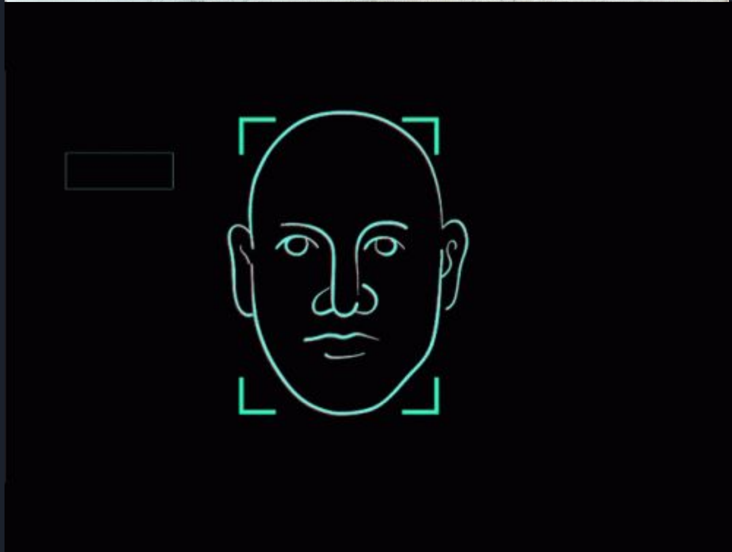
Currency/Bill Recognition

- Mohammad Hossain
- Nicholas Gryzb
- Jairo Iqbal Gil
- Midyan Elghazali

Table of Contents:



INTRODUCTION

Goal: *Develop a sophisticated machine learning model using Python, which will enable a webcam integrated into the program to accurately identify and classify bills as USD, EURO, or Egyptian currency. This technology aims to facilitate real-time currency recognition, enhancing applications in automated financial systems, aiding in transactions, and providing valuable assistance in settings where quick and reliable bill identification is crucial.*

Real World Benefits: *This technology aims to facilitate real-time currency recognition, enhancing applications in automated financial systems, aiding in transactions, and providing valuable assistance in settings where quick and reliable bill identification is crucial. Using a webcam for live detection make this solution particularly applicable in dynamic environments, paving the way for innovative applications in commerce, travel, and accessibility for the visually impaired.*

LIBRARIES: Tensorflow, OpenCV, YOLOv5

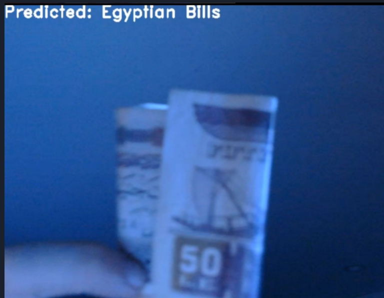
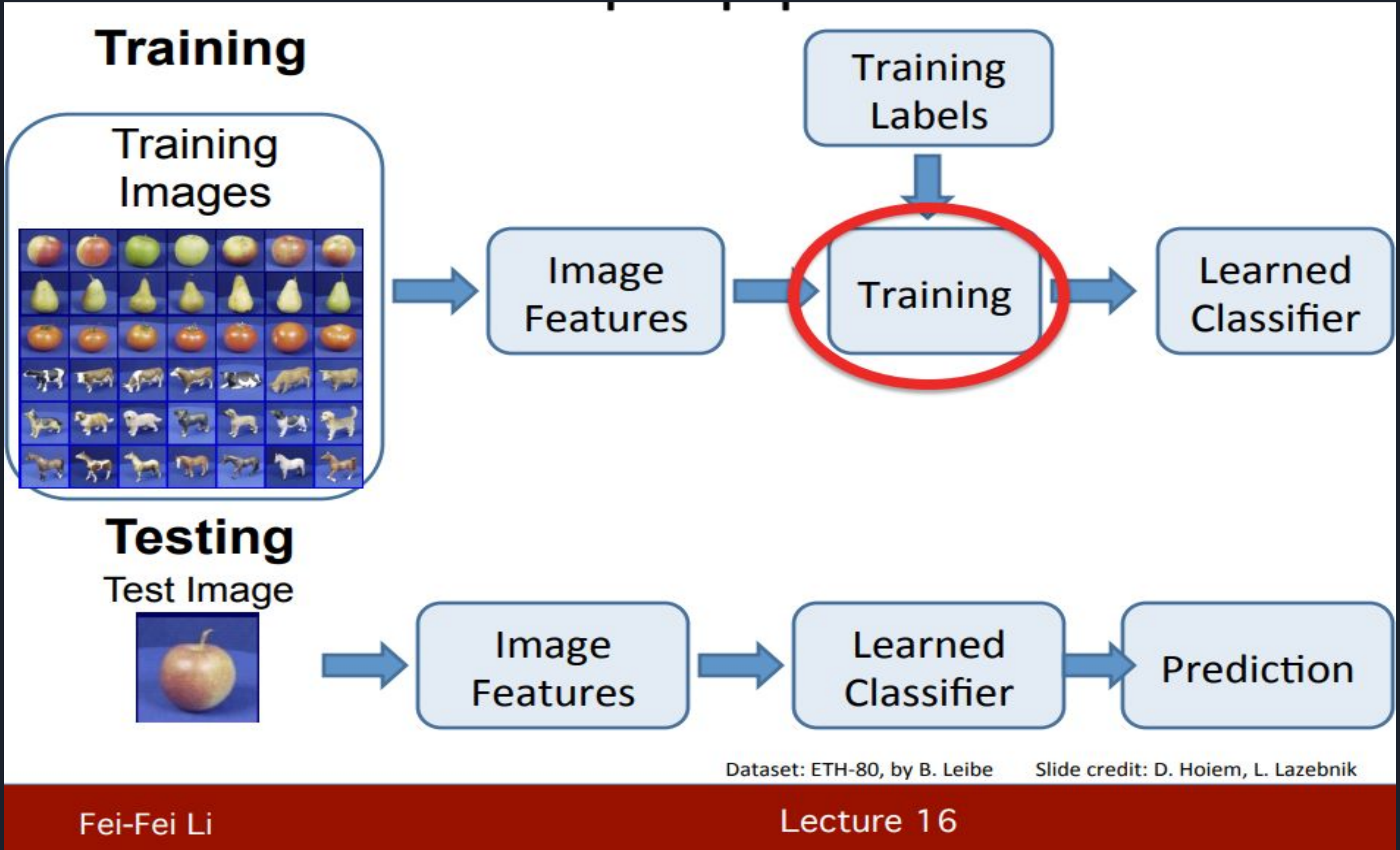


