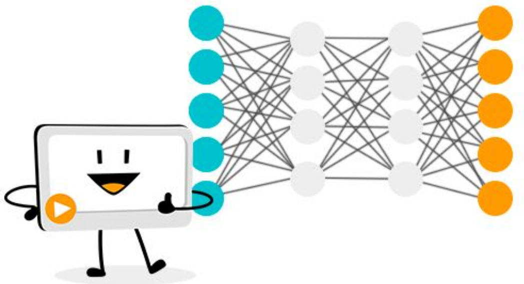
रा.इ.सू.प्रौ.सं
NIELIT
Calicut



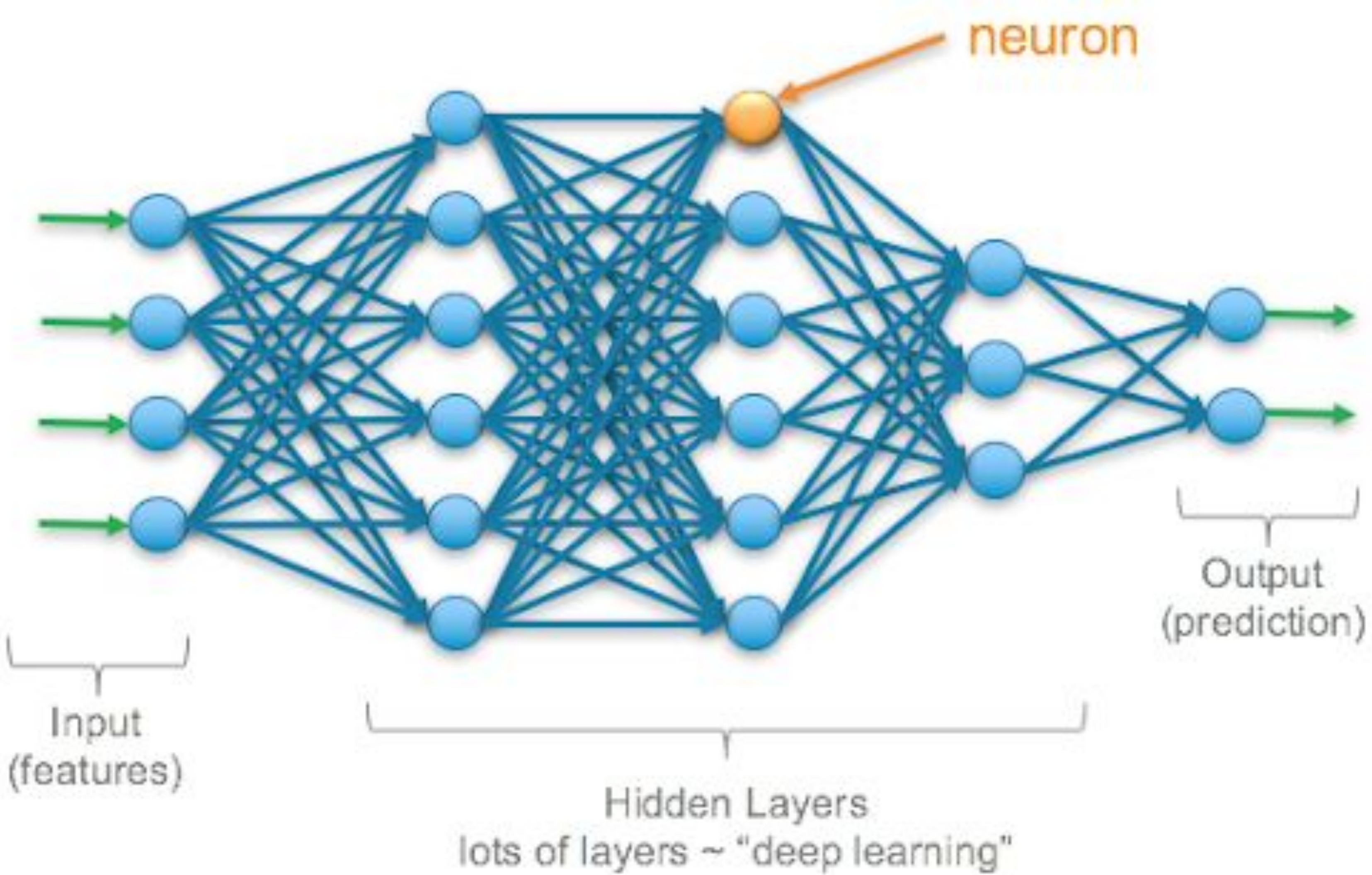
DL-7

Lessons in Deep Learning

This presentation contains 4 projects on Applications of Deep learning for various domains. We used Keras and other python modules to execute the programs. This whole project is a product of Learnings and guidance under -

SMT. VIMALA MATHEW

(SCIENTIST, NIELIT,CALICUT)


neuron
Input
(features)
Hidden Layers
lots of layers ~ "deep learning"
Output
(prediction)

