**ABSTRACT**

Just like the phrase “What-you-see-is-what-you-get” says, human brains make vision easy. It doesn’t take any effort for humans to tell apart a dog, a cat or a flying saucer. But this process is quite hard for a computer to imitate: they only seem easy because God designs our brains incredibly good in recognizing images. Image recognition is a part of Computer Vision (CV).

In this project, I have worked on the image recognition technique using *python* in addition to various inbuilt libraries. This project is aimed to take a defined data-set and train the machine against this data set. Then the trained machine with the help of *machine learning* will be able to recognise the images and categories them into the different categories. (*defined by the CIFAR -10 library*)



The above is the data set categories which is defined for the sake of this project.

**INTRODUCTION**

Definition of Image Recognition:

Image recognition refers to technologies *that identify places, logos, people, objects, buildings, and several other variables in images*. Users are sharing vast amounts of data through apps, social networks, and websites. Additionally, mobile phones equipped with cameras are leading to the creation of limitless digital images and videos. The large volume of digital data is being used by companies to deliver better and smarter services to the people accessing it.

Image Processing:

**Image Processing :**

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it.

It is a type of signal dispensation in which input is an image, like video frame or photograph and output may be image or characteristics associated with that image.

Usually Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.

**Purpose of Image processing**

The purpose of image processing is divided into 5 groups. They are :

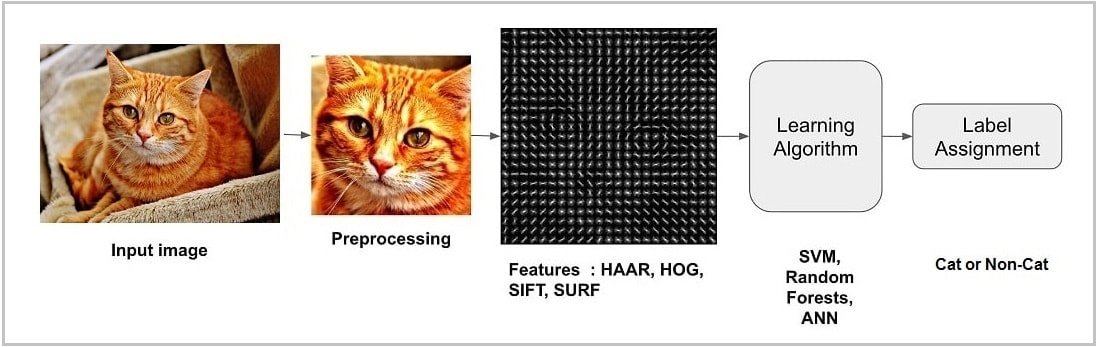
**Visualization** - Observe the objects that are not visible.

**Image sharpening and restoration** - To create a better image.

**Image retrieval** - Seek for the image of interest.

**Measurement of pattern** – Measures various objects in an image.

**Image Recognition** – Distinguish the objects in an image.



The basic layout of the process of image recognition.

**LITERATURE SURVEY AND GAPS:**

Resources :

1. Keras tutorial – build CNN in 11 lines <https://adventuresinmachinelearning.com/keras-tutorial-cnn-11-lines/>
2. Build your first CNN to recognize images

<https://medium.com/intuitive-deep-learning/build-your-first-convolutional-neural-network-to-recognize-images-84b9c78fe0ce>

1. Creating your first Image Recognition using Tensor-flow backend

<https://medium.com/nybles/create-your-first-image-recognition-classifier-using-cnn-keras-and-tensorflow-backend-6eaab98d14dd>

1. Machine learning download practice data sets for training

<https://www.superdatascience.com/pages/machine-learning>

Research Papers :

1. Research of Image Recognition and Classification Based on NIN Model

<https://www.researchgate.net/search.Search.html?type=publication&query=cnn%20for%20image%20recognition>

Focus on the weak feature expression ability of traditional Convolution Neural Network (CNN) in image recognition and classification, the CNN has been improved and optimized by introduction of Mlpconv layer to form a Network in Network (NIN). The experiments of weld image recognition and classification show that the optimized CNN can improve the feature expression ability of the whole network by reducing the number of parameters, obtain higher recognition and classification precision, and avoid the fitting of the network model effectively. The CNN achieve improvement in both performance and efficiency