IMAGE PROCESSING



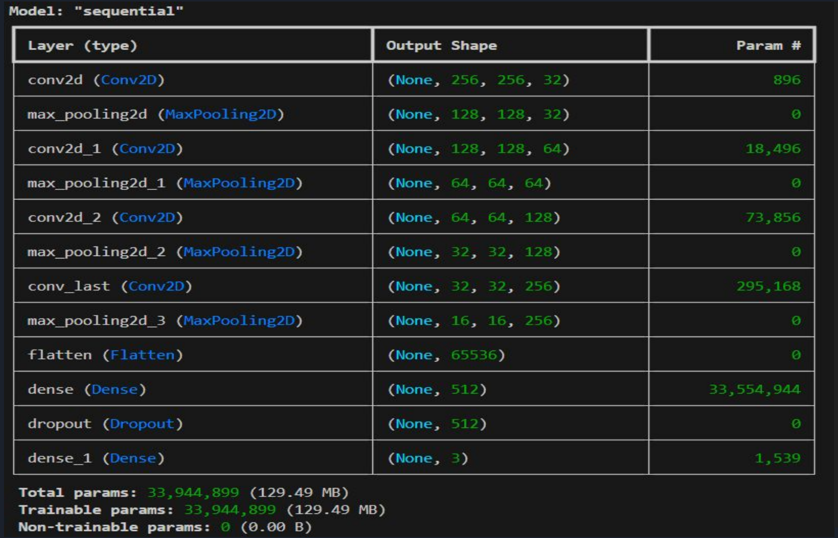
STEPS FOR IMPLEMENTATION (Tensorflow)

We've built a sequential linear model to accurately recognize different types of currency. It consists of 11 layers that work together to process images:

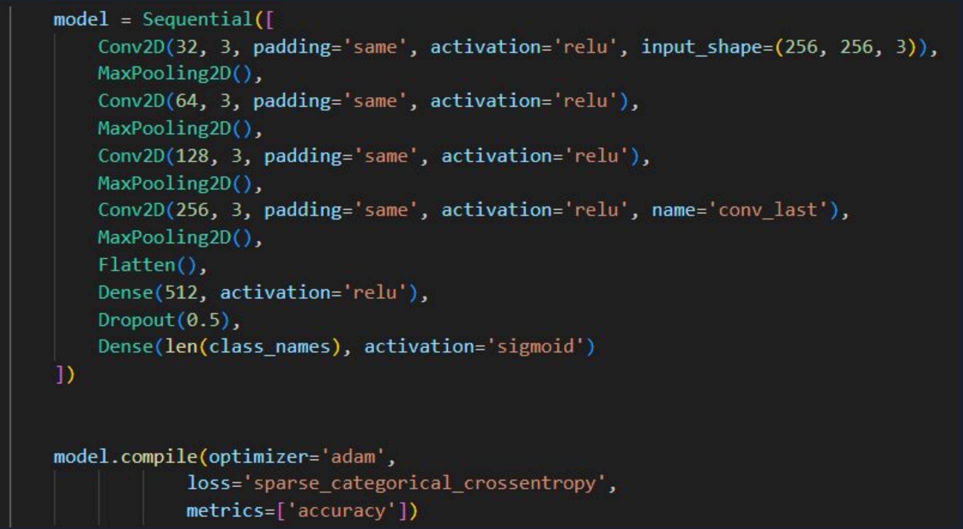
- Convolutional Layers: These are four layers designed to pick up features like edges and textures from the images, using small 3x3 filters.
- ReLU Activation: This helps our model make sense of complex patterns without getting bogged down.
- Pooling Layers: After each feature-detecting layer, we have pooling layers that shrink the image size down, making our model faster and more efficient.
- Flatten and Dense Layers: Near the end, we flatten the output to a single list and process it through dense layers to finalize our understanding of the image.
- Output Layer: The last layer decides what type of currency the image shows, using a simple decision function.