Project 1: Image Captioning with Keras and Tensorflow

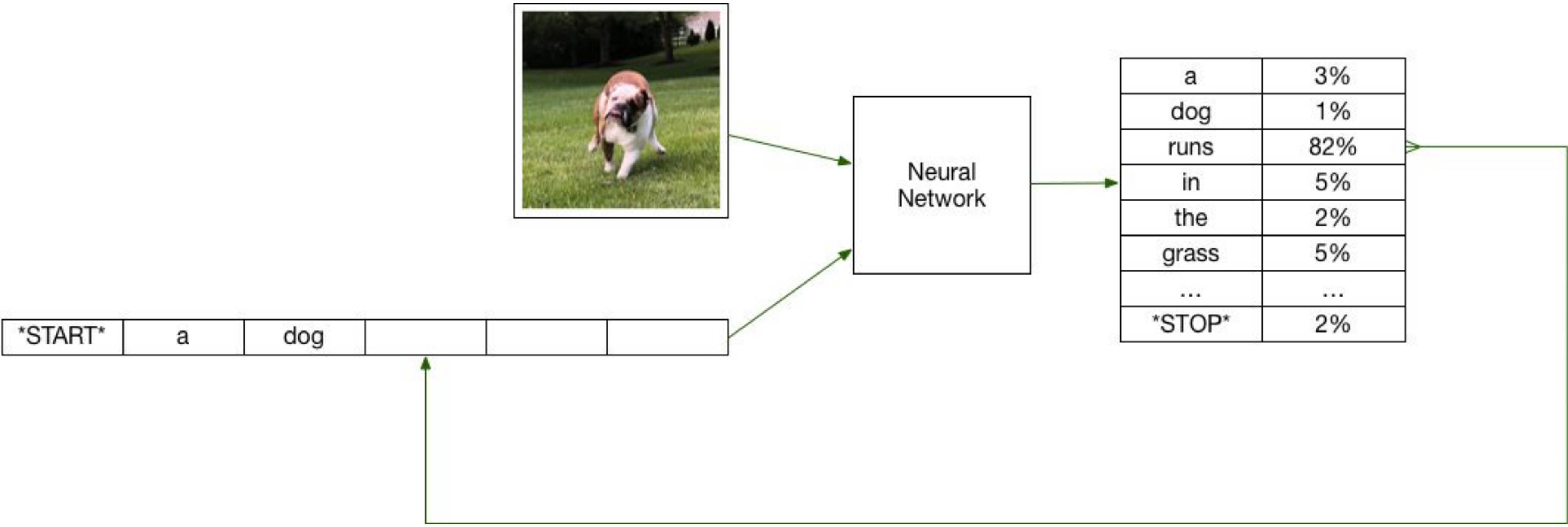
Using multi-image recognition and natural language processing it is possible to create a neural network that can write captions for images. Here, we show how to create and train an image captioning neural network for Keras. Transfer learning is used to greatly reduce training time. Use of glove and InceptionV3 have been made for the integration.

Data used

We will need to download the following data and place it in a folder for this Project.Lets point the root_captioning string at the folder that we are using for the caption generation. This folder should have the following sub-folders.

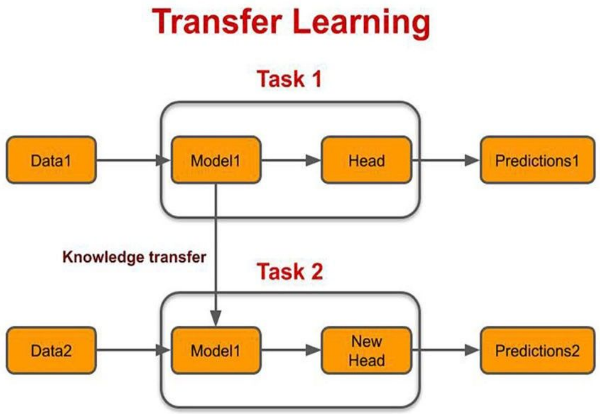
- data - Create this directory to hold saved models.
- glove.6B - Glove embeddings.
- Flicker8k_Dataset - Flicker dataset.
- Flicker8k_Text

Structure of the Project



Choosing a Computer Vision Neural Network to Transfer

We used InceptionV3(2048 features) to extract features from the image. Though, use of MobileNet could help getting more detailed features, the processing time would have been a limitation. One characteristic that we are seeking for the image neural network is that it does not have too many outputs (once you strip the 1000-class imagenet classifier, as is common in transfer learning).



The Project overall uses two Neural Networks that w can use via Transfer Learning. Besides InceptionV3 for images, We use Glove for text Embedding. Both of these transfers serve to extract features from the raw text and the images. Without this prior knowledge transferred in, this example would take considerably more training.

Using a Data Generator

Up to this point, we've always generated training data ahead of time and fit the neural network to it. It is not always practical to create all of the training data ahead of time. The memory demands can be considerable. If we generate the training data as the neural network needs it, it is possible to use a Keras generator. The generator will create new data as it is needed. The generator provided here creates the training data for the caption neural network, as it is needed.

X1	X2							Y
	START	a	dog	runs	in	the	grass	*STOP*
	*START*	a	dog	runs	in	the		grass
	START	a	dog	runs	in			the
	START	a	dog	runs				in
	START	a	dog					runs
	START	a						dog
	START							a
	START	a	dog	wears	a	coat		*STOP*
	*START*	a	dog	wears	a			coat
	START	a	dog	wears				a
	START	a	dog					wears
	START	a	dog					dog
	START	a						a

How Caption Training Data works ?

If we generate the training data as the neural network needs it, it is possible to use a Keras generator. The generator will create new data as it is needed. The generator provided here creates the training data for the caption neural network, as it is needed.

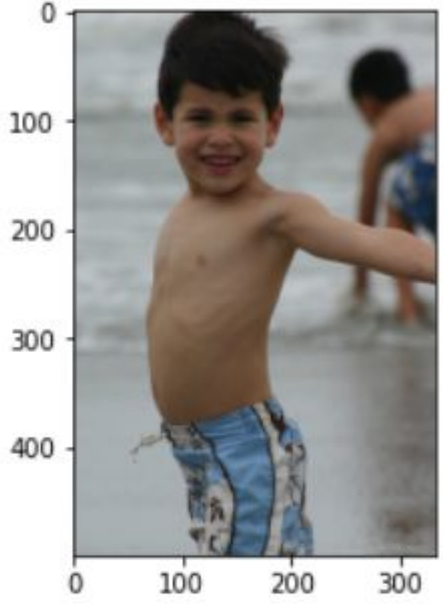
X1

X2							Y
START	a	dog	runs	in	the	grass	*STOP*
START	a	dog	runs	in	the		grass
START	a	dog	runs	in			the
START	a	dog	runs				in
START	a	dog					runs
START	a						dog
START							a

X2							Y
START	a	dog	wears	a	coat		*STOP*
START	a	dog	wears	a			coat
START	a	dog	wears				a
START	a	dog					wears
START	a	dog					dog
START	a						a

Output

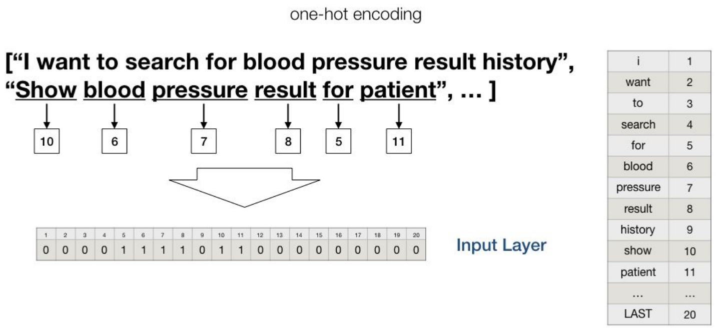
The trained model studies the image and provides appropriate caption for the picture.



