Vision Team

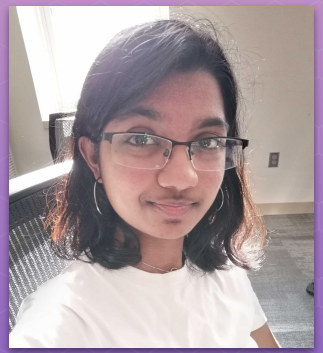
Sahil Raina, Janani Kannan,
Angie Chen, Tyler Pham,
Stephen Ma

Team

Sahil Raina

Fun Fact: Too good at Procrastination
Year: Second Year
Major: Computer Science

Janani Kannan



Fun Fact: I can solve the Megaminx (12-sided Rubik's cube)
Year: Freshman
Major: Computer Science

Angie Chen

Fun Fact: I ~~am obsessed with~~ like birds
Year: Freshman
Major: Computer Science

Team

Tyler Pham

Fun Fact: I dance for Choreos
Year: Freshman
Major: Computer Science

Stephen Ma

Fun Fact: I'm also a Music major
Year: Freshman
Major: Computer Science

INTRODUCTION

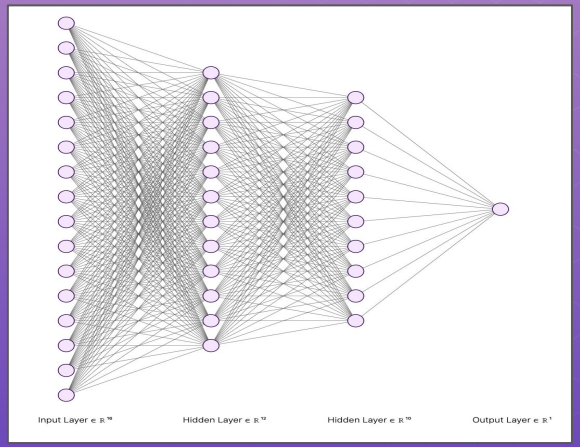
The Vision Team is a subteam that focused on developing the Vision Module for the project as to classify waste into biodegradable and non-biodegradable materials for automatic sorting.

OUR Process


Data Collection
Searched for and used the a related DataSet from Kaggle
Building the Model
Developed a CNN Model using Tensorflow
Testing
Tested and improved the model using the Raspberry Pi attached to the bin

TECHNOLOGIES

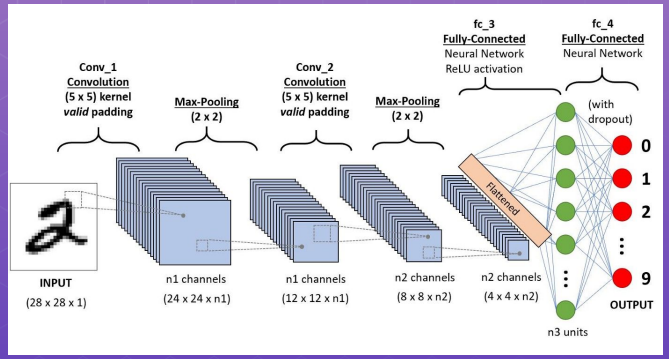
```
'''CNN'''
def build_model(hp):
  model = keras.Sequential([
    keras.layers.Conv2D(
        filters=hp.Int('conv_1_filter', min_value=32, max_value=128, step=16),
        kernel_size=hp.Choice('conv_1_kernel', values = [3,5]),
        activation='relu',
        input_shape=(50,50,1)
    ),
    keras.layers.Conv2D(
        filters=hp.Int('conv_2_filter', min_value=32, max_value=64, step=16),
        kernel_size=hp.Choice('conv_2_kernel', values = [3,5]),
        activation='relu'
    ),
    keras.layers.Flatten(),
    keras.layers.Dense(
        units=hp.Int('dense_1_units', min_value=32, max_value=128, step=16),
        activation='relu'
    ),
    keras.layers.Dense(10, activation='softmax')
  ])
```