Why this design? It's simple to train and runs quickly, making it both powerful and practical for recognizing money efficiently.

Tensorflow Visualizer



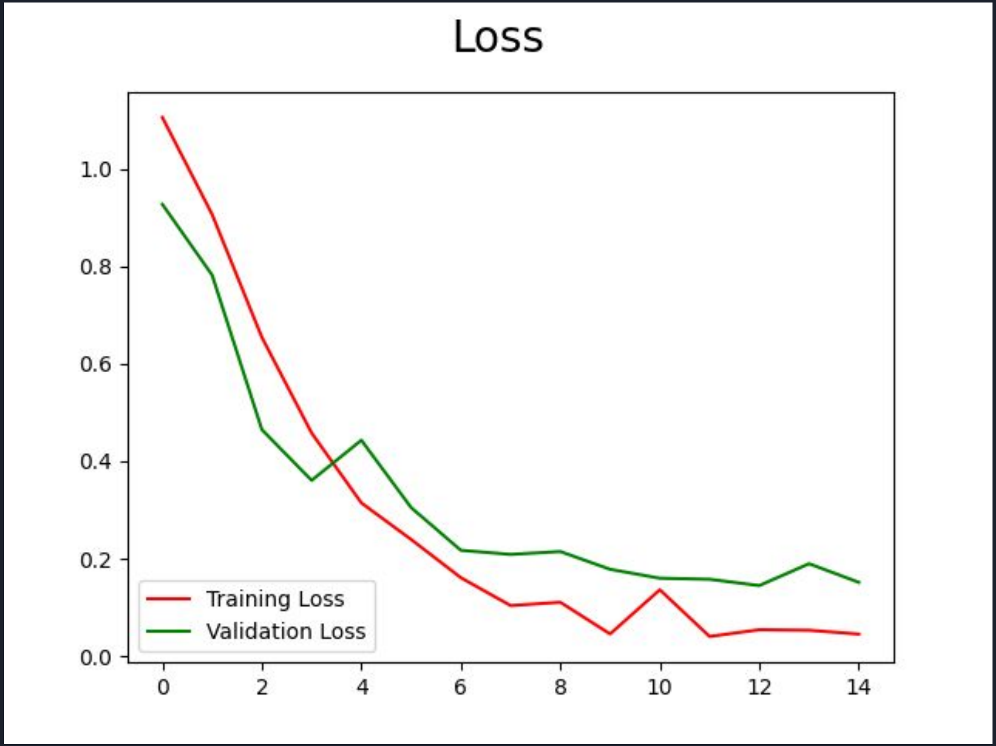
```
Model: "sequential"
┌────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                   │ Output Shape           │       Param # │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d (Conv2D)                │ (None, 256, 256, 32)   │           896 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)   │ (None, 128, 128, 32)   │             0 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)              │ (None, 128, 128, 64)   │        18,496 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D) │ (None, 64, 64, 64)     │             0 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_2 (Conv2D)              │ (None, 64, 64, 128)    │        73,856 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_2 (MaxPooling2D) │ (None, 32, 32, 128)    │             0 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ conv_last (Conv2D)             │ (None, 32, 32, 256)    │       295,168 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_3 (MaxPooling2D) │ (None, 16, 16, 256)    │             0 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)              │ (None, 65536)          │             0 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                  │ (None, 512)            │    33,554,944 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)              │ (None, 512)            │             0 │
├────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                │ (None, 3)              │         1,539 │
└────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 33,944,899 (129.49 MB)
 Trainable params: 33,944,899 (129.49 MB)
 Non-trainable params: 0 (0.00 B)
```

