CURRENCY DENOMINATION PREDICTION IN REAL-TIME

USING TENSORFLOW & OPENCV

Aryan Khatana
BLUEBELLS SCHOOL INTERNATIONAL | XII-A

ABOUT THE PROJECT

In this project we use pretrained Machine Learning model to predict the denomination of Indian currency using the machine's camera in real-time. The program shows the preview of the captured frames along with vital info like the prediction, the probability of the prediction being correct and in future versions it would be moulded into an android and IOS app and will be able to speak out the predicted class of the currency along the huge model improvements. This app could become useful for people with visual-impairment.

TARGETED USERS FOR THE APP

Consider this scenario, a lone visually-impaired person with no trusted person goes to a store to buy something. They take out the cash from their pocket but don't know how much they are giving to the owner they ask the owner to take the required amount and return the rest, some owners might be honest, some might not be. So, in order to protect potential victims of being robbed our app can come into play.

Now consider this, the person buys the stuff they need and take out the cash from their pockets instead of relying in the store owner or some random by passer to help them they can simply take out their phone and know the amount they are giving. In this case the person can't be robbed because they are not trusting anyone with their money.

MODULES AND TECHNOLOGY USED

We used a variety of modules to create this project, they can be seen below: -

- TensorFlow - TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. We used this module to train a Convolutional Neural Network on our particular use case with data downloaded from the web.

- NumPy - NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. We used this module to do all the mathematical tasks and reshaping images.

- Keras - Keras is an open-source high-level library that provides a Python interface for neural networks. Keras acts as an interface for the TensorFlow library. Now it is integrated into TensorFlow and was used to create the neural network.

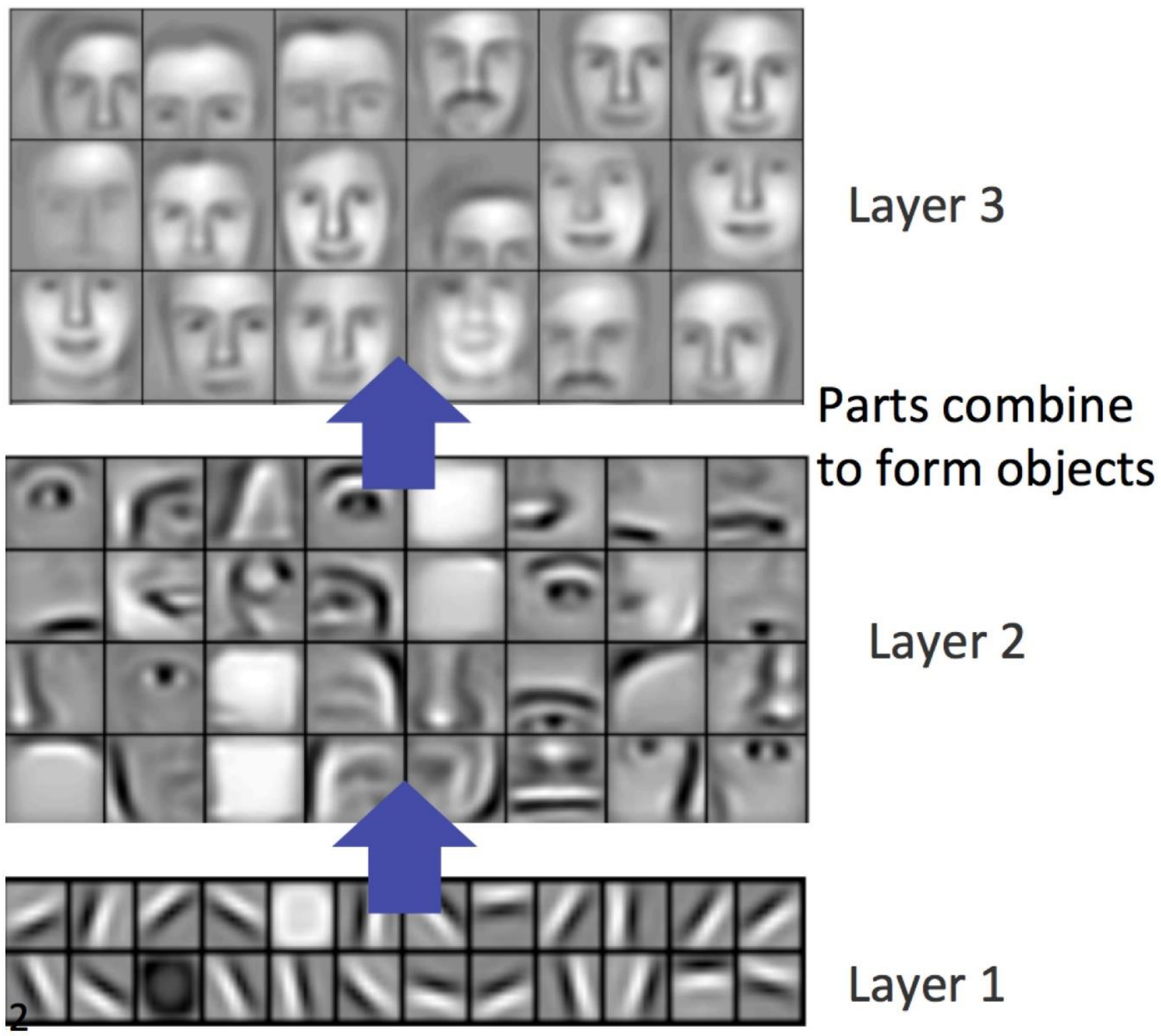
- Python - Python is an interpreted high-level general-purpose programming language. It was used as the base language to write the entire codebase and all the libraries are built in Python.