1. Image Recognition ML

<https://www.researchgate.net/search.Search.html?type=publication&query=cnn%20for%20image%20recognition>

1. Image Rcognition and Image Processing System

<https://www.researchgate.net/search.Search.html?type=publication&query=cnn%20for%20image%20recognition>

1. CNN application in face recognition

<https://www.researchgate.net/publication/339803695_CNN_application_in_face_recognition>

**NOVELTY OF WORK:**

* The project is based on Deep Learning and training the computer to recognize images based on a data set fed to it.
* As a whole, this is a majorly unexplored field, especially at university level.
* The improvement on previous projects/ algorithms is that with the application of machine learning, the image now does not have to belong to the data set itself to be recognized, but once it is recognized, it gets added to the computer’s “memory” and can be used further.
* This algorithm has many applications like :
  + Automatic ”tagging” of photos on facebook
  + In e- commerce and advertising.
  + In automated image organization via cloud.

**ARCHITECTURE/BLOCK DIAGRAM/DESIGN DIAGRAM:**

INSTALL DEPENDENCIES

LOAD DATASET

EXTRACT PIXEL FEATURES FORM dssformFROforIMAGE

PREPARE DATASET TO TRAIN

TRAIN THE MODEL TO CATEGORIZE THE IMAGES

BUILD CNN

INSERT NEW IMAGE

RECOGNIZE THE INSERTED IMAGE THROUGH IMAGE PROCESSING

**MODULES/FUNCTIONLITITES IN DETAIL:**

**MODULES**

**Tensor Flow :**

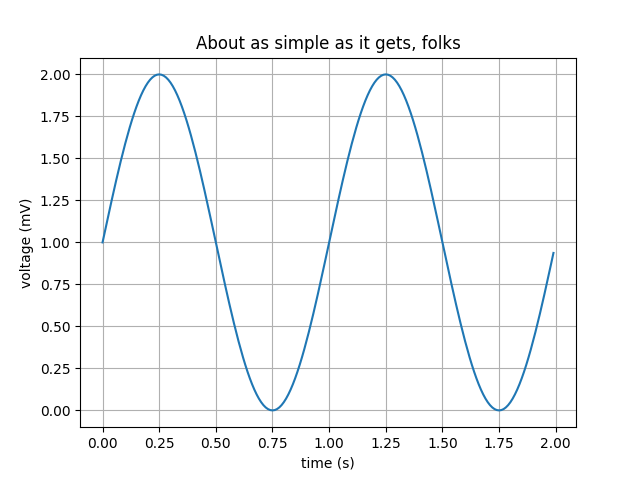
TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

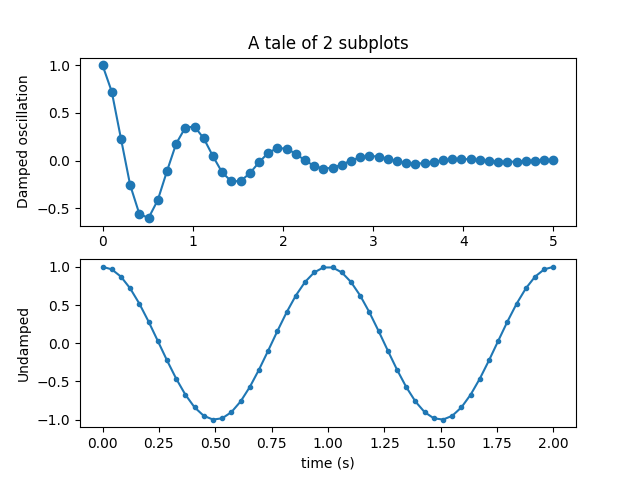
!pip install tensorflow

Code to Install tensor Flow into python.

