too by a large margin.



CNN's were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. This was a major drawback for CNNs at that period and hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.

In deep learning, a **convolutional neural network** (**CNN/ConvNet**) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics **convolution** is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other



Coding

```
from keras.datasets import cifar10
import matplotlib.pyplot as plt
(train_X,train_Y),(test_X,test_Y)=cifar10.load_data()
n=6
plt.figure(figsize=(20,10))
for i in range(n):
    plt.subplot(330+1+i)
    plt.imshow(train_X[i])
plt.show()
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils

train_x=train_X.astype('float32')
test_X=test_X.astype('float32')

train_X=train_X/255.0
test_X=test_X/255.0
```

```
train_Y=np_utils.to_categorical(train_Y)
test_Y=np_utils.to_categorical(test_Y)
num_classes=test_Y.shape[1]

model=Sequential()
model.add(Conv2D(32,(3,3),input_shape=(32,32,3),
   padding='same',activation='relu',
   kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32,(3,3),activation='relu',padding='same',kernel_constraint=m
axnorm(3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(512,activation='relu',kernel_constraint=maxnorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

sgd=SGD(lr=0.01,momentum=0.9,decay=(0.01/25),nesterov=False)
model.compile(loss='categorical_crossentropy',
 optimizer=sgd,
 metrics=['accuracy'])
model.summary()

model.fit(train_X,train_Y,
   validation_data=(test_X,test_Y),
   epochs=25,batch_size=32)

_,acc=model.evaluate(test_X,test_Y)
print(acc*100)
model.save("model1_cifar_10epoch.h5")

results={
  0:'aeroplane',
  1:'automobile',
  2:'bird',
  3:'cat',
  4:'deer',
```

```
  5:'dog',
  6:'frog',
  7:'horse',
  8:'ship',
  9:'truck'
}
from PIL import Image
import numpy as np
im=Image.open("dog.jpeg")
# the input image is required to be in the shape of dataset, i.e (32,32,3)

im=im.resize((32,32))
im=np.expand_dims(im,axis=0)
im=np.array(im)
pred=model.predict_classes([im])[0]
print(pred,results[pred])

import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import numpy

#load the trained model to classify the images

from keras.models import load_model
model = load_model('model1_cifar_10epoch.h5')

#dictionary to label all the CIFAR-10 dataset classes.

classes = {
   0:'aeroplane',
   1:'automobile',
   2:'bird',
   3:'cat',
   4:'deer',
   5:'dog',
```

```
   6:'frog',
   7:'horse',
   8:'ship',
   9:'truck'
}
#initialise GUI

top=tk.Tk()
top.geometry('800x600')
top.title('Image Classification CIFAR10')
top.configure(background='#CDCDCD')
label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = Label(top)

def classify(file_path):
   global label_packed
   image = Image.open(file_path)
   image = image.resize((32,32))
   image = numpy.expand_dims(image, axis=0)
   image = numpy.array(image)
   pred = model.predict_classes([image])[0]
   sign = classes[pred]
   print(sign)
   label.configure(foreground='#011638', text=sign)

def show_classify_button(file_path):
   classify_b=Button(top,text="Classify Image",
  command=lambda: classify(file_path),padx=10,pady=5)
   classify_b.configure(background='#364156', foreground='white',
font=('arial',10,'bold'))
   classify_b.place(relx=0.79,rely=0.46)

def upload_image():
   try:
      file_path=filedialog.askopenfilename()
      uploaded=Image.open(file_path)
      uploaded.thumbnail(((top.winfo_width()/2.25),
```

```
    (top.winfo_height()/2.25)))
        im=ImageTk.PhotoImage(uploaded)
        sign_image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass

upload=Button(top,text="Upload an image",command=upload_image,
  padx=10,pady=5)

upload.configure(background='#364156', foreground='white',
    font=('arial',10,'bold'))

upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True)
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="Image Classification CIFAR10",pady=20,
font=('arial',20,'bold'))

heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

Conclusion

In conclusion, this research is about image classification by using deep learning via framework TensorFlow. It has three objectives that have achieved throughout this research. The objectives are linked directly with conclusions because it can determine whether all objectives are successfully achieved or not. It can be concluded that all results that have been obtained, showed quite impressive outcomes. The deep neural network (DNN) becomes the main agenda for this research, especially in image classification technology. DNN technique was studied in more details starting from assembling, training model and to classify images into categories. The roles of epochs in DNN was able to control accuracy and also prevent any problems such as overfitting. Implementation of deep learning by using framework TensorFlow also gave good results as it is able to simulate, train and classified with up to 75% percent of accuracy towards different types of images that have become a trained model. Lastly, Python have been used as the programming language throughout this research since it comes together with framework TensorFlow which leads to designing of the system involved Python from start until ends.

References

- https://data-flair.training/blogs/image-classification-deep-learning-project-python-keras/
- https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/
- TensorFlow: TensorFlow, www.tensorflow.org/
- Kang, N.: Multi-Layer Neural Networks with Sigmoid Function – Deep Learning for Rookies. https://towardsdatascience.com/multilayer-neural-networks-with-sigmoid-function-deep-learning-forrookies-2-bf464f09eb7f
- https://www.completegate.com/2017022864/blog/deep-machine-learning-images-lenet-alexnet-cnn/all-pages