A TEAM PRESENTATION OF "THE PROJECT"

Includes:

- Ezzy/JJ-Scrub Master/Coding
- Elias- Coding/Scrub Master/File Organizer/ Presenter
- Treston- Coding/File Organizer
- Nick- Coding/File Organizer/Presenter

THE OBJECTIVE

The objective for our project was to create a robot capable of recognizing a posture or signals when it appears in front of a camera. We focused on the classic game gestures: Rock, Paper, and scissors.

To accomplish this, we started by developing a database of images. We took several pictures of each gesture: rock, paper, and scissors. We then organized them into separate folders. These images then served as the training data for our robot.

After collecting the data, we then trained the robot to recognize each gesture accurately. So, when the camera detects someone showing a rock, paper, or scissor gesture, the robot can correctly identify and respond to it.

THE SETUP

To be able to make the robot recognize gesture, we had to use a program it and we decided to use the application Google Collab that uses the coding language python.

First we had to mount it to our drive so that it can access the drive and grab the information that it needs to be able to function:

"from google,colab import drive

drive.mount("/content/gdrive", force_remount=True)"

Once this was mounted we then had to load the pictures to the machine by using the zip file, "rps.zip," which had separate folders containing pictures of the different gestures, rock paper and scissors.

This is the entire code that was used to load the data to the machine

```
import cv2
import glob
import numpy as np

X_scissor = np.asarray([cv2.imread(file) for file in glob.glob('train/scissors/*.jpg')])
y_scissor = np.zeros(X_scissor.shape[0])
X_rock = np.asarray([cv2.imread(file) for file in glob.glob('train/rock/*.jpg')])
y_rock = np.zeros(X_rock.shape[0]) + 1
X_paper = np.asarray([cv2.imread(file) for file in glob.glob('train/paper/*.jpg')])
y_paper = np.zeros(X_paper.shape[0]) + 2

X_train = np.concatenate((X_scissor, X_rock, X_paper), axis=0)
y_train = np.concatenate((y_scissor, y_rock, y_paper), axis=0)

print(X_train.shape)
print(y_train.shape)

X_scissor = np.asarray([cv2.imread(file) for file in glob.glob('test/scissors/*.jpg')])
y_scissor = np.zeros(X_scissor.shape[0])
X_rock = np.asarray([cv2.imread(file) for file in glob.glob('test/rock/*.jpg')])
y_rock = np.zeros(X_rock.shape[0]) + 1
X_paper = np.asarray([cv2.imread(file) for file in glob.glob('test/paper/*.jpg')])
y_paper = np.zeros(X_paper.shape[0]) + 2

X_test = np.concatenate((X_scissor, X_rock, X_paper), axis=0)
y_test = np.concatenate((y_scissor, y_rock, y_paper), axis=0)

print(X_test.shape)
print(y_test.shape)
```

This leads to the Whole Baseline Project Dataset Example: base_2024_project_template_v5.ipynb

```
def draw_img(i):
    im = X_train[i]
    c = y_train[i]
    plt.imshow(im)
    plt.title("Class %d (%s)" % (c, class_name[c]))
    plt.axis('on')

def draw_sample(X, y, n, rows=4, cols=4, imfile=None,
fontsize=12):
    for i in range(0, rows*cols):
        plt.subplot(rows, cols, i+1)
        im = X[n+i].reshape(480,640,3)
        plt.imshow(im, cmap='gnuplot2')
        plt.title("{}".format(class_name[y[n+i]]), fontsize=fontsize)
        plt.axis('off')
        plt.subplots_adjust(wspace=0.6, hspace=0.01)
        #plt.subplots_adjust(hspace=0.45, wspace=0.45)
        #plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=1.0)
    if imfile:
        plt.savefig(imfile)

nb_classes = 3
class_name = {
    0: 'scissor',
    1: 'rock',
    2: 'paper'
}

draw_img(0)

draw_sample(X_train, y_train, 0,3,5)
```

THE METRICS

Class	TP	FN	FP	TN	Recall	Precision
Scissors	5	1	0	12	0.8333333333	1
Rock	6	0	2	10	1	.75
Paper	4	2	1	11	0.66666666667	.8

Chat GPT

Imagine we have a robot that is trying to guess whether an object is a scissor, rock, or paper. The table shows us how well our robot did in this guessing game for each type of object. Let's break down what each part of the table means and how to interpret it.