**MatPlotlib:**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python

[](https://matplotlib.org/gallery/lines_bars_and_markers/simple_plot.html)

[](https://matplotlib.org/gallery/subplots_axes_and_figures/subplot.html)

**KERAS:**

It is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) [neural-network](https://en.wikipedia.org/wiki/Artificial_neural_network) library written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). It is capable of running on top of [TensorFlow](https://en.wikipedia.org/wiki/TensorFlow), [Microsoft Cognitive Toolkit](https://en.wikipedia.org/wiki/Microsoft_Cognitive_Toolkit), [R](https://en.wikipedia.org/wiki/R_(programming_language)), [Theano](https://en.wikipedia.org/wiki/Theano_(software)), or [PlaidML](https://en.wikipedia.org/wiki/PlaidML). Designed to enable fast experimentation with [deep neural networks](https://en.wikipedia.org/wiki/Deep_learning), it focuses on being user-friendly, modular, and extensible.

pip install keras

To install keras Into python

**CONVOLUTIONAL NEURAL NETWORK**

In [deep learning](https://en.wikipedia.org/wiki/Deep_learning), a convolutional neural network (CNN, or ConvNet) is a class of [deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and [translation invariance](https://en.wikipedia.org/wiki/Translation_invariance) characteristics. They have applications in [image and video recognition](https://en.wikipedia.org/wiki/Computer_vision), [recommender systems](https://en.wikipedia.org/wiki/Recommender_system), [image classification](https://en.wikipedia.org/wiki/Image_classification), [medical image analysis](https://en.wikipedia.org/wiki/Medical_image_computing), [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), and financial [time series](https://en.wikipedia.org/wiki/Time_series).

**POOLING**

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average. *Max pooling* uses the maximum value from each of a cluster of neurons at the prior layer. *Average pooling* uses the average value from each of a cluster of neurons at the prior layer.

The types of layers we have used in this project is

1. Max pooling (2D)

**Max pooling** is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned

[](https://www.google.com/url?sa=i&url=https%3A%2F%2Fcomputersciencewiki.org%2Findex.php%2FMax-pooling_%2F_Pooling&psig=AOvVaw1sHwqI0g6wnNfL-i5jnF6b&ust=1585507861541000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCLjC5pjrvegCFQAAAAAdAAAAABAD)

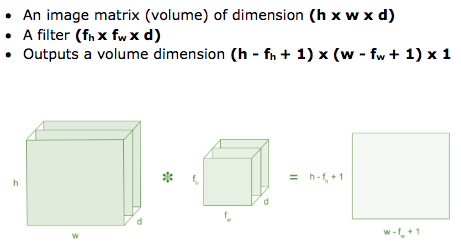
1. Convolutional Layer:

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable [filters](https://en.wikipedia.org/wiki/Filter_(signal_processing)) (or [kernels](https://en.wikipedia.org/wiki/Kernel_(image_processing))), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is [convolved](https://en.wikipedia.org/wiki/Convolution) across the width and height of the input volume, computing the [dot product](https://en.wikipedia.org/wiki/Dot_product) between the entries of the filter and the input and producing a 2-dimensional [activation map](https://en.wikipedia.org/wiki/Activation_function) of that filter. As a result, the network learns filters that activate when it detects some specific type of [feature](https://en.wikipedia.org/wiki/Feature_(machine_learning)) at some spatial position in the input

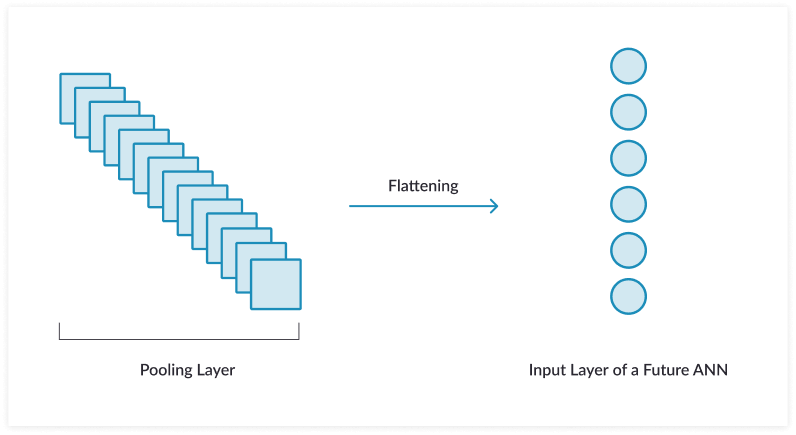
CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d( h = Height, w = Width, d = Dimension



Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.



