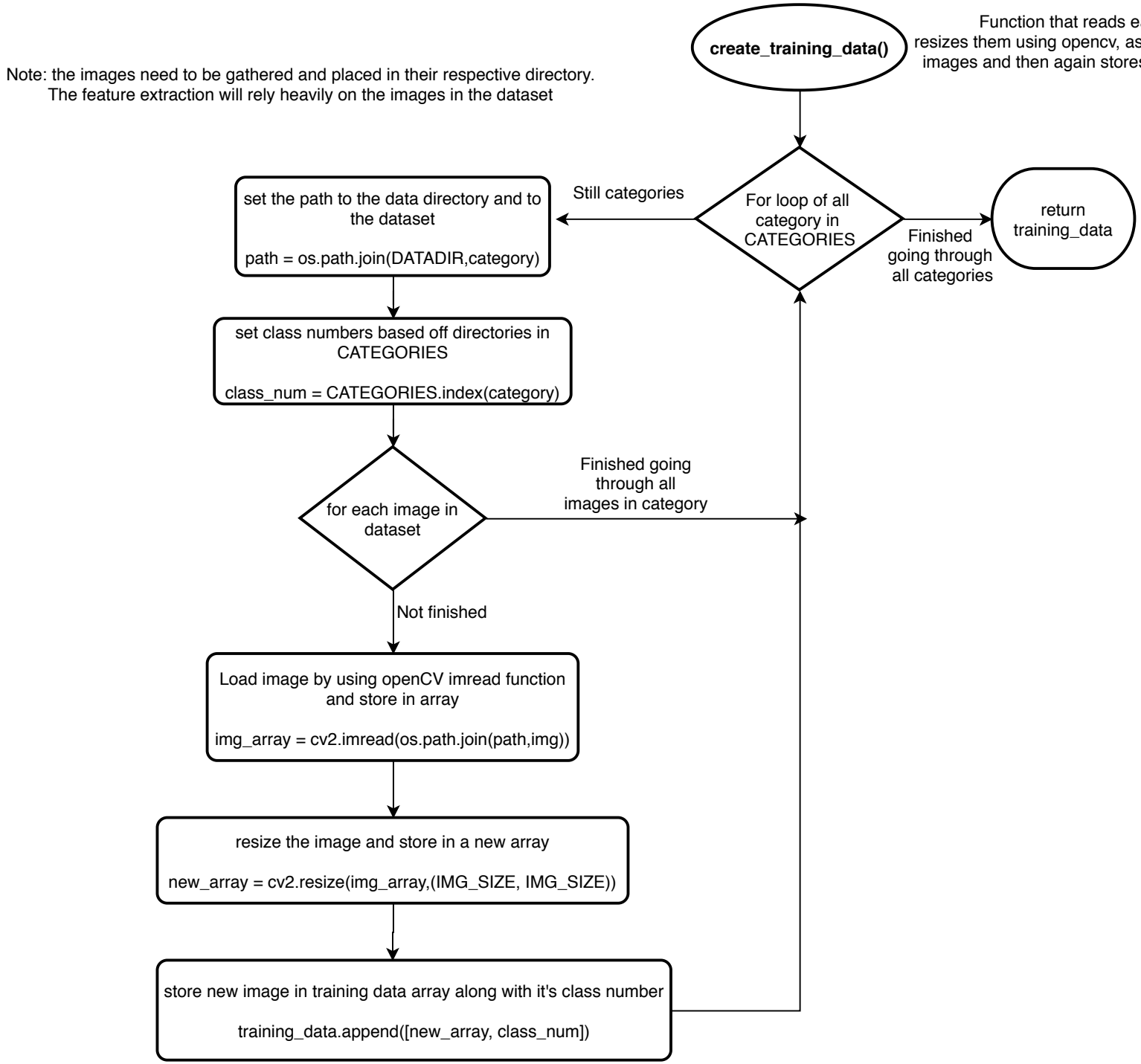
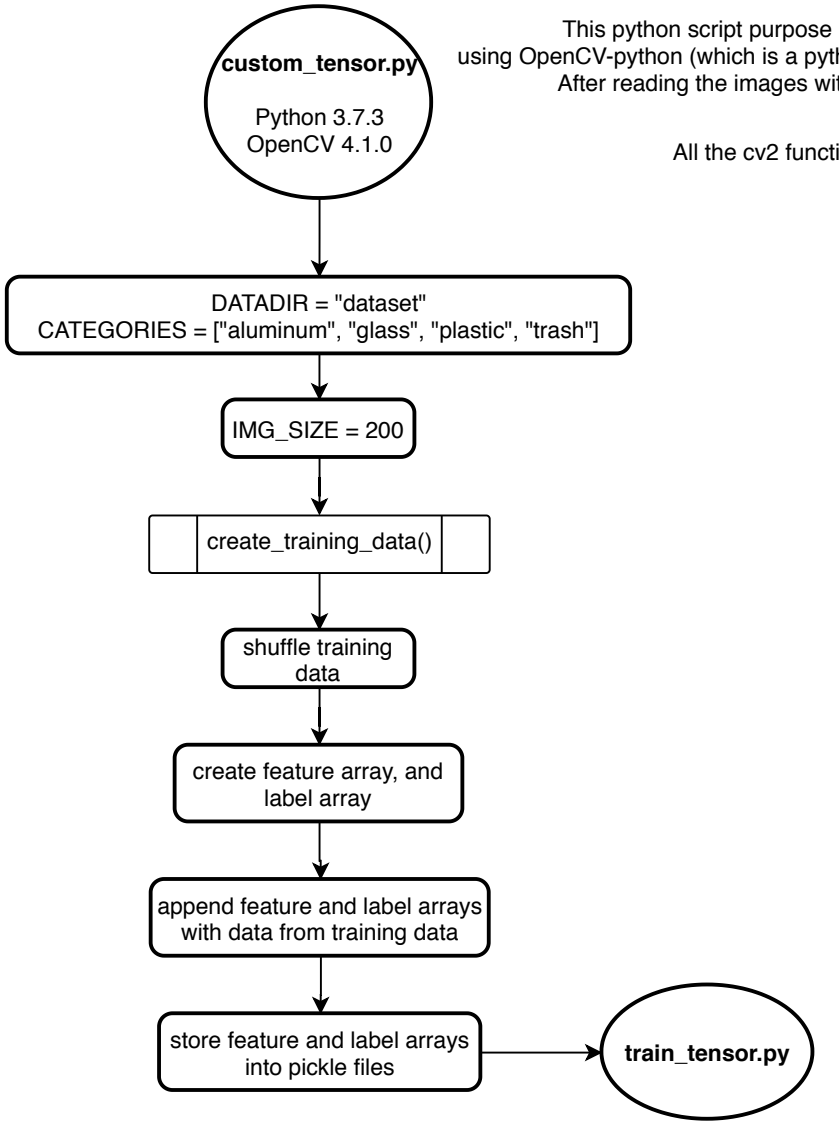