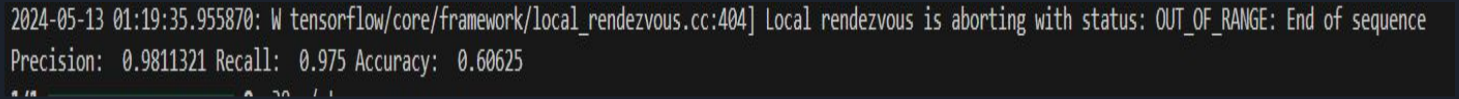


```python
model = Sequential([
    Conv2D(32, 3, padding='same', activation='relu', input_shape=(256, 256, 3)),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(128, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(256, 3, padding='same', activation='relu', name='conv_last'),
    MaxPooling2D(),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(len(class_names), activation='sigmoid')
])


model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

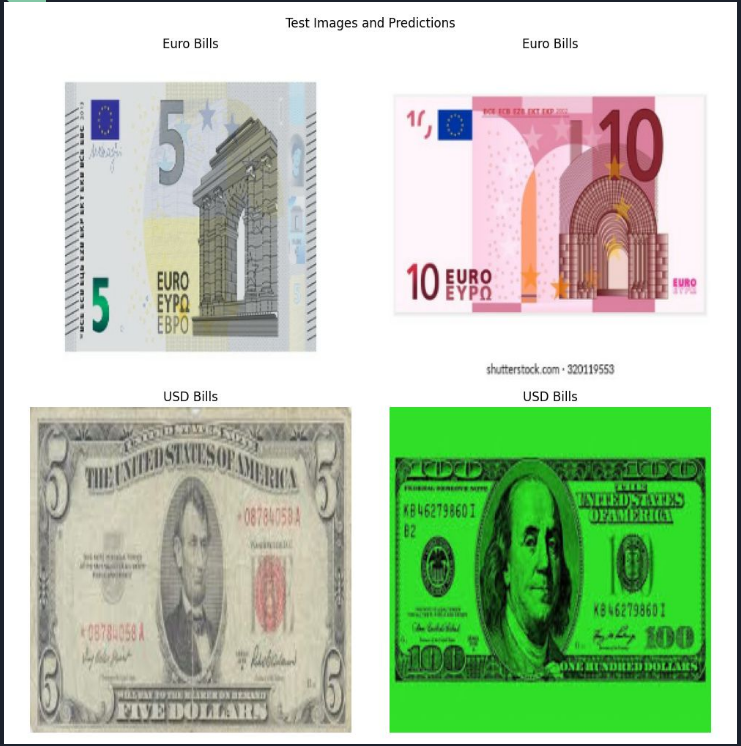
Enough Talking, How Good Is It? Well...

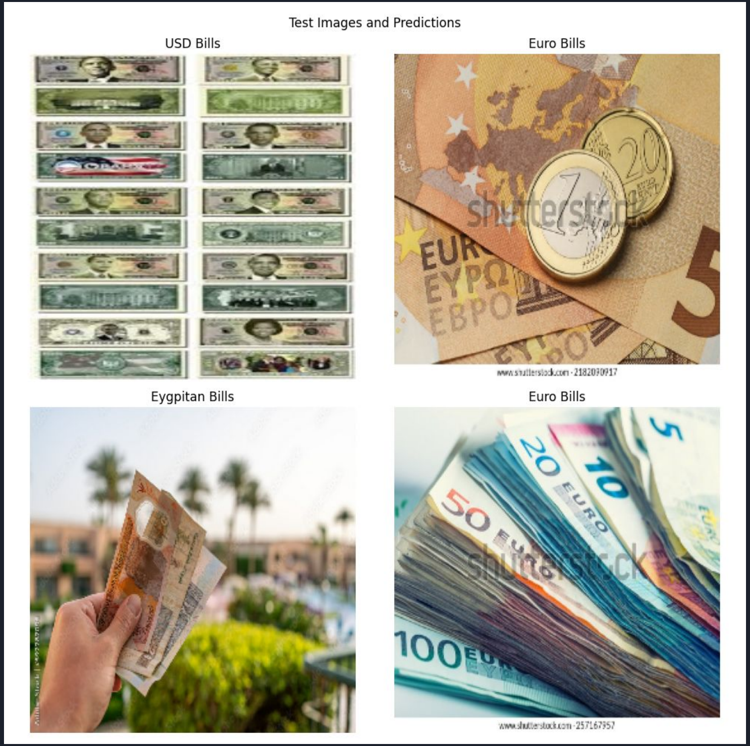