Flattening layer



A **flatten** operation on a tensor reshapes the tensor to have the shape that is equal to the number of elements contained in tensor non including the batch dimension.

**LIBRARIES**

from keras.datasets import cifar10

**CIFAR 10**

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **airplane** |  |  |  |  |  |  |  |  |  |  |
| **automobile** |  |  |  |  |  |  |  |  |  |  |
| **bird** |  |  |  |  |  |  |  |  |  |  |
| **cat** |  |  |  |  |  |  |  |  |  |  |
| **deer** |  |  |  |  |  |  |  |  |  |  |
| **dog** |  |  |  |  |  |  |  |  |  |  |
| **frog** |  |  |  |  |  |  |  |  |  |  |
| **horse** |  |  |  |  |  |  |  |  |  |  |
| **ship** |  |  |  |  |  |  |  |  |  |  |
| **truck** |  |  |  |  |  |  |  |  |  |  |

* from keras.utils import to\_categorical :

A binary matrix representation of the input. The classes axis is placed last.

* **from** keras.models **import** Sequential

The Sequential model is a linear stack of layers.

* **Rectified Linear Activation Function (reLu)**

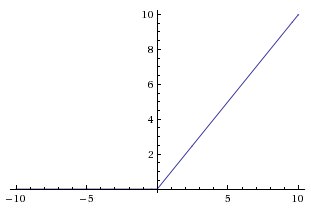
In order to use stochastic gradient descent with backpropogation of errors to train deep neural networks, an activation function is needed that looks and acts like a linear function, but is, in fact, a nonlinear function allowing complex relationships in the data to be learned.

The solution is to use the rectified linear activation function, or ReL for short.A node or unit that implements this activation function is referred to as a **rectified linear activation unit**, or ReLU for short. Often, networks that use the rectifier function for the hidden layers are referred to as rectified networks.

The function returns 0 if it receives any negative input, but for any positive value x it return that valur back.

Mathematically it can be written as *f(x)=max(0,x)*

Graphically as follows



**History**

The History.history attribute is a dictionary recording training loss values and metrics values at successive epochs, as well as validation loss values and validation metrics values (if applicable)

**Sk-image**

scikit-image is a collection of algorithms for image processing

SYNTAX

**from** **skimage** **import** data, io, filters

skimage.transform import resize

Resize image to match a certain size.

Performs interpolation to up-size or down-size N-dimensional images. Note that anti-aliasing should be enabled when down-sizing images to avoid aliasing artifacts. For down-sampling with an integer factor also see [skimage.transform.downscale\_local\_mean](https://scikit-image.org/docs/dev/api/skimage.transform.html#skimage.transform.downscale_local_mean).

**Numpy.argsort()**

Perform an indirect sort along the given axis using the algorithm specified by the kind keyword. It returns an array of indices of the same shape as a that index data along the given axis in sorted orde

**IMPORTS**

* import tensorflow as tf

installed TensorFlow is imported it into your workspace under the alias tf:

* from keras.backend.tensorflow\_backend import set\_session

this specifies the TensorFlow as backend to the program

* from keras.models import Sequential

The sequential model is a linear stack of layers. Inbuilt feature of keras library.

* from keras import applications

Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning.

* from keras.datasets import cifar10

Dataset of 50,000 32x32 color training images, labeled over 10 categories, and 10,000 test images.

* from keras.utils import to\_categorical

You use to\_categorical to transform your training data before you pass it to your model.

If your training data uses classes as numbers, to\_categorical will transform those numbers in proper vectors for using with models. You can't simply train a classification model without that.

* import matplotlib.pyplot as plt

import matplotlib.pyplot as plt gives an unfamiliar reader a hint that [pyplot](https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/pyplot.py) is a module, rather than a function which could be incorrectly assumed from the first form

* from keras.layers import Dense , Flatten , Conv2D , MaxPooling2D

Importing the different layers from Keras library.

* from skimage.transform import resize

Resize image to match a certain size.

* import numpy as np

You can import the entire module with a short name as shown below. This enables you to work with all the functions present in the module.