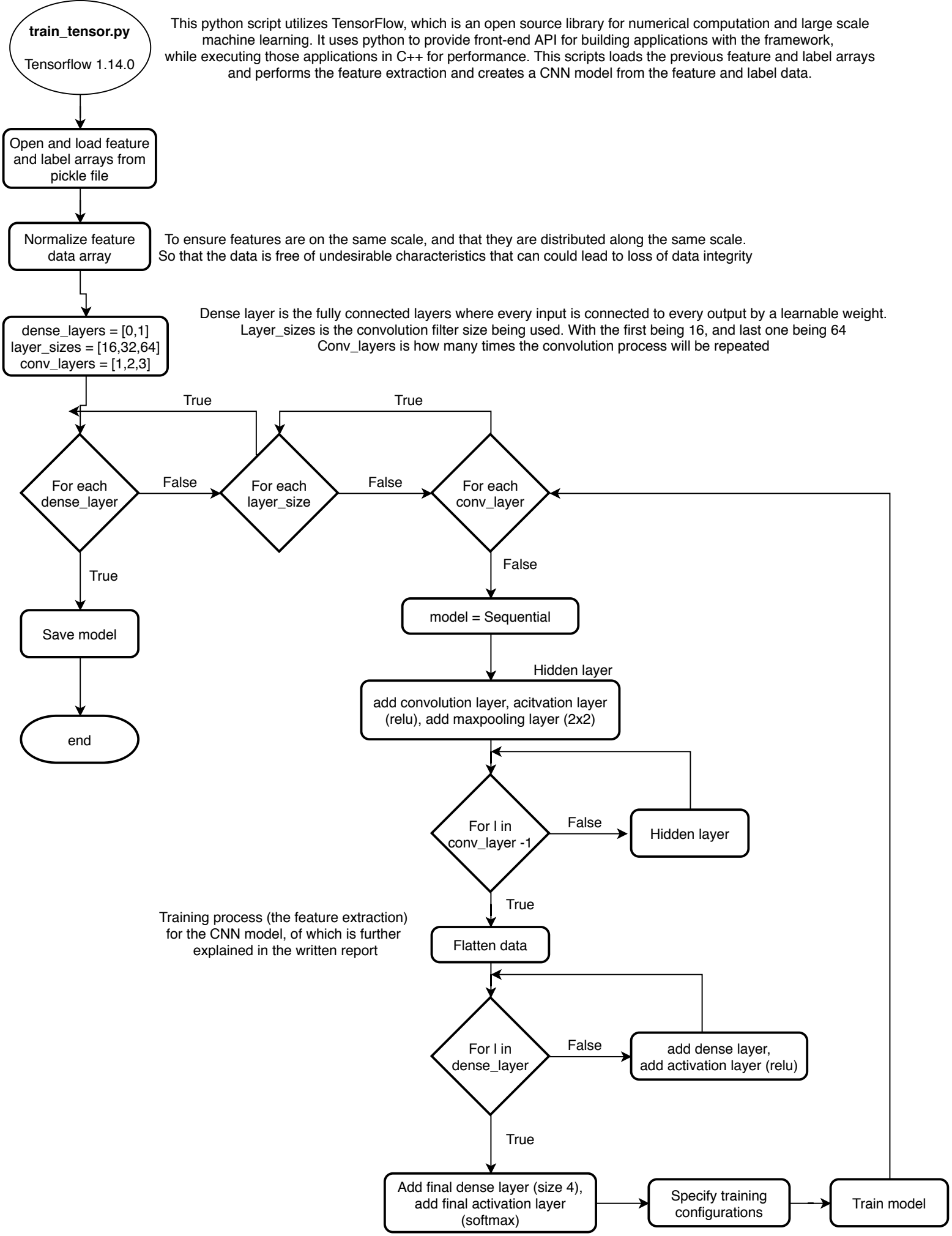