Caption: boy in blue shorts and blue shorts is splashing in puddle

Requirements

1. Various Keras dependencies
2. Glove.6B (text embedding)
3. Pickle (to store the trained data into byte stream)
4. Other regular modules.



References-

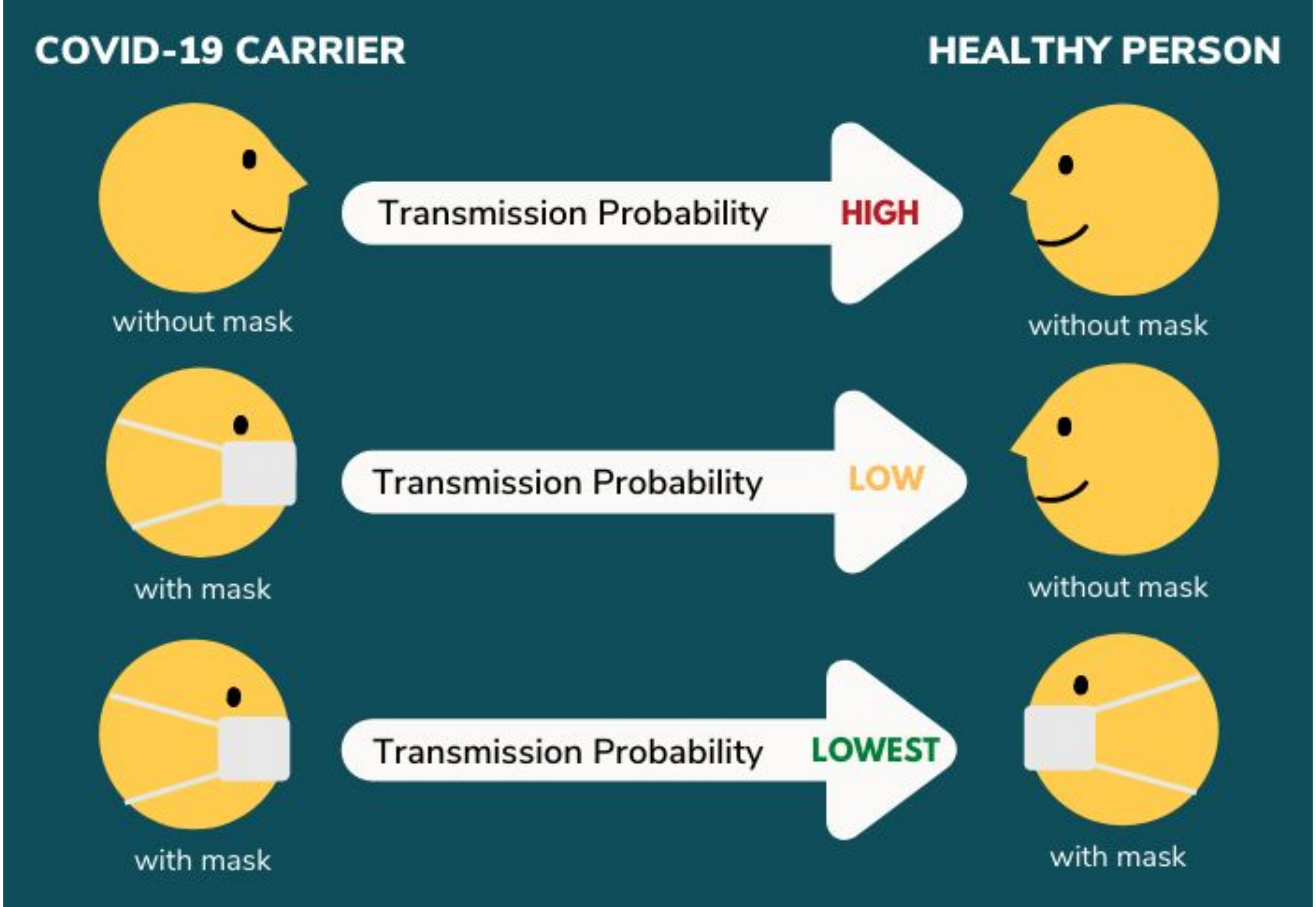
1. Vimala Mam's DL Lectures
2. Washington University(for transfer learning)
3. Stanford University (for Glove)


Surprised
Happy

Project 2 : Computer Vision

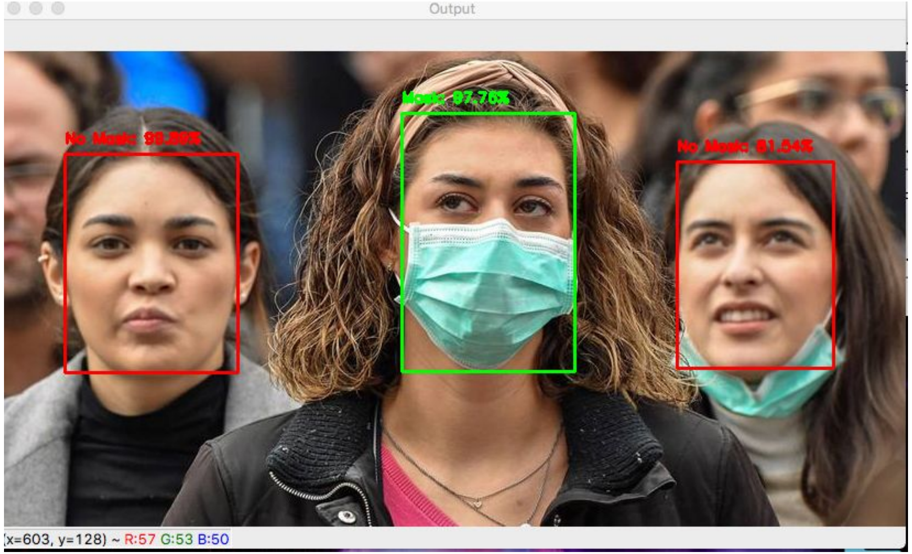
COVID-19: Face Mask Detector

Our goal is to train a custom deep learning model to detect whether a person is or is not wearing a mask using Computer Vision, running on a webcam