Adam optimizer

Adam is an adaptive learning rate optimization algorithm that’s been designed specifically for training deep neural networks

**CODE ALONG WITH AN EXPLANATION**

**Explaination**

**Conditions**

**Importing the Library and Datasets:**

**!**pip install keras

!pip install tensorflow

import tensorflow as tf

print(tf.\_\_version\_\_)

from keras.backend.tensorflow\_backend import set\_session

from keras.models import Sequential

from keras import applications

import tensorflow as tf

from keras.datasets import cifar10

* **importing the modules from their respective libraries**
* **\*CIFAR-10 Mentioned above.**
* **#x\_train (the variable that contains the images to train on)**
* **#y\_train(it contains the labels of the images of the training data set)**
* **#x\_test(Variable that contains the images to be tested)**
* **#y test(Variable that contains the lables of the images that are tested)**

(x\_train,y\_train),(x\_test,y\_test)=cifar10.load\_data()

**initializing 4 variables**

**Exploring the data:**

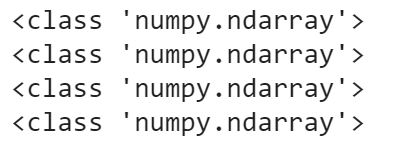
print(type(x\_train))

print(type(y\_train))

print(type(x\_test))

print(type(y\_test))

**output:**



* **In the above snippent we , Print the type of the data that has been imported using CIFAR 10 Library .**

# Get the shapes to know the range of data

print('x\_train shape:',x\_train.shape)

print('y\_train shape:',y\_train.shape)

print('x\_test shape:',x\_test.shape)

print('y\_test shape:',y\_test.shape)

Output

A screenshot of a cell phone

Description automatically generated

* **The shape of x\_train set is a 4-D array with 50,000 rows of 32 x 32 pixel image with depth =3(RGB).**
* **The y\_train data sent is a 2D array with 50,000 rows and 1 coloumn.**
* **The shape of x\_test set is a 4-D array with 50,000 rows of 32 x 32 pixel image with depth =3(RGB).**
* **The y\_test data sent is a 2D array with 50,000 rows and 1 coloumn.**

x\_train[0]

Output : Of Indexed Element =0

array([[[ 59, 62, 63],

[ 43, 46, 45],

[ 50, 48, 43],

...,

[158, 132, 108],

[152, 125, 102],

[148, 124, 103]],

[[ 16, 20, 20],

[ 0, 0, 0],

[ 18, 8, 0],

...,

[123, 88, 55],

[119, 83, 50],

[122, 87, 57]],

[[ 25, 24, 21],

[ 16, 7, 0],

[ 49, 27, 8],

...,

[118, 84, 50],

[120, 84, 50],

[109, 73, 42]],

...,

[[208, 170, 96],

[201, 153, 34],

[198, 161, 26],

...,

[160, 133, 70],

[ 56, 31, 7],

[ 53, 34, 20]],

[[180, 139, 96],

[173, 123, 42],

[186, 144, 30],

...,

[184, 148, 94],

[ 97, 62, 34],

[ 83, 53, 34]],

[[177, 144, 116],

[168, 129, 94],

[179, 142, 87],

...,

[216, 184, 140],

[151, 118, 84],

[123, 92, 72]]], dtype=uint8)

* **Take a look at the first image located at(x=0) in the trainind i.e**
* **x\_train data set as a numpy array. This shows tge series of pixel values.**
* **we get back the array of pixels**
* **value 59 corresponds to Red  62 corresponds to Green 63 corresponds to**

**SHOWING IMAGE IN ARRAY AS A PICTURE**

we have to use another library called **matplotlib**

import matplotlib.pyplot as plt # giving matplotlib an alias =  plt

img=plt.imshow(x\_train[0])  #imshow()is inbuilt feature of matplotlib.