create_training_data()
Function that reads each image, resizes them using opencv, assigns a class to the images and then again stores it in another array
Note: the images need to be gathered and placed in their respective directory. The feature extraction will rely heavily on the images in the dataset
For loop of all category in CATEGORIES
Still categories
Finished going through all categories
return training_data
set the path to the data directory and to the dataset
path = os.path.join(DATADIR,category)
set class numbers based off directories in CATEGORIES
class_num = CATEGORIES.index(category)
for each image in dataset
Finished going through all images in category
Not finished
Load image by using openCV imread function and store in array
img_array = cv2.imread(os.path.join(path,img))
resize the image and store in a new array
new_array = cv2.resize(img_array,(IMG_SIZE, IMG_SIZE))
store new image in training data array along with it's class number
training_data.append([new_array, class_num])

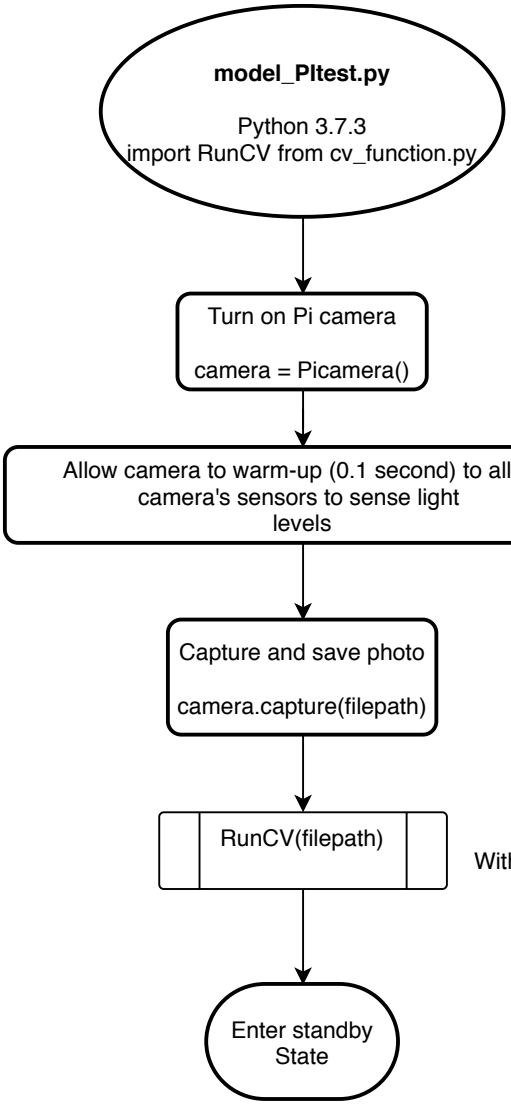


This python script purpose is to read the images that will be used to train the CNN model
using OpenCV-python (which is a python wrapper of opencv, since it is actually C++ code in the background).
After reading the images with opencv, it is stored in stored and saved into an numpy array

All the cv2 functions are instances of when OpenCV-Python is being used

Pickling is a way to convert a python object (list, dict, etc.)
into a character stream. The idea is that this character stream contains
all the information necessary to reconstruct the object in another python script.