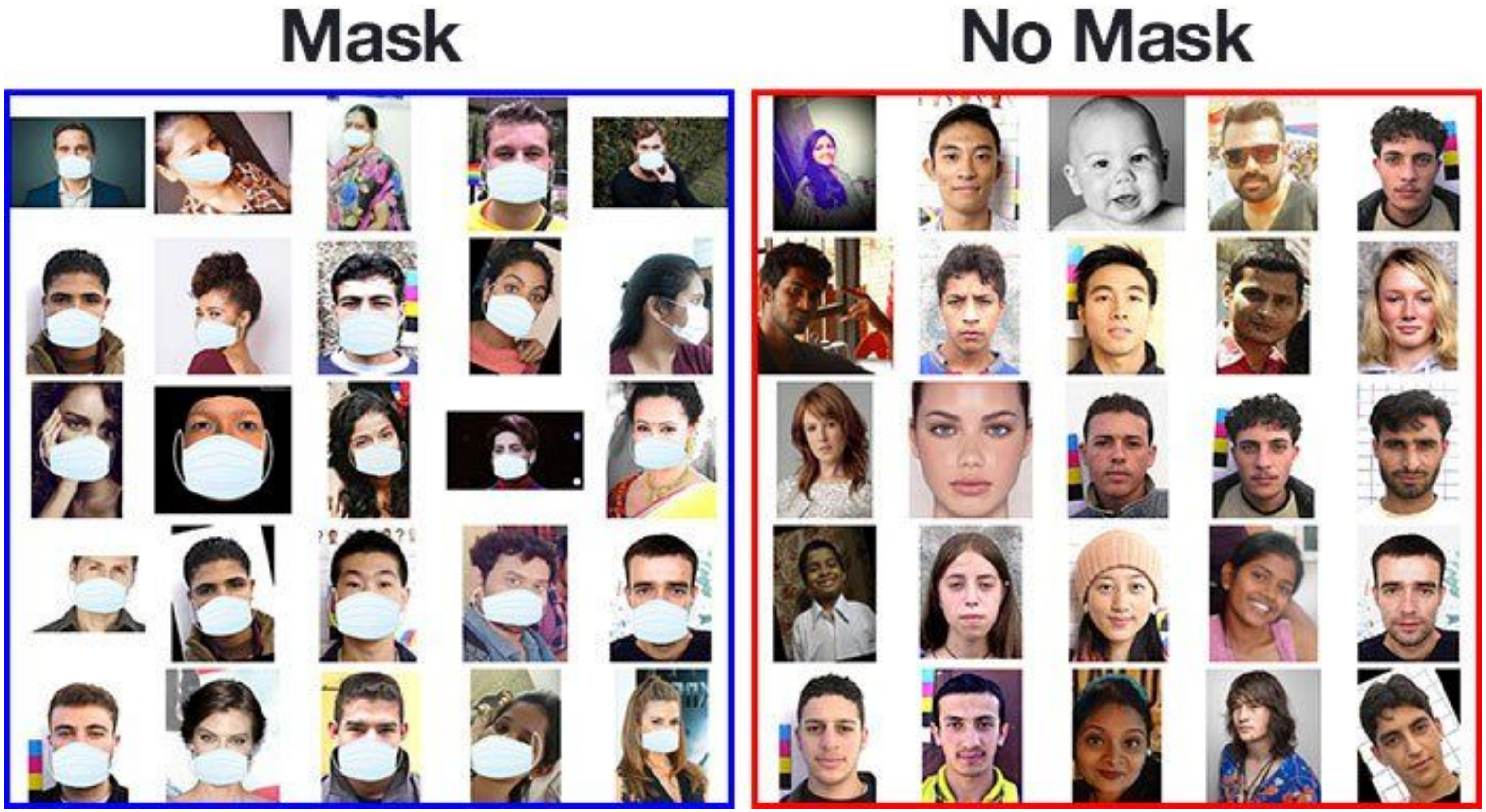

COVID-19 CARRIER
HEALTHY PERSON
Transmission Probability
HIGH
without mask
without mask
Transmission Probability
LOW
with mask
without mask
Transmission Probability
LOWEST
with mask
with mask

What's the Need?

In the new world of coronavirus, multidisciplinary efforts have been organized to slow the spread of the pandemic. The AI community has also been a part of these endeavors. In particular, developments for monitoring social distancing or identifying face masks have **made-the-headlines**.

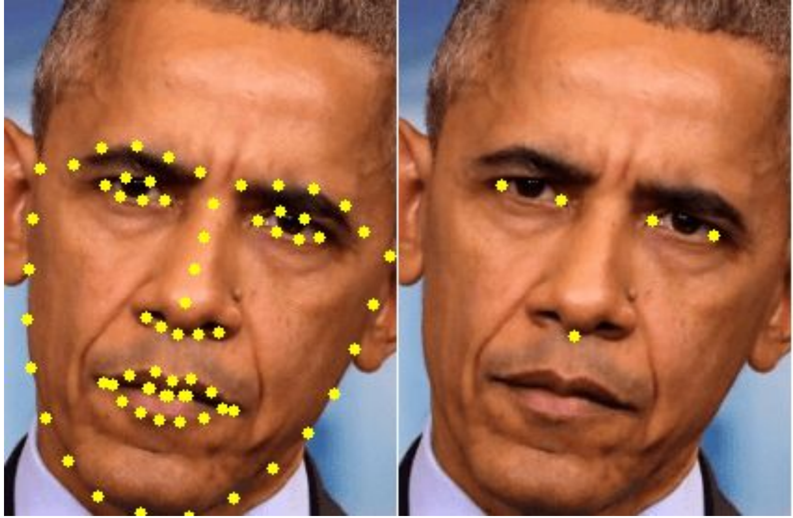


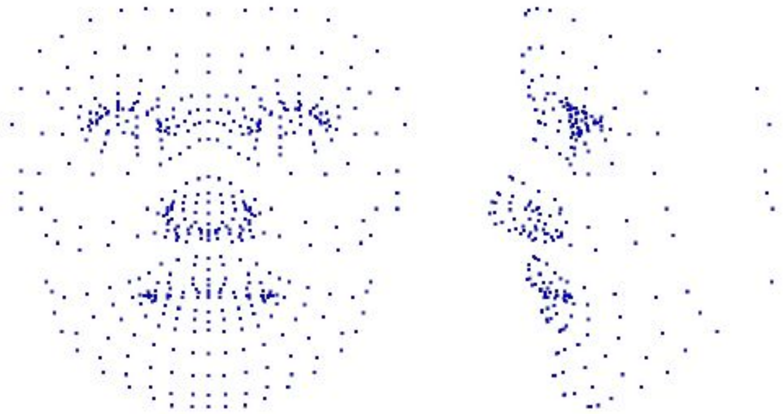
How the data will work?