Table Breakdown:

1. **Class**: This is the type of object our robot is trying to guess (scissor, rock, or paper).
2. **TP (True Positives)**: The number of times our robot correctly guessed the object. For example, if it guessed "scissor" and it really was a scissor.
3. **FN (False Negatives)**: The number of times our robot missed the object. For example, if it should have guessed "rock" but didn't.
4. **FP (False Positives)**: The number of times our robot guessed wrong. For example, if it guessed "paper" but there was no paper.
5. **TN (True Negatives)**: The number of times our robot correctly said the object wasn't there. For example, if it guessed "not scissor" and there really wasn't a scissor.

What Each Metric Means:

- **Recall**: This is how good our robot is at finding all the objects it's supposed to find. For example, if there are 6 rocks and it finds all 6, the recall is 1.0 (which is perfect).
- **Precision**: This tells us how many of the robot's guesses were correct. If it guessed 8 times that something was a scissor and was right 6 times, the precision is 6 out of 8 (or 0.75).
- **Accuracy**: This shows how often the robot's guesses were correct overall. If it made 20 guesses and 18 of them were right, the accuracy is 18 out of 20 (or 0.9).
- **F1 Score**: This combines precision and recall into one number. It helps us understand how well the robot balances between being correct when it guesses and finding all the objects it's supposed to find.
- **Specificity**: This tells us how good our robot is at correctly identifying when an object is not there. For example, if it should correctly say "not paper" and does so 10 out of 11 times, the specificity is about 0.91.
- **MCC (Matthews Correlation Coefficient)**: This is a more advanced score that combines all the other numbers to give a single score between -1 and 1. A score of 1 means the robot is perfect, 0 means it's guessing randomly, and -1 means it's always wrong.
- **Jaccard Index**: This measures how often the robot's guesses match the actual objects. If it usually guesses right, this number will be high.

Interpreting the Table:

For each type of object (scissor, rock, paper), the table tells us:

- **Scissor**: The robot guessed "scissor" correctly 5 times (TP), missed it once (FN), never guessed "scissor" when it wasn't there (FP), and correctly said it wasn't "scissor" 12 times (TN). It's very good at identifying scissors with high recall and precision.
- **Rock**: The robot guessed all 6 rocks correctly (TP), didn't miss any rocks (FN), but guessed "rock" incorrectly 2 times (FP). This means it's very good at finding rocks but makes a few mistakes by guessing rock when it's not there.
- **Paper**: The robot guessed "paper" correctly 4 times (TP), missed it 2 times (FN), and guessed "paper" incorrectly 1 time (FP). This means it missed some papers, so its recall is lower, but when it does guess "paper," it's usually right.

Summary:

The table gives us a detailed look at how well our robot is playing the "guess the object" game. We can see how often it gets each type of object right, how often it misses, and how accurate its guesses are overall. Each metric helps us understand different parts of the robot's performance:

- **Recall** tells us how good it is at finding objects.
- **Precision** tells us how accurate its guesses are.
- **Accuracy** gives us an overall success rate.
- **F1 Score** balances between recall and precision.
- **Specificity** shows how good it is at saying something isn't there.
- **MCC** combines all these into one score.
- **Jaccard Index** measures the similarity between its guesses and the actual objects.

These metrics help us see where our robot is doing well and where it can improve.

DATA SET INFORMATION

File Name: rps.zip (Rock, Paper, Scissors)

Dataset Source: Proprietary dataset from ASU students.

Number of samples: 60

Image Dimensions: 480x640x3

Class Labels: Rock, Paper, and Scissors.

Distribution: 20/20/20

Data augmentation: TensorFlow's and Keras's (keras.preprocessing.image.ImageDataGenerator)

Ex: Input: (480,640,3) Output: (240, 320, 4)

Highest accuracy: 50%

WHAT WE LEARNED

We will explain what we learned and what to avoid.

Learned:

- Organize the files you download on your pc.
- When coding, be very specific and precise, or else the coding cell may not work.
- Take good photos, so that ai can recognize them to get high accuracy.

Avoidance:

- Having too many tabs open.
- Files of the same name
- Unorganized files