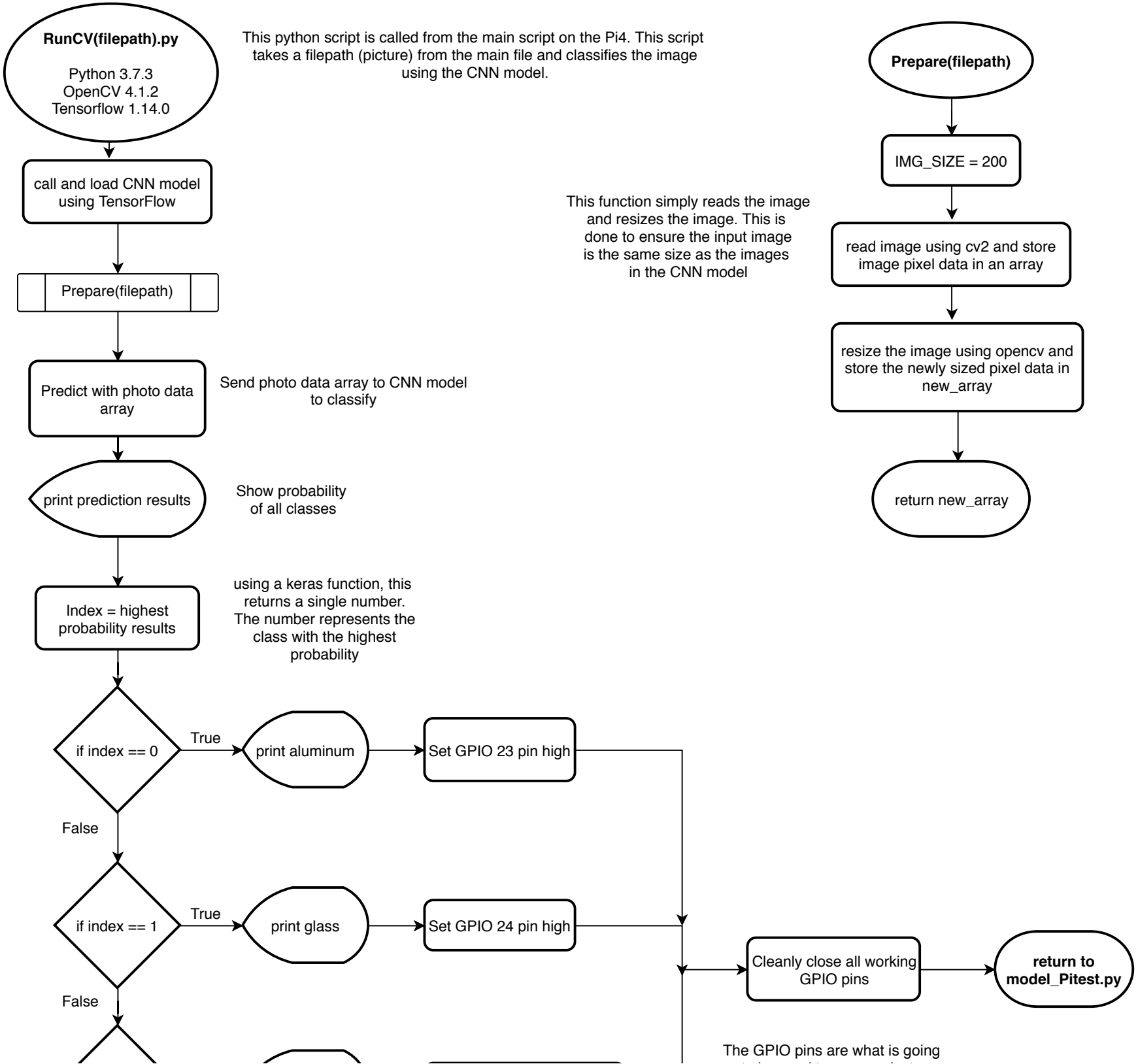

train_tensor.py
Tensorflow 1.14.0
This python script utilizes TensorFlow, which is an open source library for numerical computation and large scale machine learning. It uses python to provide front-end API for building applications with the framework, while executing those applications in C++ for performance. This scripts loads the previous feature and label arrays and performs the feature extraction and creates a CNN model from the feature and label data.
Open and load feature and label arrays from pickle file
Normalize feature data array
To ensure features are on the same scale, and that they are distributed along the same scale. So that the data is free of undesirable characteristics that can could lead to loss of data integrity
dense_layers = [0,1]
layer_sizes = [16,32,64]
conv_layers = [1,2,3]
Dense layer is the fully connected layers where every input is connected to every output by a learnable weight.
Layer_sizes is the convolution filter size being used. With the first being 16, and last one being 64
Conv_layers is how many times the convolution process will be repeated
True
True
For each dense_layer
False
For each layer_size
False
For each conv_layer
True
False
Save model
model = Sequential
Hidden layer
add convolution layer, acitvation layer (relu), add maxpooling layer (2x2)
end
For l in conv_layer -1
False
Hidden layer
True
Training process (the feature extraction) for the CNN model, of which is further explained in the written report
Flatten data
For l in dense_layer
False
add dense layer, add activation layer (relu)
True
Add final dense layer (size 4), add final activation layer (softmax)
Specify training configurations
Train model

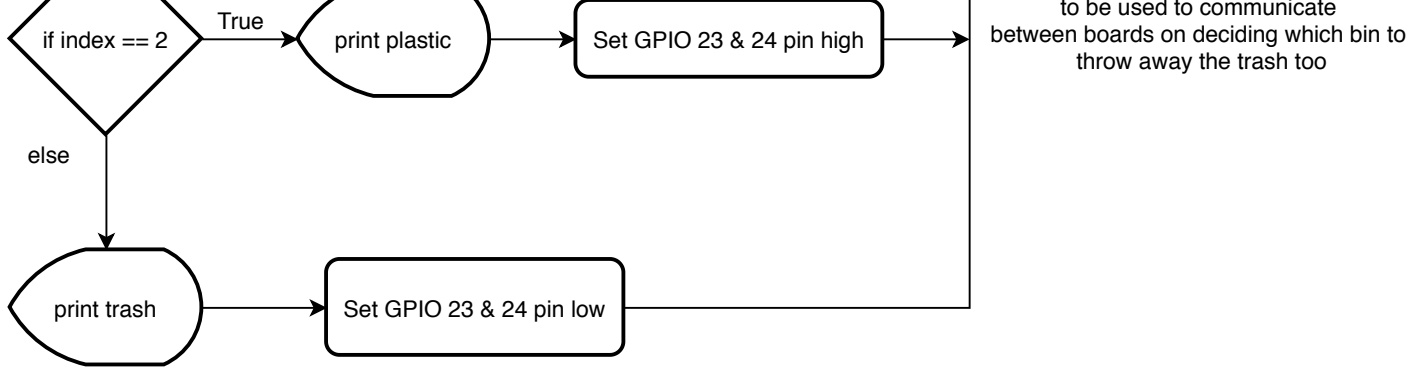