Neural Networks and CNN

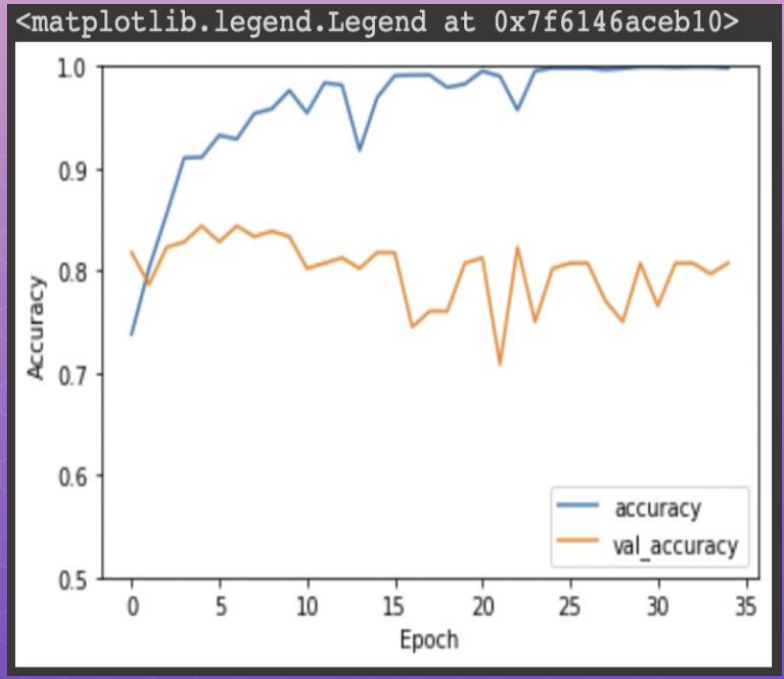
Convolutional Neural Networks (CNN)

- Common image classification algorithm
- Based on identifying and weighting key features to classify an image
- Three Key Components: Convolutional Layers, Pooling Layers, and Activation Layers



Prediction and Analysis

- Model accuracy:
  - accuracy and val_accuracy – what are they?
- Training epochs
  - How does changing the epochs affect accuracy?



Tensorflow Lite and Setup

- Original model is converted into a TFLITE model

```
tf_lite_converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = tf_lite_converter.convert()
tflite_model_name = "TFLITE_MODEL.tflite"
open(tflite_model_name, "wb").write(tflite_model)
```

- Necessary modules are installed onto Raspbian OS
  - tflite_runtime.interpreter
    - Part of the TensorFlow module that is lighter and only contains the necessary components for runtime
    - Intended for use in embedded systems
  - PIL
    - Contains Image, which is used to preprocess the camera photo
  - Numpy
- Configure the pi

Tensorflow Lite – Model

- Code components:
  - Read TFLITE model/get information
  - Camera code (took the photo to the right)
  - Read and preprocess image data
  - Run model to identify image

```
#uses camera to take photo; CAMERA_TIMER is the wait time
def takePhoto():
  camera = PiCamera()
  camera.start_preview()
  sleep(CAMERA_TIMER)
  camera.capture(PHOTO_PATH)
  camera.stop_preview()
```

```
[ ]
#preprocess the image
def modify_image(image):
  width = WIDTH
  height = HEIGHT

  image = ImageOps.grayscale(image)
  image = image.resize((height, width))
  image = np.array(image)
  image = np.expand_dims(image, axis=-1)
  image = image.astype(np.float32)

  resized_image = image

  return np.expand_dims(resized_image, axis=0)
```

Tensorflow Lite – Model

Reads the TFLite model

```
interpreter = tflite.Interpreter(model_path=str(TF_LITE_MODEL_FILE_NAME))
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()[0]
output_details = interpreter.get_output_details()[0]
```

```
[ ]
#classify the image into one of two classes:
#0 - non-biodegradable, 1 - biodegradable
#also returns accuracy
def run_tflite_model(interpreter, test_image):
    interpreter.set_tensor(input_details["index"], test_image)
    interpreter.invoke()
    output = interpreter.get_tensor(output_details["index"])[0]

    outname = output_details['name']

    # print("output details:", output_details)

    prediction  = output.argmax()
    print("output:", output)

    return prediction,output[prediction]

```

IDs the camera image

Challenges and Resolution

- Solving the bugs in the TFLite model
  - Finding different code sources, understanding the code, and testing each individual function
- Issues with installing Open-CV
  - Image from PIL used as a substitute
- Low prediction accuracies
  - Altering the CNN model
  - Experimenting with changes to layer types, training epochs, and number of layers

THANK YOU

ANY QUESTIONS?