- OpenCV - OpenCV is a library of programming functions mainly aimed at real-time computer vision. We used this library to capture and pre-process the video feed from the camera to make it ready to be fed into the model. It also helped in displaying the text on the screen.

CONVOLUTIONAL NEURAL NETWORK

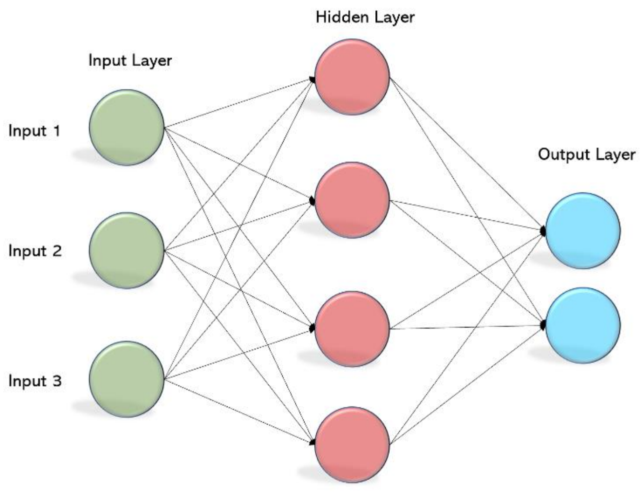
To build this model we used a Convolutional Neural Network, which is specifically designed to work with image data. The process we followed is given below: -

1. First the image was passed though a few convolutional layers to extract low-level features from the images such as lines, curves, and other patterns using filters. A filter is a matrix of numbers which when multiplied with the image pixels can show the low-level features.



The low-level features are then combined with other low-level features to give high-level features such as numbers, patterns on the notes etc. this is done to get a sense of the image.