Facial Landmarks

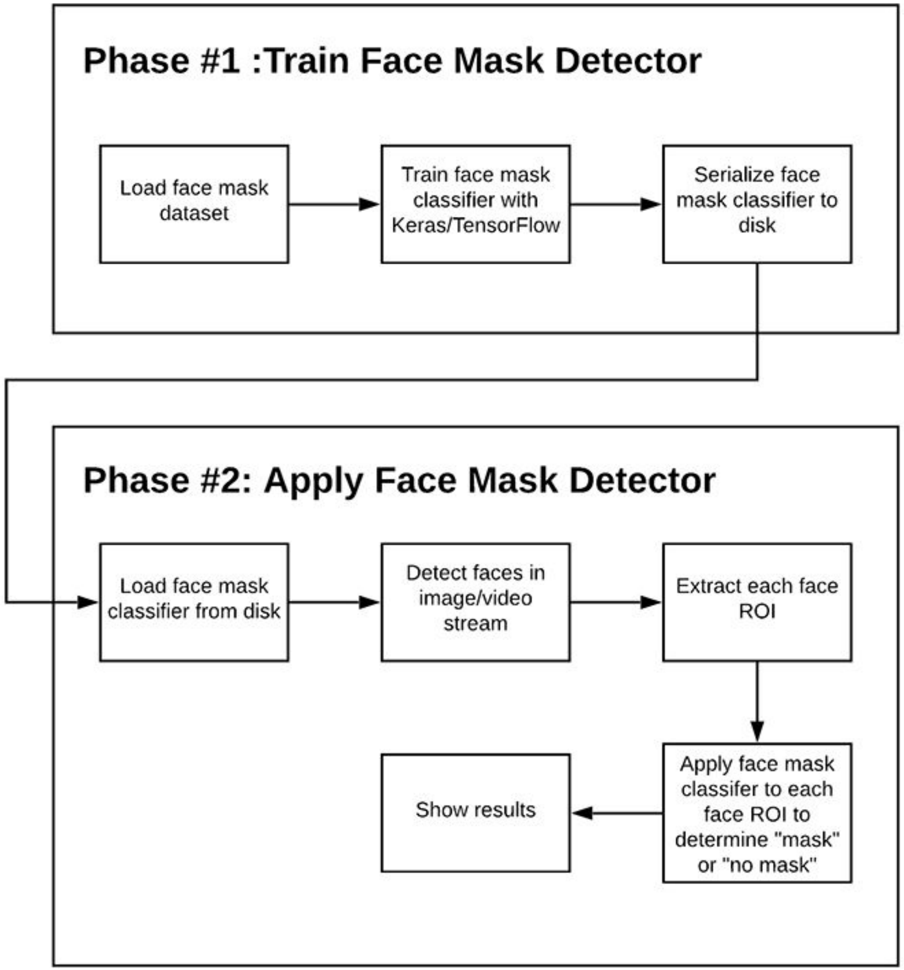
- Facial landmarks allow us to automatically infer the location of facial structures, including:
  - Eyes
  - Eyebrows
  - Nose
  - Mouth
  - Jawline
- Therefore, training the dataset with masks makes it easier to categorize





Steps

1. Training the Face mask Detector
2. Applying Face Mask Detector
3. Load the input image from disc
4. Detect Faces in the image
5. Check the results
6. Retrain if more accuracy needed.
7. Deploy when ready