```
2024-05-13 01:19:35.955870: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
Precision:  0.9811321 Recall:  0.975 Accuracy:  0.60625
```

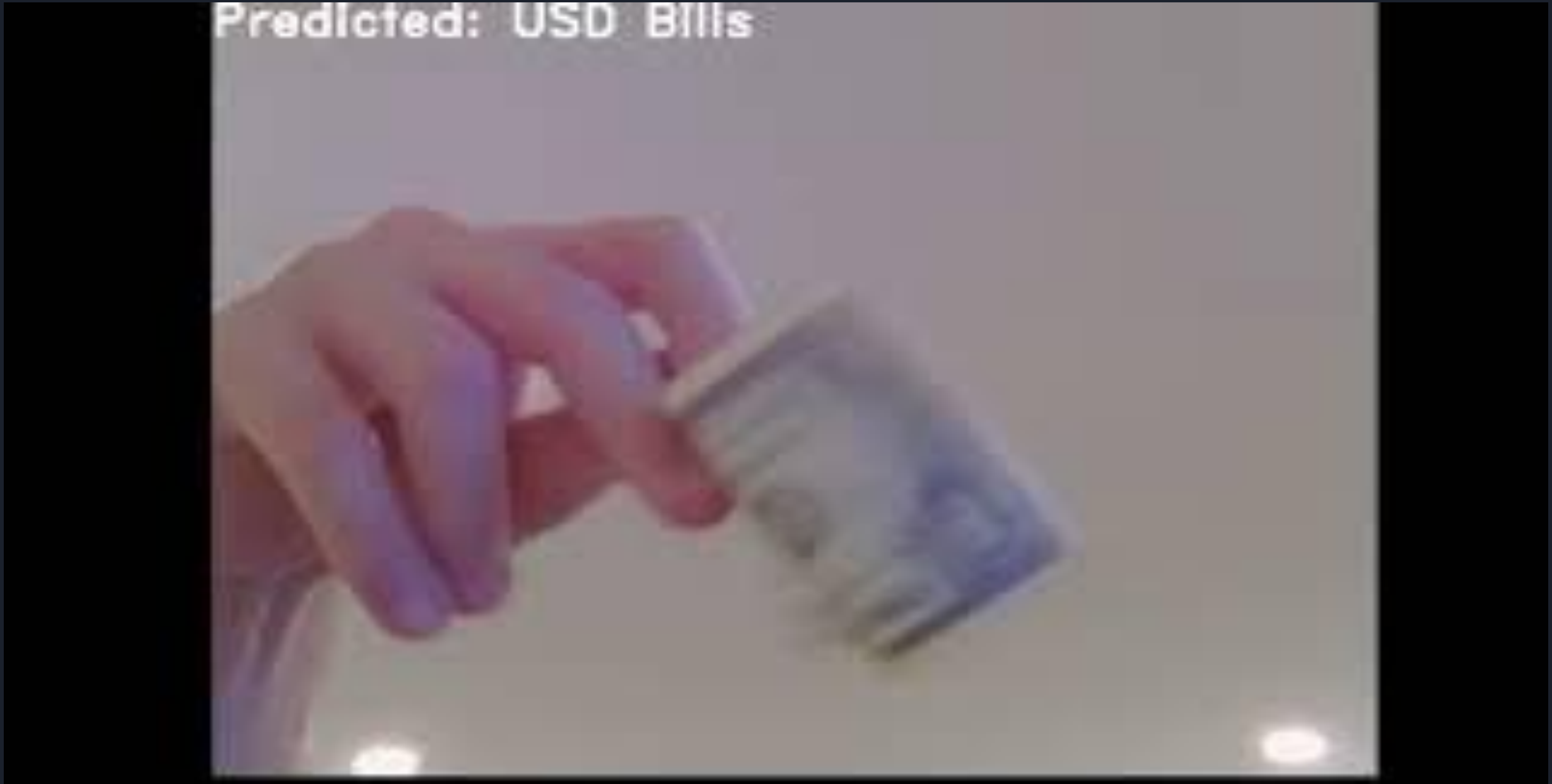
Ok But That's Just Graph Thingy, Where Is The Proof?





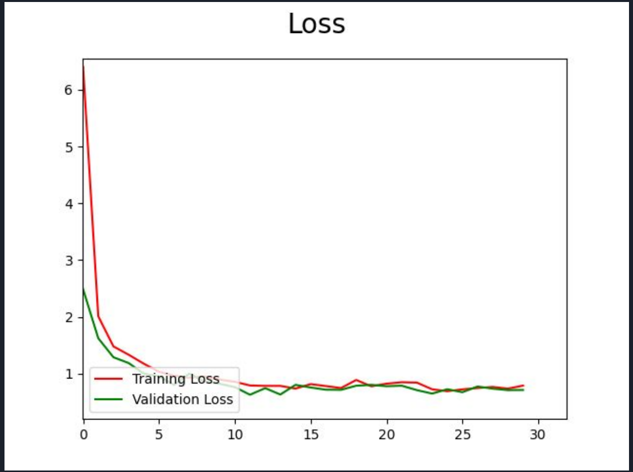
Pretty Good Isn't It?

But Thats All Talk, Let Me See A Demo. SUREEEE!!!

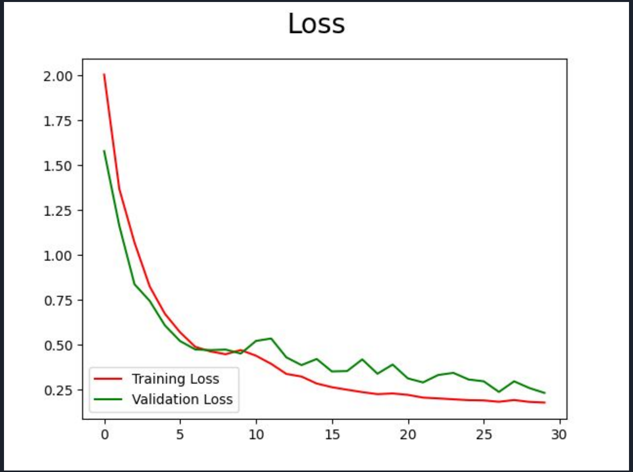


Not All Sunshine And Rainbow

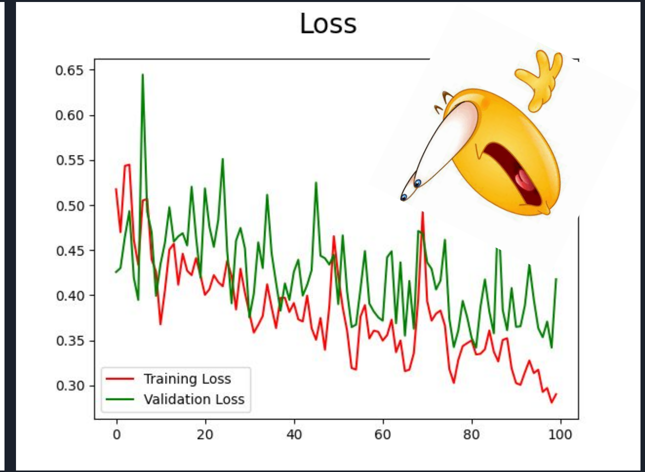
Model #1



Model #2

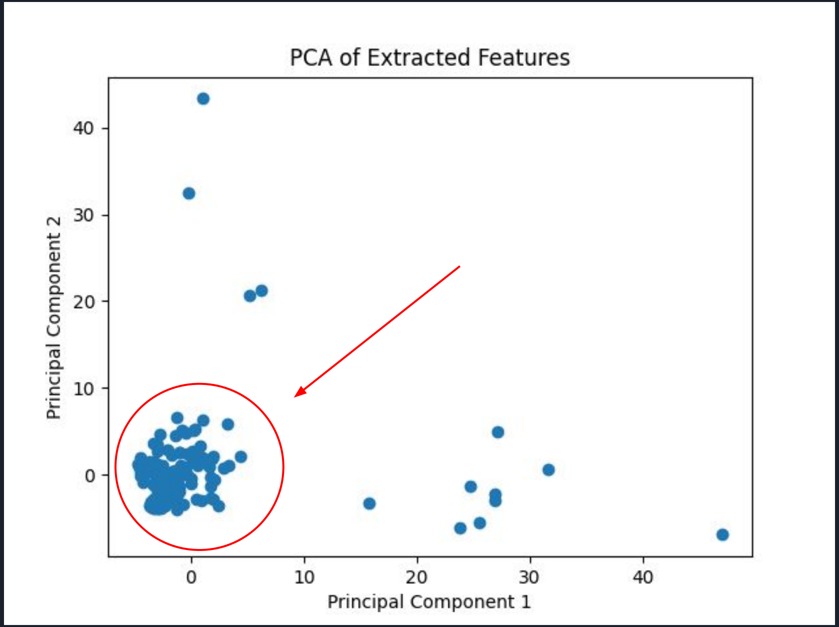


Model #3

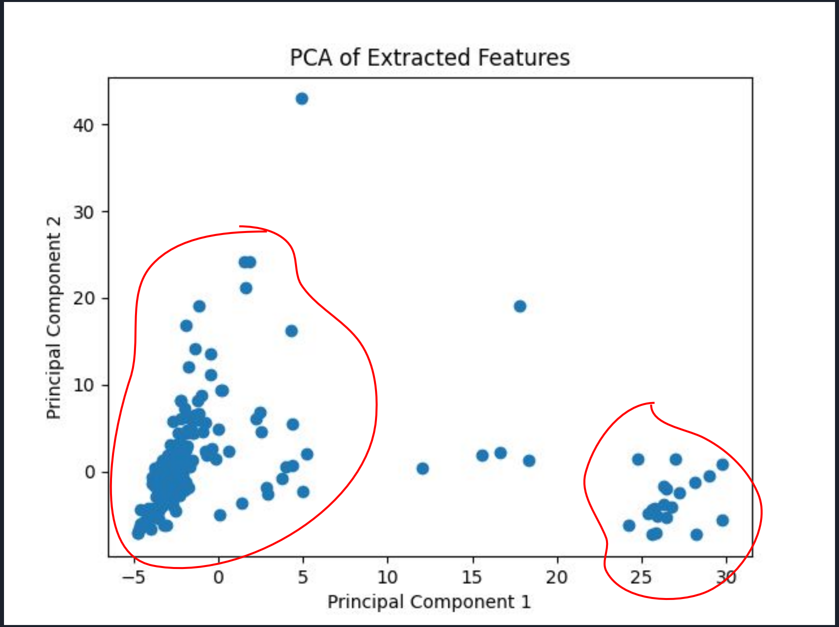


PCA Jumpscare :)

PCA For Test_Images



PCA For Validation



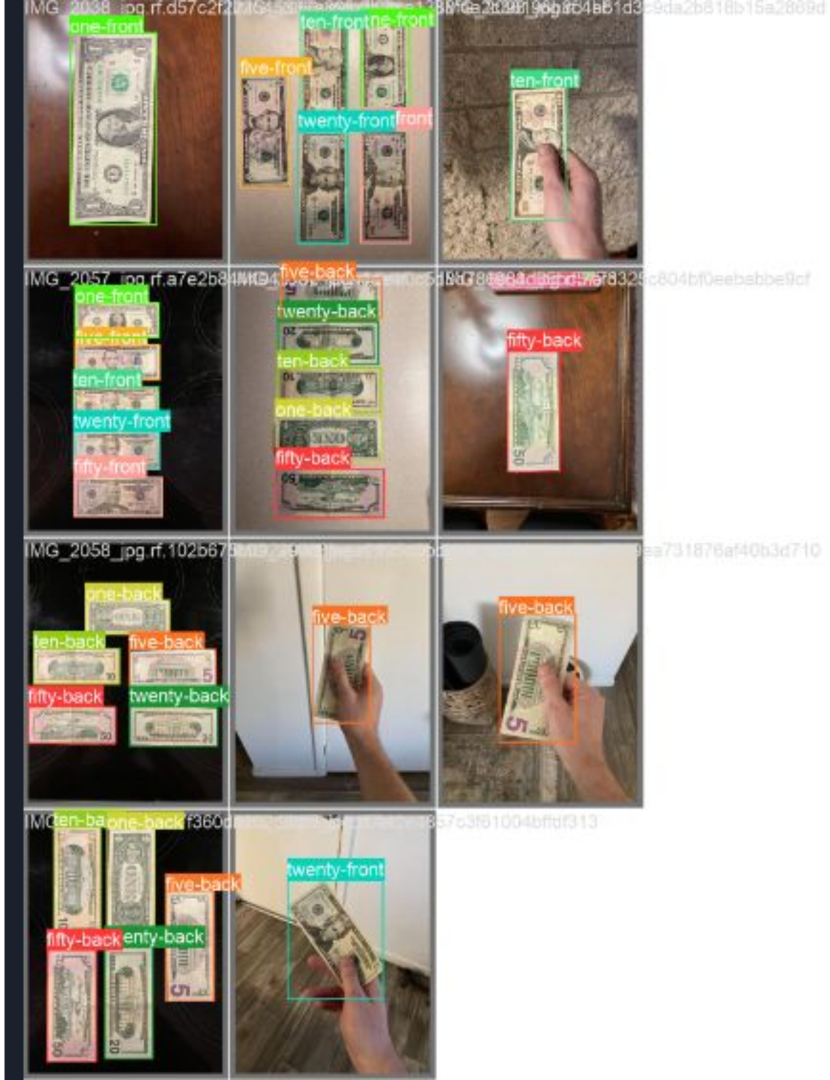
Using Yolov5:

Training Data Preparation: The success of YOLOv5 hinges on the availability and quality of training data. In this context, a diverse dataset of annotated currency images representing different denominations, orientations, lighting conditions, and backgrounds is essential. Preprocessing techniques such as data augmentation (e.g., rotation, scaling, flipping) may be employed to augment the dataset and improve model robustness.

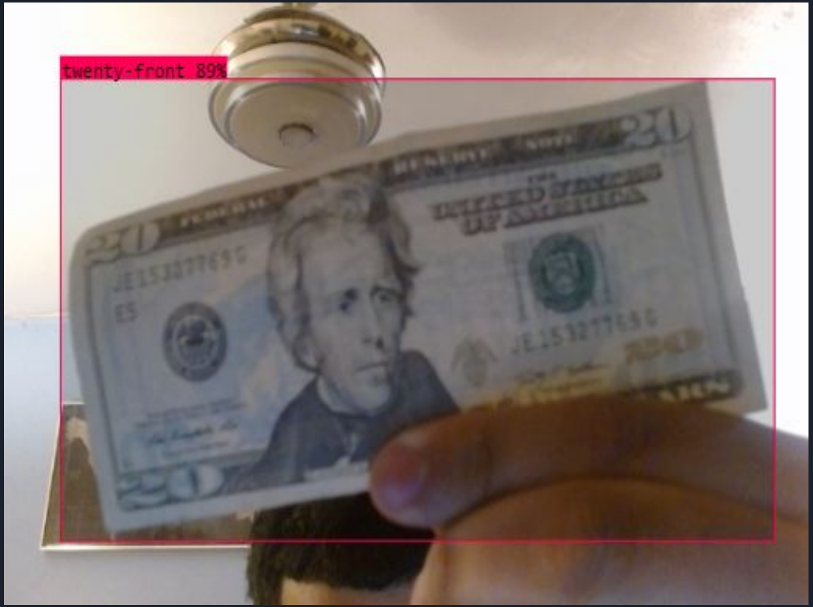