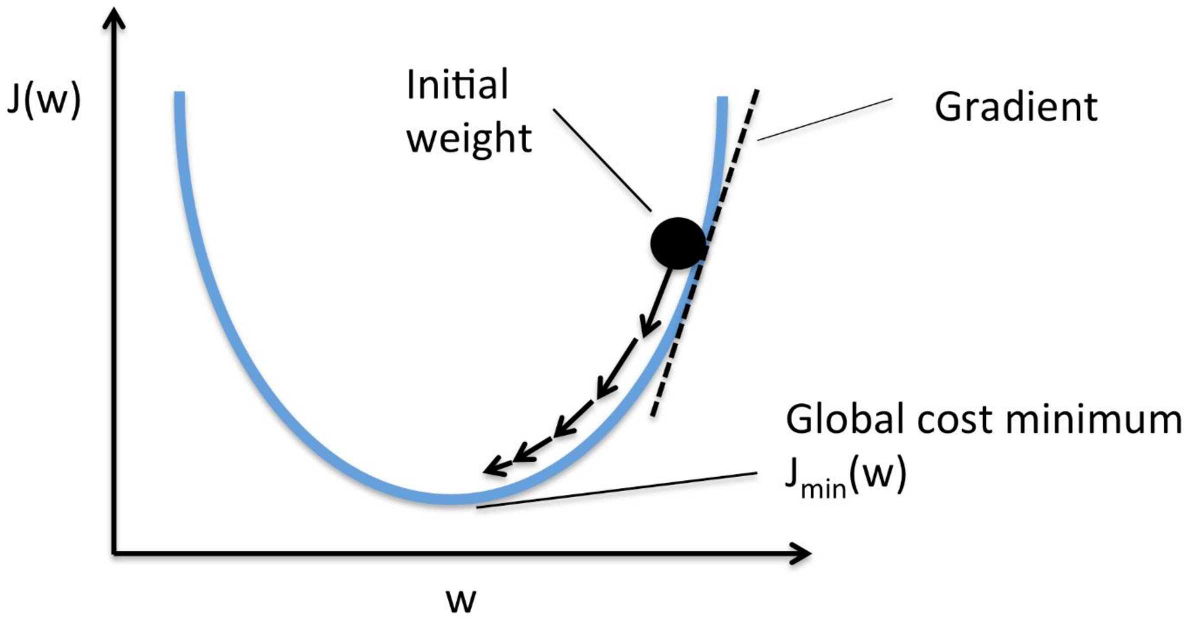
2. Then the image is flattened and passed through a multi-layered perception or a fully-connected Neural Network in which the pixels are multiplied by a weight matrix and a bias is added to get a prediction.



3. There are non-linear activation functions that are added in intermediate layers to capture the non-linearities of the image because the image won't always have a linear relationship with the prediction.

4. With the prediction in hand, we tally it with the label and find out the cost function or loss.

5. Then we backpropagate to find the gradients of each weight matrix with respect to the loss and update the weight matrix to minimize the cost function so that we come close to the actual label. This is called Gradient Descent.



6. Once we have trained the model, we save the weights to use in the future.

OPENCV FOR VIDEO CAPTURING

We use the OpenCV library for capturing the video feed and applying some pre-processing to be fed in to the model. The steps are given below: -

1. We specify the device we want to use to capture the video feed. Then we set the dimensions of the preview window.

```
frameWidth= 640         # CAMERA RESOLUTION
frameHeight = 480
brightness = 180
threshold = 0.5         # PROBABLITY THRESHOLD
font = cv2.FONT_HERSHEY_SIMPLEX

cap = cv2.VideoCapture(0)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
cap.set(10, brightness)
```

2. We load the model weights that we trained before for inference on test data which is in this case real-world data.

```
model = tf.keras.models.load_model('keras_model.h5')
```