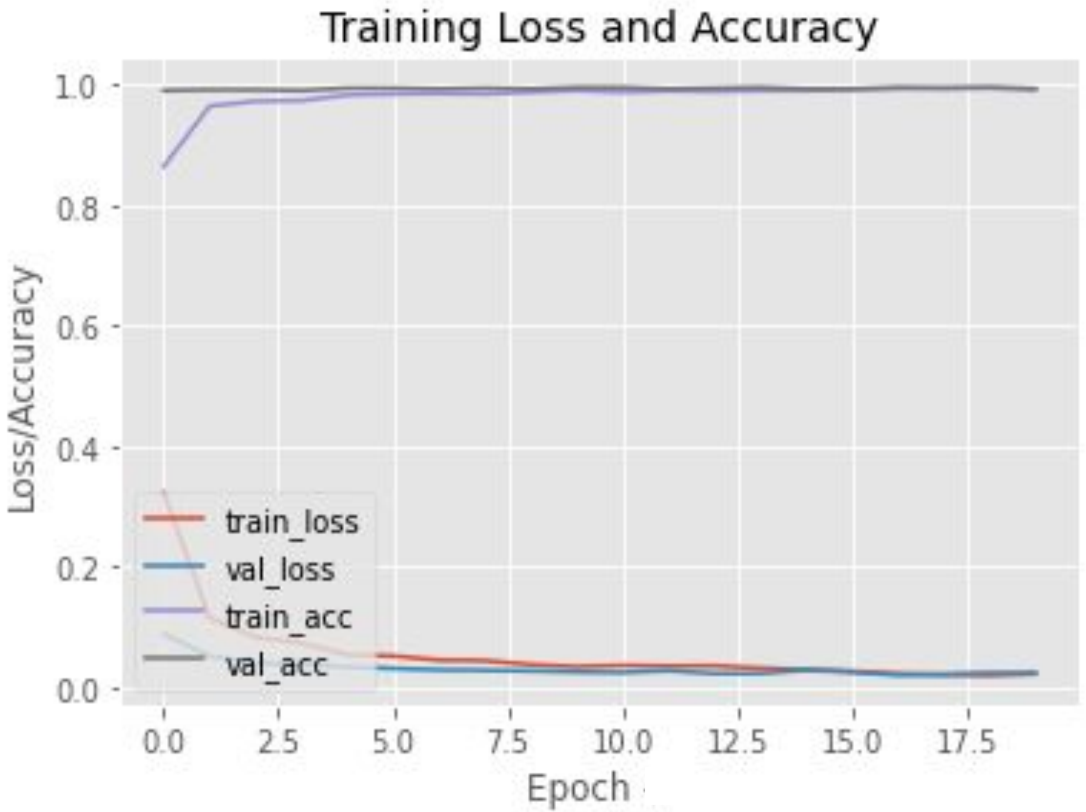
Fitting the model

1. load the MobileNetV2 network, ensuring the head FC layer sets are left off
2. construct the head of the model that will be placed on top of the the base model
3. loop over all layers in the base model and freeze them so they will
4. Fit the model

```
base = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))

head = base.output
head = AveragePooling2D(pool_size=(7, 7))(head)
head = Flatten(name="flatten")(head)
head = Dense(128, activation="relu")(head)
head = Dropout(0.5)(head)
head = Dense(2, activation="softmax")(head)

model = Model(inputs=base.input, outputs=head)

for layer in base.layers:
    layer.trainable = False

model.compile(loss="binary_crossentropy",
    optimizer=Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS),
    metrics=["accuracy"])

History=model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)
```

Training Loss and Accuracy

1. Training loss -0.99
2. Training accuracy-0.99



Creating the Function

To detect and predict mask images, we take frame, faceNet and maskNet as parameters.

Using Computer Vision and setting Facial Landmarks, our function will differentiate the categories according to a given threshold.

```
def detect_and_predict_mask(frame, faceNet, maskNet):
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
        (104.0, 177.0, 123.0))

    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    faces = []
    locs = []
    preds = []

    for i in range(0, detections.shape[2]):

        confidence = detections[0, 0, i, 2]

        if confidence > 0.5:
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face)
            face = preprocess_input(face)

            faces.append(face)
            locs.append((startX, startY, endX, endY))

    if len(faces) > 0:
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)

    return (locs, preds)
```

Real time Face Detection

- Load the model and Initialize the video stream
- Loop over the frames from the video stream
- Detects faces in the frame and determine if they are wearing a face mask or not
- Determine the class label and color
- Then, with the detected image, we also check the probability of being accurate.

```
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
maskNet = load_model("mask_detector.model")

vs = VideoStream(src=0).start()

while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=400)
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

    for (box, pred) in zip(locs, preds):
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

        cv2.putText(frame, label, (startX, startY - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("A"):
        break
```

Output

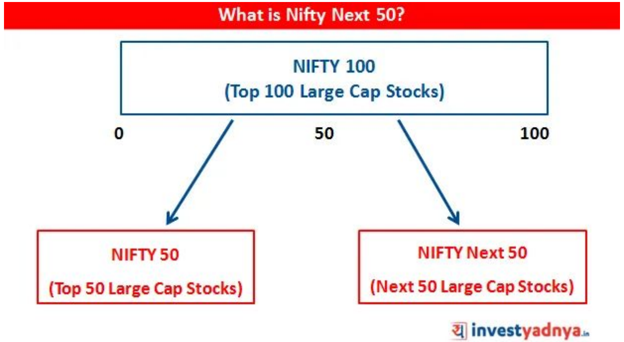

PREDICT STOCK MARKET PRICES USING PYTHON
PYTHON FOR FINANCE

Project 3 : Stock Market Predictions for NIFTY

Getting the Data

The Data has been Scraped using Beautiful Soup and wikipedia.Wikipedia helped get the stocks for NIFTY50

The Quandl API has been used to get the useful parameters for the stocks.



Wikipedia

The Free Encyclopedia

How it works?

```
In [4]: import quandl
        import os
        import pandas as pd
        import pickle
        import bs4 as bs
        import requests

        quandl.ApiConfig.api_key='qsA-s6MZZ6iAdJNN4zd'

        startdate="2012-11-22"
        enddate="2020-10-25"

        def nifty_50_list():
            resp = requests.get('https://en.wikipedia.org/wiki/NIFTY_50')
            soup = bs.BeautifulSoup(resp.text, 'lxml')

            table = soup.find('table', {'class': 'wikitable sortable'},'tbody')

            tickers = []
            for row in table.findAll('tr')[1:]:
                ticker = row.findAll('td')[1].text
                # print(f"ticker{ticker}")
                tickers.append(ticker)
            l=[]
            print(tickers)
            for i in tickers:
                split_string = i.split(".", 1)
                l.append(split_string[0])
            print(l)
```

```
    with open("nifty50_list.pickle","wb") as f:
        pickle.dump(l,f)

    return l

#function to scrap NIFTY50 list from WIKI only if not already obtained
def get_nifty50_list(scrap=False):
    if scrap:
        tickers=nifty_50_list()
    else:
        with open("nifty50_list.pickle","rb") as f:
            tickers=pickle.load(f)
    return tickers


#function to fetch stock prices from Quandl and then storing them to avoid mak
def getStockdataFromQuandl(ticker):
    quandl_code="NSE/"+ticker
    try:
        if not os.path.exists(f'stock_data/{ticker}.csv'):
          data=quandl.get(quandl_code,start_date=startdate,end_date=enddate)
          data.to_csv(f'stock_data/{ticker}.csv')
        else:
            print(f"stock data for {ticker} already exists")
    except quandl.errors.quandl_error.NotFoundError as e:
        print(ticker)
        print(str(e))
```

Preparing the Neural Network?

After preparing the dataset, we need to build a neural Network model to train it. Using this model helps predict the stocks for next 7 days with better accuracy.