* **Understanding of the image can be improved by marking it to the respective label of the y\_train. So that we get to know which ategory does the image belong to**

**printing the label**

**print('Lable is:', y\_train[0])**

Output:

**\**Label table mentioned above*.**



A screenshot of a cell phone

Description automatically generated

* **As you can see in the first image the label =6 i.e ‘Frog’.**
* **Similarly in the second image the label =1 i.e ‘automobile’.**

**# One-Hot Encoding**:**:**

* **Use One-Hot Encoding to convert the labels into a set of 10 numbers to input into the neural network. The numbers of course corresponds with the number of labels to classify the images.**

from keras.utils import to\_categorical

y\_train\_one\_hot= to\_categorical(y\_train)

y\_test\_one\_hot=to\_categorical(y\_test)

***\*to\_categorical = mentioned above***

print(y\_train\_one\_hot)

print('\*\*\*\*')

print(y\_test\_one\_hot)

* **printing new lables in the training set.**

OUTPUT:

A screenshot of a cell phone

Description automatically generated

**Printing the values of the changed labels**

print('The one hot label is:',y\_train\_one\_hot[0])

**OUTPUT**

The one hot label is: [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]

* **THE ARRAY IS FROM 0-9 CORRESPONDING TO THE SET OF THE LABELS   
   The value 1 corresponds to the postion 6 which is the value of the label initially before converting.**

**NORMALIZE THE PIXELS:**

* **In CNN the images should be value between 0-1, they are normally values between 0 and 255, doing this will help the neural network.**

x\_train=x\_train/255

x\_test=x\_test/255

**Building CNN (Convolutional Neural Network)**

from keras.models import Sequential

from keras.layers import Dense , Flatten , Conv2D , MaxPooling2D

***\*Sequential , Flatten,Conv2D, Maxpooling 2D explained above***

**Creating The architecture**

model = Sequential()

model.add(Conv2D(32,(5,5) , activation='relu' , input\_shape=(32,32,3)))model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Conv2D(32,(5,5) , activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2))

model.add(Flatten())

**Creating our Layers with 1000 Neurons using softmax function**

model.add( Dense(1000, activation='relu')) ***\*relu explained above***

model.add(Dense(10,activation='softmax'))

**COMPILE THE MODEL**

* **Give it the categorical\_crossentropy loss function which is used for classes greater than 2, the adam optimizer, and the accuracy of the model**.

model.compile(loss='categorical\_crossentropy',optimizer ='adam',metrics =['accuracy'])

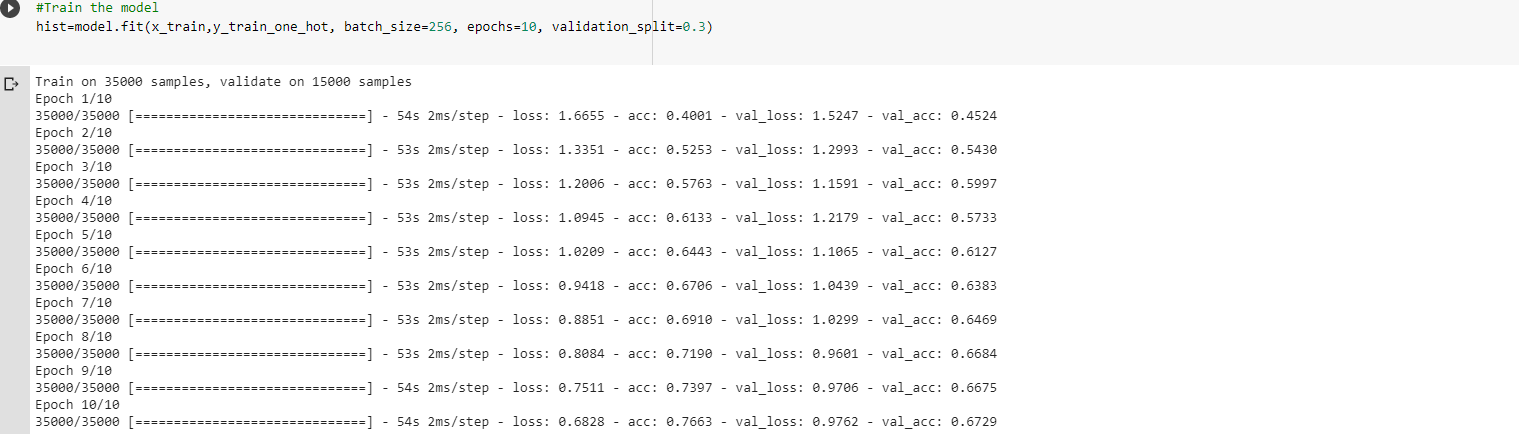
**TRAIN THE MODEL**

* Train the model using the fit() method, which is another word for train. We will train the model on the training data with batch size =256, epochs =10, and split the data into training on 70% of the data and using the other 30% as validation. Training may take some time to finish.
* ***Batch****: Total number of training examples present in a single batch*
* ***Epoch:****The number of iterations when an ENTIRE dataset is passed forward and backward through the neural network only ONCE.*
* ***Fit:*** *Another word for train*

hist=model.fit(x\_train,y\_train\_one\_hot, batch\_size=256, epochs=10, validation\_split=0.3)