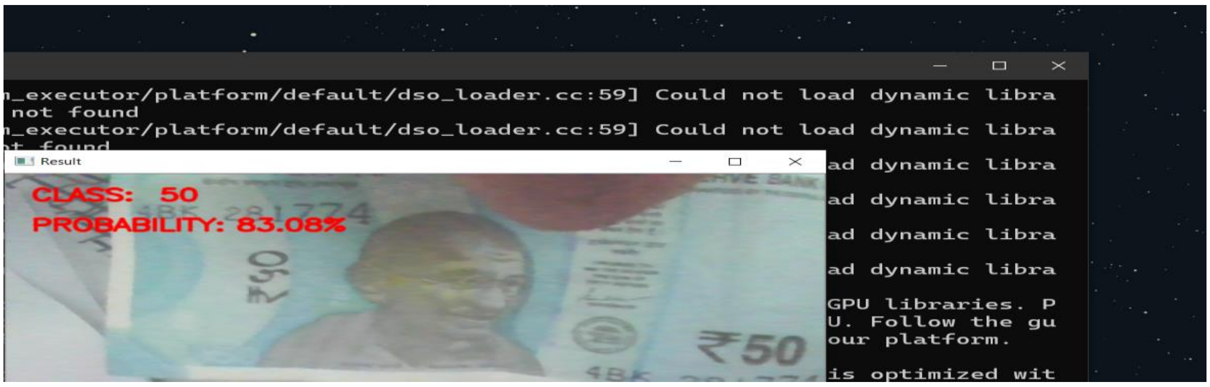
3. We create the pre-processing functions

```
def grayscale(img):
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    return img

def preprocessing(img):
    img = grayscale(img)
    img = img/255.0
    return img
```

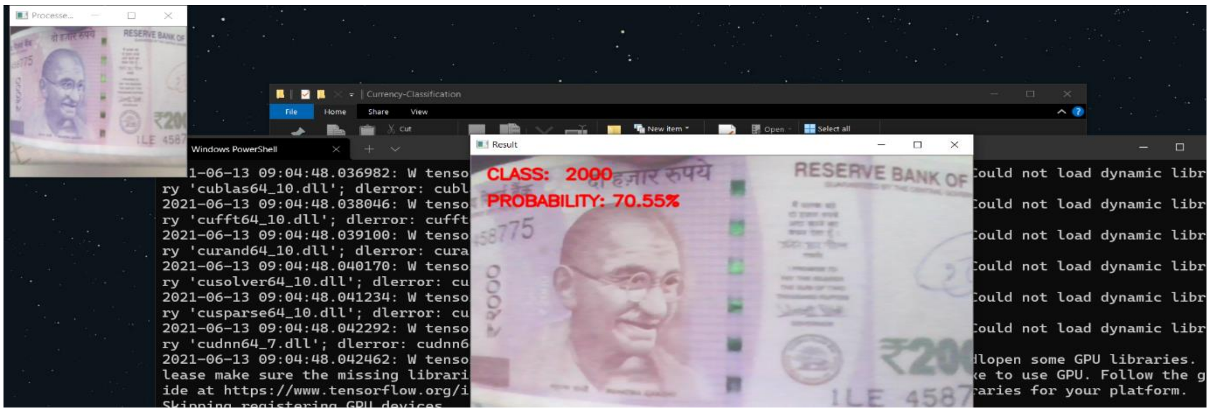
4. We capture the frames of the video and pass it into the model to get a prediction and display the prediction and probability of that prediction in the preview window.

```
while True:

    # READ IMAGE
    _, imgOrignal = cap.read()

    # PROCESS IMAGE
    img = np.asarray(imgOrignal)
    img = cv2.resize(img, (224, 224))
    img = preprocessing(img)
    cv2.imshow("Processed Image", img)
    img = img.reshape(1, 224, 224, 3)
    cv2.putText(imgOrignal, "CLASS: " , (20, 35), font, 0.75, (0, 0, 255), 2, cv
    cv2.putText(imgOrignal, "PROBABILITY: ", (20, 75), font, 0.75, (0, 0, 255),
    # PREDICT IMAGE
    predictions = model.predict(img)
    result = np.argmax(predictions)
    classIndex = classes[result]
    probabilityValue = np.amax(predictions)
    if probabilityValue > threshold:
        #print(getCalssName(classIndex))
        cv2.putText(imgOrignal,classIndex, (120, 35), font, 0.75, (0, 0, 255), 2
        cv2.putText(imgOrignal, str(round(probabilityValue*100,2) )+"%", (180, 7
    cv2.imshow("Result", imgOrignal)

    if cv2.waitKey(1) and 0xFF == ord('q'):
        break
```