We use Sequential model from Keras. Add layers with LSTM,while utilizing Batch Normalizing and use ‘reLu’ and ‘softmax’ functions to develop the model.

```
NAME="NIFTY50PRED"
BATCH_SIZE=64
EPOCHS=100


def build_model():

    model=Sequential()
    model.add(LSTM(256,input_shape=(train_x.shape[1:]),return_sequences=True))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())

    model.add(LSTM(256,return_sequences=True))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())

    model.add(LSTM(256,return_sequences=False))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())

    model.add(LSTM(128,return_sequences=False))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())

    model.add(Dense(32,activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(2,activation='softmax'))
```

Accuracy and error score

Validation accuracy -75

Loss function -77

```
Epoch 99/100
3/3 [==============================] - 1s 274ms/step - loss: 0.1291 - accurac
y: 0.9493 - val_loss: 0.7861 - val_accuracy: 0.7500
Epoch 100/100
3/3 [==============================] - 1s 275ms/step - loss: 0.1641 - accurac
y: 0.9275 - val_loss: 0.7713 - val_accuracy: 0.7500
1/1 [==============================] - 0s 2ms/step - loss: 0.7713 - accuracy:
0.7500
Validation accuracy percentage 75.0
Validation loss percentage 77.12905406951904
```

*A **loss** function is used to optimize a machine learning algorithm. The **loss** is calculated on training and **validation** and its interpretation is based on how well the model is doing in these two sets. ... An **accuracy** metric is used to measure the algorithm's performance in an interpretable way.

What next?

Once our Neural Network is trained, we can save our model as an h5 file and use it for deployment.

This step is in the Process.

Project 4 : Deployment of Deep learning Model

A Deep learning model is usually saved as an h5 file.

We can use flask,Django, cloud platforms and other ways to deploy our models.

Here, we provide a simple method to deploy.

Flask

web development,
one drop at a time

Neural network deployment is a complex process, usually carried out by a company's Information Technology (IT) group. When large numbers of clients must access your model, scalability becomes essential. The cloud usually handles this. The designers of Flask did not design for high-volume systems. When deployed to production, you will usually wrap models in Gunicorn or TensorFlow Serving. However, Flask is directly compatible with the higher volume Gunicorn system. It is common to use a development system, such as Flask, when developing your initial system.

```
Skipping registering GPU devices...
2020-10-31 21:48:58.581190: I tensorflow/core/platform/cpu_feature_guard.cc:142] This
h oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in
AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler
2020-10-31 21:48:58.611662: I tensorflow/compiler/xla/service/service.cc:168] XLA ser
platform Host (this does not guarantee that XLA will be used). Devices:
2020-10-31 21:48:58.619401: I tensorflow/compiler/xla/service/service.cc:176]   Strea
 Version
2020-10-31 21:48:58.626100: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1257]
with strength 1 edge matrix:
2020-10-31 21:48:58.634895: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1263]
 * Debugger is active!
 * Debugger PIN: 224-614-066
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [31/Oct/2020 21:49:04] "[33mPOST /api/mask HTTP/1.1[0m" 404 -
***2:dog.jpg
127.0.0.1 - - [31/Oct/2020 21:49:22] "[37mPOST /api/image HTTP/1.1[0m" 200 -
```

Server creation

We can also accept images from web services. We will create a web service that accepts images and classifies them using MobileNet. Usually,We will use the Neural Network and load it as .h5 file in image_server.py and run the server in POSTman.But, keras provide MobileNet as a pretrained Model. Using that will save our computing.

POSTMAN

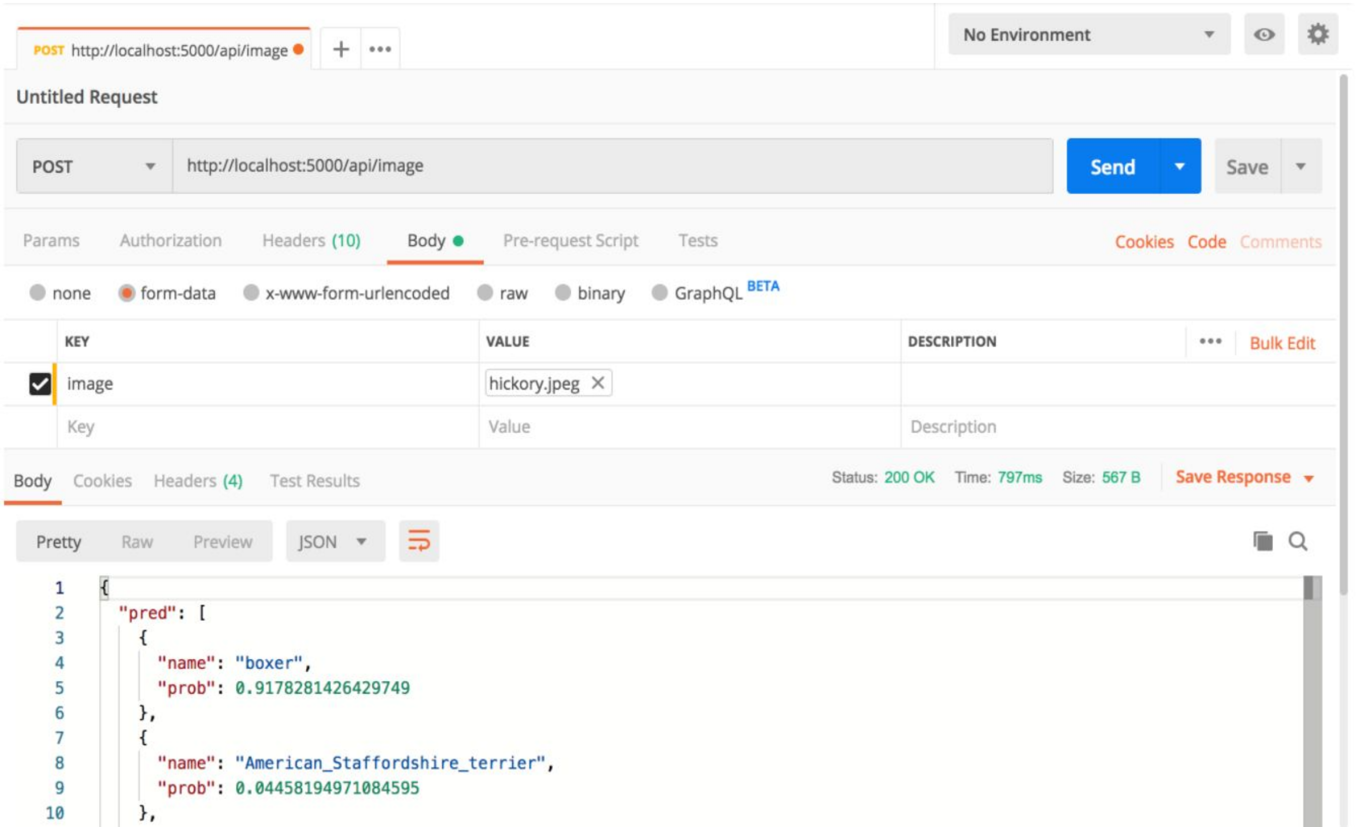
```python
from tensorflow.keras.models import load_model
import numpy as np
import os
from flask import Flask, request, redirect, url_for
from werkzeug.utils import secure_filename
from tensorflow.keras.applications import MobileNet
from PIL import Image, ImageFile
from io import BytesIO
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.mobilenet import preprocess_input
from tensorflow.keras.applications.mobilenet import decode_predictions

IMAGE_WIDTH = 224
IMAGE_HEIGHT = 224
IMAGE_CHANNELS = 3


def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

model = MobileNet(weights='imagenet',include_top=True)

@app.route('/api/mask', methods=['POST'])
def upload_image():
  # check if the post request has the file part
  if 'image' not in request.files:
      return jsonify({'error':'No posted image. Should be attribute named image.'})
  file = request.files['image']

  if file.filename == '':
      return jsonify({'error':'Empty filename submitted.'})
  if file and allowed_file(file.filename):
      filename = secure_filename(file.filename)
      print("***2:"+filename)
      #file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
      x = []
      ImageFile.LOAD_TRUNCATED_IMAGES = False
      img = Image.open(BytesIO(file.read()))
      img.load()
      img = img.resize((IMAGE_WIDTH,IMAGE_HEIGHT),Image.ANTIALIAS)
      x = image.img_to_array(img)
      x = np.expand_dims(x, axis=0)
      x = preprocess_input(x)
```