The main python script on the raspberry PI 4, which is called on from within the state machine. This simply takes a picture and sends the picture to a function to be classified using the CNN model

The photo is saved onto disk, and later opened from disk as well. With the time being undistinguished by user. Once a new photo is taken, it will overwrite the previous photo.

The python script ends, but the PI state machine enters the standby state after a picture classification is made, and the bin has thrown away the trash


RunCV(filepath).py
Python 3.7.3
OpenCV 4.1.2
Tensorflow 1.14.0
This python script is called from the main script on the Pi4. This script takes a filepath (picture) from the main file and classifies the image using the CNN model.
call and load CNN model using TensorFlow
Prepare(filepath)
Predict with photo data array
Send photo data array to CNN model to classify
print prediction results
Show probability of all classes
Index = highest probability results
using a keras function, this returns a single number. The number represents the class with the highest probability
if index == 0
True
print aluminum
Set GPIO 23 pin high
False
if index == 1
True
print glass
Set GPIO 24 pin high
False
Cleanly close all working GPIO pins
return to model_Pitest.py
The GPIO pins are what is going
Prepare(filepath)
IMG_SIZE = 200
This function simply reads the image and resizes the image. This is done to ensure the input image is the same size as the images in the CNN model
read image using cv2 and store image pixel data in an array
resize the image using opencv and store the newly sized pixel data in new_array
return new_array


if index == 2
True
print plastic
Set GPIO 23 & 24 pin high
else
print trash
Set GPIO 23 & 24 pin low
to be used to communicate
between boards on deciding which bin to
throw away the trash too