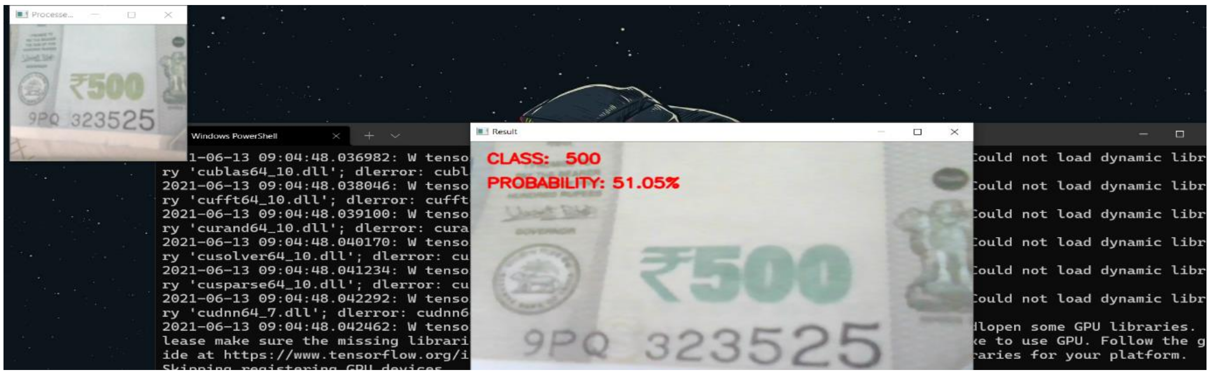
SAMPLE PREDICTIONS OF THE APP



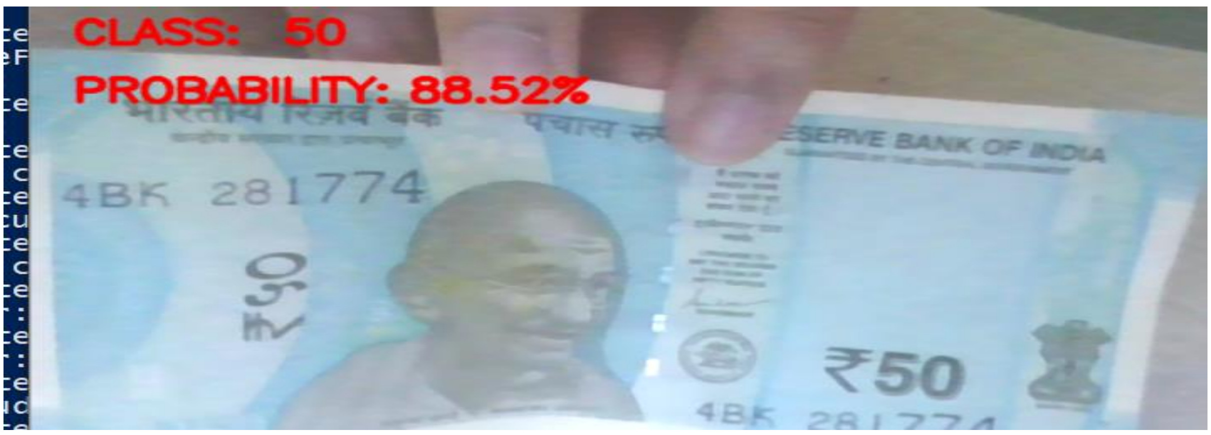
```
n_executor/platform/default/dso_loader.cc:59] Could not load dynamic libra
 not found
n_executor/platform/default/dso_loader.cc:59] Could not load dynamic libra
nt found
                                              ad dynamic libra
                                              ad dynamic libra
                                              ad dynamic libra
                                              ad dynamic libra
                                              GPU libraries. P
                                              U. Follow the gu
                                              our platform.
                                              is optimized wit
```



```
1-06-13 09:04:48.036982: W tenso          Could not load dynamic libr
ry 'cublas64_10.dll'; dlerror: cubl
2021-06-13 09:04:48.038046: W tenso       Could not load dynamic libr
ry 'cufft64_10.dll'; dlerror: cufft
2021-06-13 09:04:48.039100: W tenso       Could not load dynamic libr
ry 'curand64_10.dll'; dlerror: cura
2021-06-13 09:04:48.040170: W tenso       Could not load dynamic libr
ry 'cusolver64_10.dll'; dlerror: cu
2021-06-13 09:04:48.041234: W tenso       Could not load dynamic libr
ry 'cusparse64_10.dll'; dlerror: cu
2021-06-13 09:04:48.042292: W tenso       Could not load dynamic libr
ry 'cudnn64_7.dll'; dlerror: cudnn6
2021-06-13 09:04:48.042462: W tenso       dlopen some GPU libraries.
lease make sure the missing librari       ke to use GPU. Follow the g
ide at https://www.tensorflow.org/i       raries for your platform.
Skipping registering GPU devices
```



```
1-06-13 09:04:48.036982: W tenso          Could not load dynamic libr
ry 'cublas64_10.dll'; dlerror: cubl       Could not load dynamic libr
2021-06-13 09:04:48.038046: W tenso
ry 'cufft64_10.dll'; dlerror: cufft       Could not load dynamic libr
2021-06-13 09:04:48.039100: W tenso
ry 'curand64_10.dll'; dlerror: cura       Could not load dynamic libr