Input File

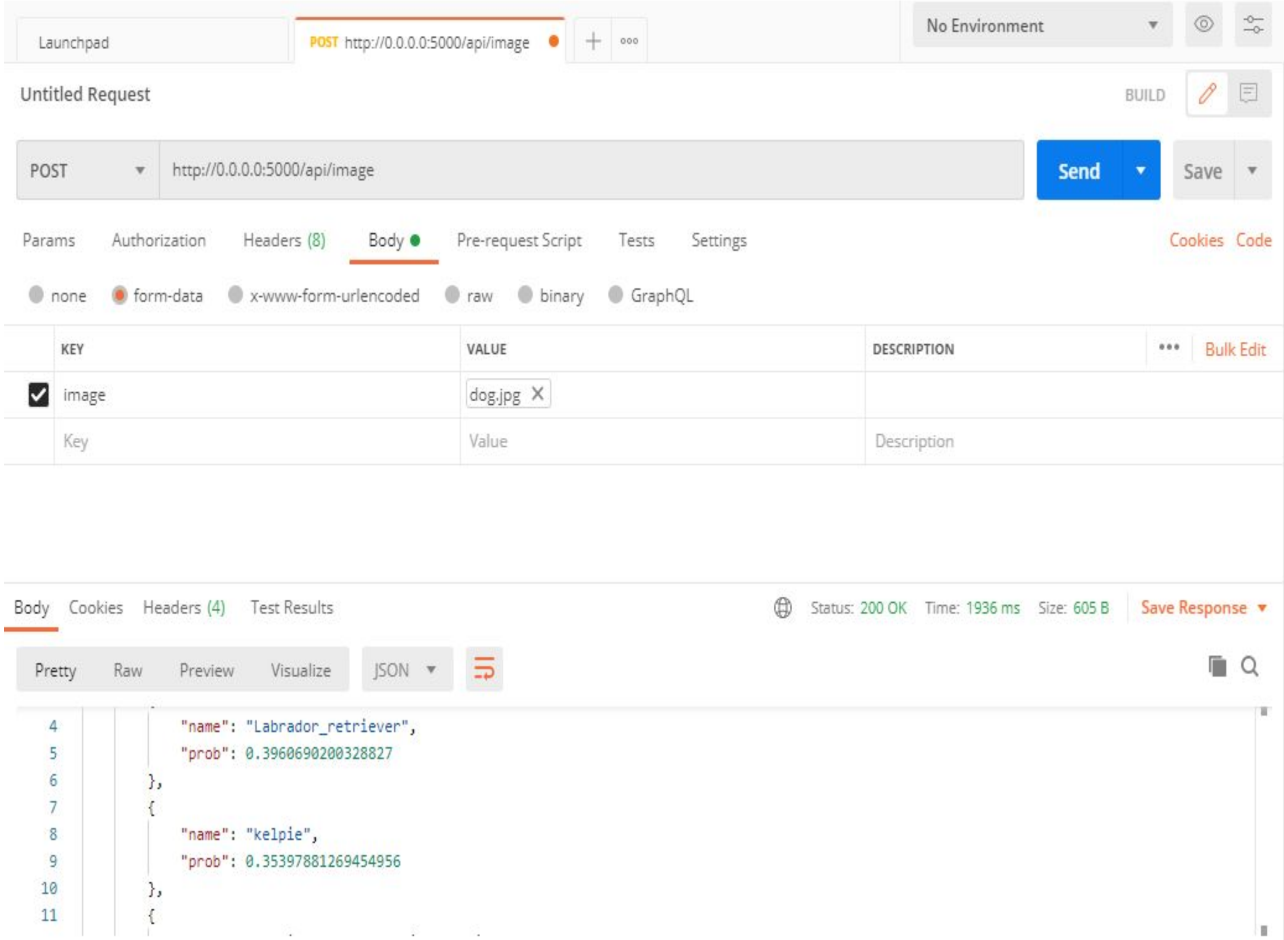
Taking a dog's Image as hickory.py, MobileNet will provide us with its features and PostMan will help us check it online using the server.

Output File

The Output file here, provides the description according to the MobileNet in this case and provides the features.



Another example



TEAM

Shravan Kumar

Rohit Gupta

Tojo Benny

Ramiz Fazal

Rakesh Ranjan

Satish Unnikrishnan

References

1. For Idea - Blogs on Towards Data Science
2. Vimala Mam's Lectures
3. Washington University
4. Youtube
5. Keras website
6. Google.com (for images)
7. POSTman

Python Libraries Used-
1. Tensorflow
2. Keras
3.Sklearn
4.Flask
5.cv2
6. Other regular modules

For Code and Datasets, Please visit-

https://github.com/rohitgupta29/NIELIT_DL3

Thank You