#hist=history ***\*explained above***

OUTPUT:



**Get The Models Metrics**

**PRINTING THE MODEL ACCURACY**

print('Model accuracy--')

model.evaluate(x\_test,y\_test\_one\_hot)[1]

**Output:**

Model accuracy--

10000/10000 [==============================] - 5s 475us/step

0.6765

**Visualize the models accuracy**

**using matplotlib** \**explained above*

plt.plot(hist.history['acc'])

plt.plot(hist.history['val\_acc'])

plt.title('Model Accuracy')

plt.ylabel("Image Accuracy")

plt.xlabel("Epoch")

plt.legend(['Train','Val'], loc='upper left')

plt.show()

OUTPUT

A close up of a map

Description automatically generated

**Visualize the models loss using matplotlib**

plt.plot(hist.history['loss'])

plt.plot(hist.history['val\_loss'])

plt.title('Model Loss')

plt.ylabel("Image Loss")

plt.xlabel("Epoch")

plt.legend(['Train','Val'], loc='upper right')

plt.show()

OUTPUT:

A screenshot of a cell phone

Description automatically generated**Test The Model**

* **Load the data that you want to classify from an image file into the variable my\_image.**

**IMPORTING & SHOWING THE IMAGES FROM THE COMPUTER**

A screenshot of a computer screen

Description automatically generated

**Show the uploaded image.**

img = plt.imshow(my\_image)

**Resize the image**

* **Resize the image using sk-image library**

from skimage.transform import resize

*\*skimage transform explained above*

my\_image\_resized=resize(my\_image,(32,32,3))

img=plt.imshow(my\_image\_resized)

**Getting Probability of the function**

import numpy as np

probabilities = model.predict(np.array([my\_image\_resized,]))

* **placing the image selected in the array, so as to map it according t**

**Printing the top 3 probabilities**

probabilities

OUTPUT:

array([[1.3949684e-02, 1.9906268e-07, 9.0245694e-02, 8.2558841e-02,

4.3213114e-04, 8.0310994e-01, 3.4364886e-03, 5.6749099e-04,

5.4357694e-03, 2.6374127e-04]], dtype=float32)

* **In the above snippet, the highest value corresponds to the probability of the class to which the image belongs**
* **Second highest value corresponds to the second most probability to which the image belongs**
* **And so on…**
* **Add the label of the classes to an array in the index of the labels corresponding equivalent number. For example ‘airplane’ will be located at index=0 , because the corresponding label as a number is 0. So as to indentify the class of the image to which it belongs**

number\_to\_class=['airplane','automobile','bird','cat','Deer','Dog','Frog','Horse','Ship','Truck']

index=np.argsort(probabilities[0,:]) #sorting in ascending order

print('Most likely class:',number\_to\_class[index[9]],'--probability',probabilities[0,index[9]]\*100 )

print('Second Most likely class:',number\_to\_class[index[8]],'--probability',probabilities[0,index[8]]\*100 )

print('Third Most likely class:',number\_to\_class[index[7]],'--probability',probabilities[0,index[7]]\*100 )

*\*argsort function explained above*

**// END OF CODE**

**SOFTWARE REQUIREMENT**

Here is the list of the software and the libraries used in this project:

* PYTHON programming language (software used GOOGLE COLAB)
* In python the libraries used are
  + Tensor Flow
  + Matplotlib( to get the model metrics)
  + Keras Library
  + Concepts of Convolution Neural Network(CNN)
    - Creating architecture with 1000 neurons and function ReLu.
    - Creating a basic layer
    - Creating pooling layer
    - Creating max-pooling layer
    - Flattening the layer
  + Concepts of Probabilities (to sort the generated model )
* Knowledge of Python Programming
* Keras Datasets for machine learning