Training Procedure: Training YOLO v5 involves optimizing its parameters to minimize a predefined loss function (e.g., bounding box regression loss, classification loss). This process typically utilizes a large dataset and is computationally intensive, often requiring specialized hardware such as GPUs. Transfer learning, where a pre-trained model is fine-tuned on the currency detection dataset, can expedite training and improve convergence.

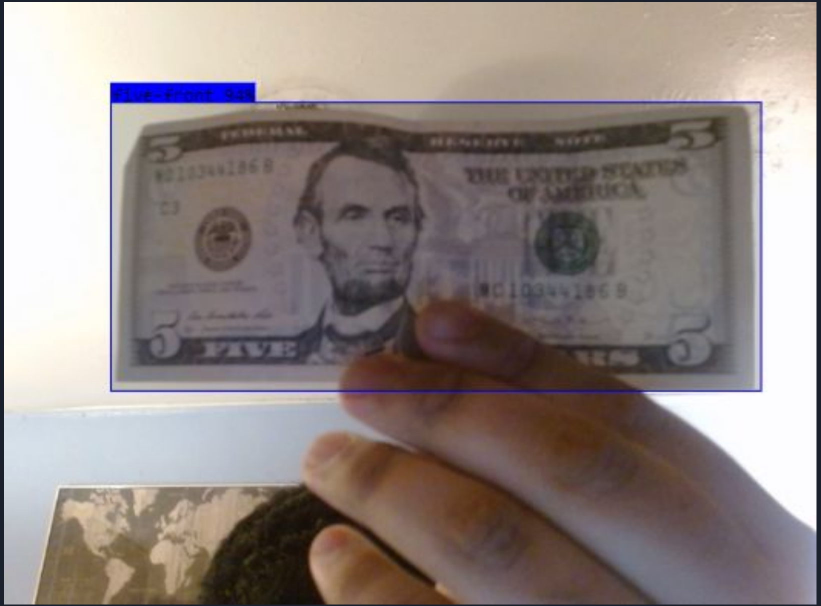
Performance Evaluation and Optimization: Finally, the performance of the YOLOv5-based currency detection system is evaluated using metrics such as precision, recall, and F1-score. Fine-tuning hyperparameters, optimizing model architecture, and conducting inference optimizations (e.g., quantization, model pruning) may be necessary to enhance performance and efficiency, especially in real-time applications.

IMG_2038_jpg.rf.d57c2f
IMG_2057_jpg.rf.a7e2b8
IMG_2058_jpg.rf.102b67
IMG_2059_jpg.rf.47f360
ea731876af40b3d710
57c3f61004bffdf313


IMG_2038_jpg.rf.d57c2f
one-front
ten-front
five-front
twenty-front
ten-front
IMG_2057_jpg.rf.a7e2b8
one-front
five-front
ten-front
twenty-front
fifty-front
five-back
twenty-back
ten-back
one-back
fifty-back
fifty-back
IMG_2058_jpg.rf.102b67
one-back
ten-back
five-back
fifty-back
twenty-back
five-back
five-back
ten-ba
one-back
five-back
fifty-back