2021-06-13 09:04:48.040170: W tenso
ry 'cusolver64_10.dll'; dlerror: cu       Could not load dynamic libr
2021-06-13 09:04:48.041234: W tenso
ry 'cusparse64_10.dll'; dlerror: cu       Could not load dynamic libr
2021-06-13 09:04:48.042292: W tenso
ry 'cudnn64_7.dll'; dlerror: cudnn6
2021-06-13 09:04:48.042462: W tenso       dlopen some GPU libraries.
lease make sure the missing librari       ke to use GPU. Follow the g
ide at https://www.tensorflow.org/i       raries for your platform.
Skipping registering GPU devices
```



FUTURE OF THIS PROJECT

Right now, the model is pretty basic and only has an accuracy of 68% but the accuracy can be pulled to around 98-99% with the following modifications: -

- Adding more data, or generating data using Generative Adversarial Networks.
- Adding a validation set to avoid overfitting to the training data.
- Augmenting data to get more data in different conditions.
- Using Transfer Learning from different state of the art models.
- Playing with Hyper-Parameters.
- Creating a database to improve the network as more and more people use it.
- Creating a full-fledged android and IOS app.
- Adding a feature sum up all the money to show the final amount.
- Training on coins to predict their denomination.

REFERENCES/BIBLIOGRAPHY

- ***Introduction to Neural Networks***
  https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

- ***Understanding of a convolutional neural network***
  https://ieeexplore.ieee.org/document/8308186

- ***How to Visualize Filters and Feature Maps in Convolutional Neural Networks*** https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/

- ***Visualizing Convolutional Neural Nets***
  https://www.cs.ryerson.ca/~aharley/vis/conv/

- ***TensorFlow Documentation***
  https://www.tensorflow.org/api_docs/python/tf/all_symbols

- ***Keras Getting Started***
  https://keras.io/getting_started/

- ***Multi-Label Classification***
  https://towardsdatascience.com/multi-label-classification-and-class-activation-map-on-fashion-mnist-1454f09f5925