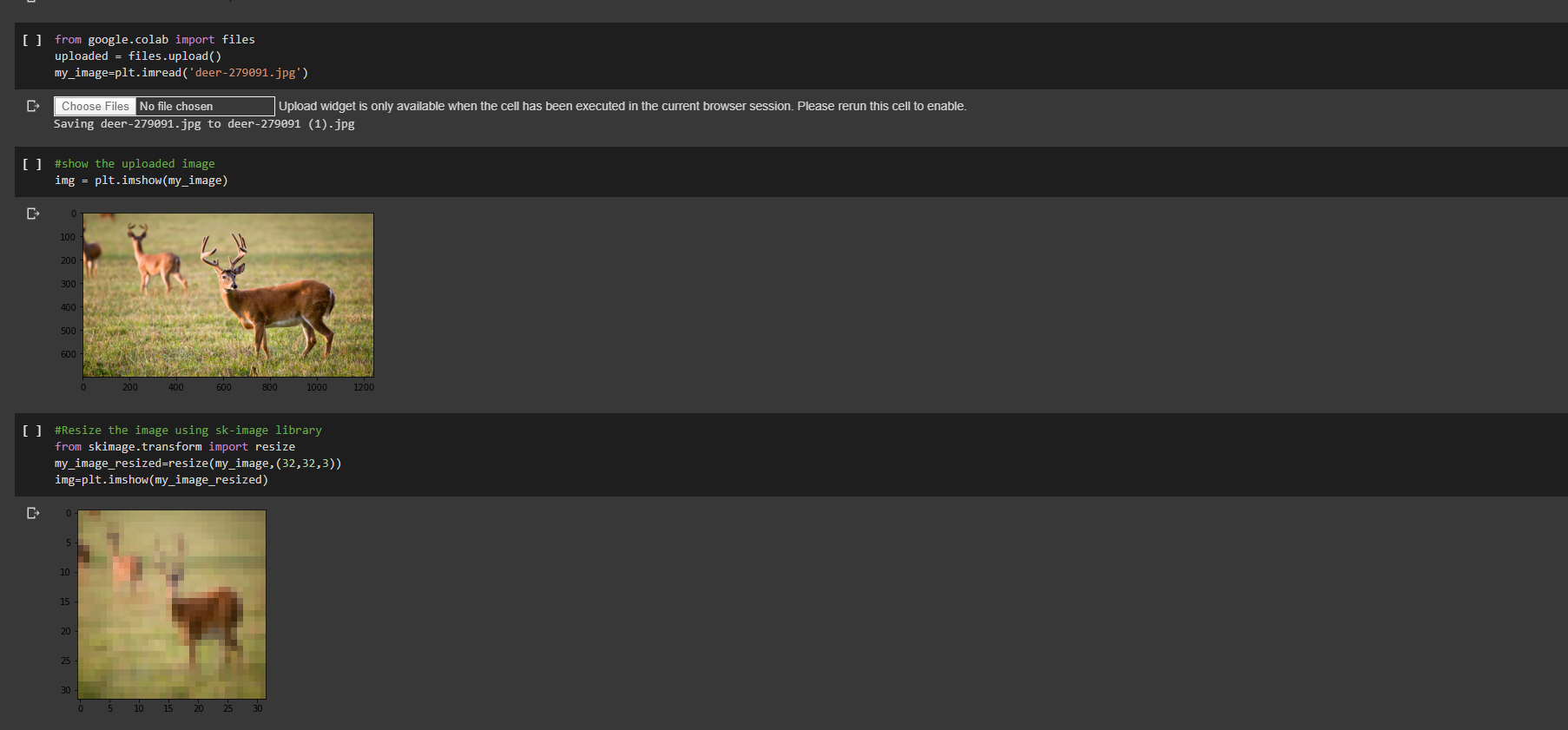
**HARDWARE REQUIREMENT**

A system with python installed.

**RESULTS/SCREENSHOTS**

**INSERTED IMAGE:**

The image inserted is a Deer



A screenshot of a computer screen

Description automatically generated

**FINAL PROBABILITY USING IMAGE PROCESSING:**

A screenshot of a cell phone

Description automatically generated

As you can see, the highest probability is Deer.

**END**