enty-back
twenty-front

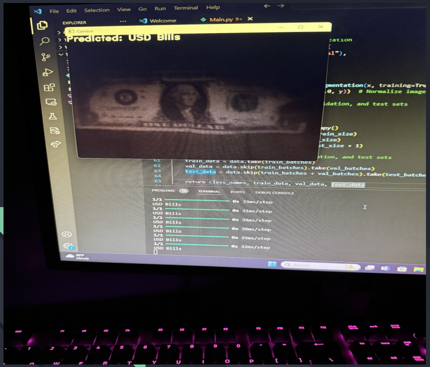
LABELED IMAGES

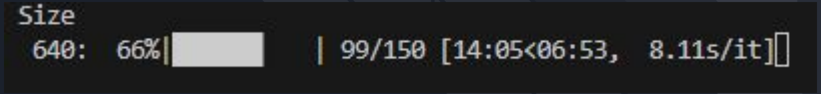




ISSUES/ERRORS

- In the early stages of our project, we ran into issues where the webcam would detect a currency even if there wasn't a bill being shown, and at some points, if the background of the webcam was a certain color, it would predict a currency that isn't even on the scene due to the colors of the pixels in the room. There were also cases of the webcam making predictions that just weren't correct and would change based on any movement being shown.
- Furthermore, we struggles with the hardware requirements to try and get a very accurate model working. Training a model took a lot of time since were we training locally and not on a very good GPU.
- The hardest part of the project was trying to find open source data or imaging bases that contained foreign bills to train our model with. Therefore the only currency we were able to get an abundance of was the US note.





```
Size
 640:  66%|█████     | 99/150 [14:05<06:53,  8.11s/it]
```

Bloopers

Predicted: Eygption Bills

How Can We Improve?

We are still working on enhancing the accuracy of our models, recognizing that several factors contribute to precision. One crucial aspect is the size and diversity of our image dataset. By incorporating a large and varied set of images, we can train our models to better recognize and distinguish between different currency denominations and variations. However, beyond just the quantity of images, we are also exploring methods to improve the quality and relevance of the dataset, ensuring it captures a comprehensive range of real-world scenarios.

Moreover, a key challenge lies in adapting our detection system to operate effectively in real 3D environments, as opposed to the controlled settings of 2D images with uniform backgrounds. This transition introduces complexities such as varying lighting conditions, perspective distortions, and occlusions that are inherent to real-world scenarios. To address this, we are actively researching and developing techniques to enable our models to accurately detect currencies within dynamic 3D environments. This involves integrating advanced computer vision algorithms and leveraging techniques like depth sensing and object tracking to enhance the robustness and adaptability of our detection system.

RESOURCES

TensorFlow:https://www.tensorflow.org/

YOLOv5:https://github.com/ultralytics/yolov5

OpenCV:https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html