Binary Image classification using CNN model

Subject: Data 603

Professor: Chaojie Wang

Yaswanth Reddy Annapureddy(gn88933@umbc.edu)

PROBLEM STATEMENT

Solid waste management is one of the world's most severe problems. Solid waste is produced by a variety of human activities including residences, businesses, farms, and other industries, and poses serious dangers to the environment and the general public's health. When ignored, it may lead to a threat to the ecosystem. Segregation is one method of managing solid waste. Different sorts of solid trash are sorted to accomplish this. Therefore, to appropriately dispose of the garbage, it is crucial to explicitly identify its type. The problem is 'how do we manage this waste, while we can't use it anymore?'. Several nations have issues related to waste because the waste production rate is not at par with its management efforts. Many people just dispose of mixed waste in dump sites because they are unaware of the effects of garbage that has not been properly segregated before it is disposed of. Kitchen garbage, agricultural waste, and human and animal waste are all examples of biodegradable waste that can be broken down biologically by live microbes. Non-biodegradable wastes, on the other hand, are wastes that cannot be broken down biologically. Plastic, metal, glass, etc. are all included. This is the reason why managing non-biodegradable trash through various technologies and solutions has gained more significance in the modern day. To categorize waste into biodegradable and non-biodegradable types, this work offers a solution for the stated problem using machine learning models, tensor flow, and convolutional neural networks. We have taken more than 150k images of biodegradable and non-biodegradable items to train the model.

OBJECTIVE

- Our aim is to classify the waste with automation as a trained model at its base to quickly separate both so that they can be sent separately to either recycling or dump accordingly.
- We aim to regularize the classification of trash for environmental purposes.
- Cities and metropolitan areas can implement this model to collect data on how citizens are using the recycling facilities, and how to improve the service.

SOLUTION

- we have taken a total of 150K images in each category and trained a CNN model with Keras to test the accuracy and efficiency of the model.
- Used traditional deep learning techniques like tensor flow, and Keras.
- If the model is paired with an AI GPU with Open cv, we can immediately classify the trash and optimize the recycling process.
- One more advantage is that cleaning robots can quickly identify and classify waste accordingly.

DATASET

Our dataset contains a total of more than 150K images in both bio-degradable and non-biodegradable materials, sourced from the Kaggle open dataset.

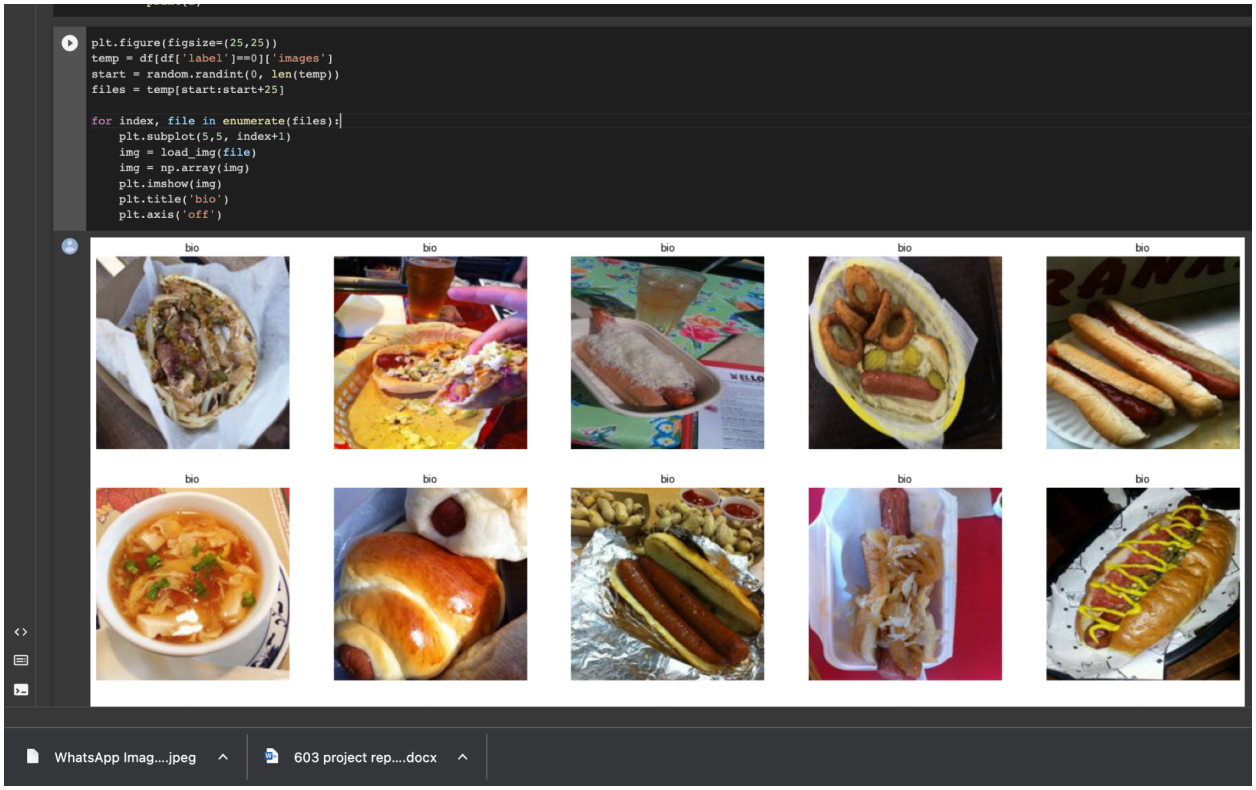


Fig 1. Sample images

Dataset parameters:

There are two parameters, namely:

1. Images

   This column contains the local paths for the images which are fed into the model to pick up images

```
df = pd.DataFrame()
df['images'] = input_path
df['label'] = label
df = df.sample(frac=1).reset_index(drop=True)
df
```

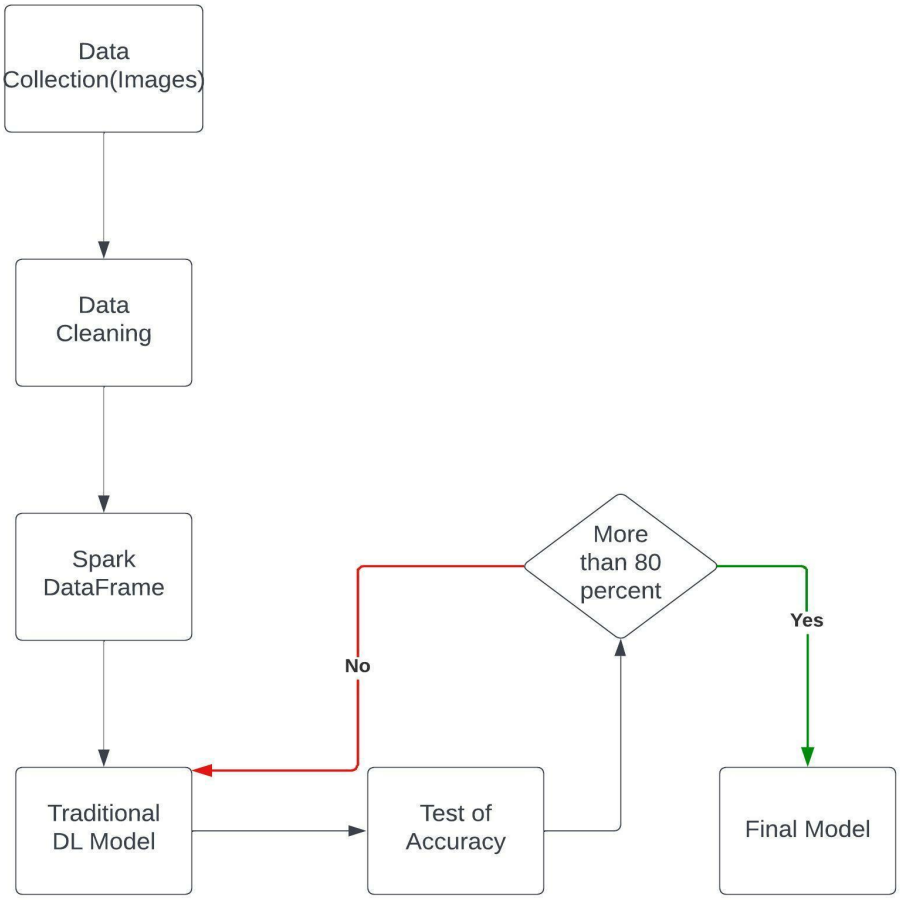
	images	label
0	PROJECT/nbio/TRAIN.1_NBIODEG_VFL_5674.jpg	1
1	PROJECT/bio/TRAIN.1_BIODEG_ORI_12050.jpg	0
2	PROJECT/bio/TRAIN.1_BIODEG_ORI_18192.jpg	0
3	PROJECT/bio/TRAIN.1_BIODEG_ORI_24019.jpg	0
4	PROJECT/bio/TRAIN.2_BIODEG_ORI_947.jpg	0
...	...	...
149800	PROJECT/bio/TRAIN.2_BIODEG_ORI_23480.jpg	0
149801	PROJECT/bio/TRAIN.2_BIODEG_ORI_24028.jpg	0
149802	PROJECT/bio/TRAIN.3_BIODEG_ORI_20956.jpg	0
149803	PROJECT/bio/TRAIN.2_BIODEG_ORI_1112.jpg	0
149804	PROJECT/nbio/TRAIN.1_NBIODEG_HFL_2149.jpg	1

149805 rows × 2 columns

Fig 2. Dataset image

2. Label
   This parameter consists of binary numbers 0 and 1. The 0 represents the biodegradable image and 1 represents the non-biodegradable image.

FRAMEWORK



Model workflow

DATA COLLECTION AND MODEL MAKING

Data Collection Process:

The study's data set consists of 150K photos of trash divided into two categories: "biodegradable" and "non-biodegradable." It is retrieved from the open-access Kaggle database. Six categories make up the dataset: trash, glass, paper, cardboard, plastic, and metal. The

non-biodegradable class used some of these images that were gathered from the six categories, and the biodegradable class used images of typical food waste.

Data Augmentation:

The performance of an algorithm improves with the amount of data it has access to. To adequately train the machine learning models, the original dataset's relatively limited number of photos needed to be augmented. Data augmentation is a way to increase the dataset's size and the model's precision. By generating fresh data from the base data, data augmentation improves data. Simply applying random changes to our images will improve the generalizability of our model. Compressions, rotations, stretches, and even color shifts are examples of these transformations. We have done this using ImageDataGenerator from tensorflow.keras.preprocessing.image.

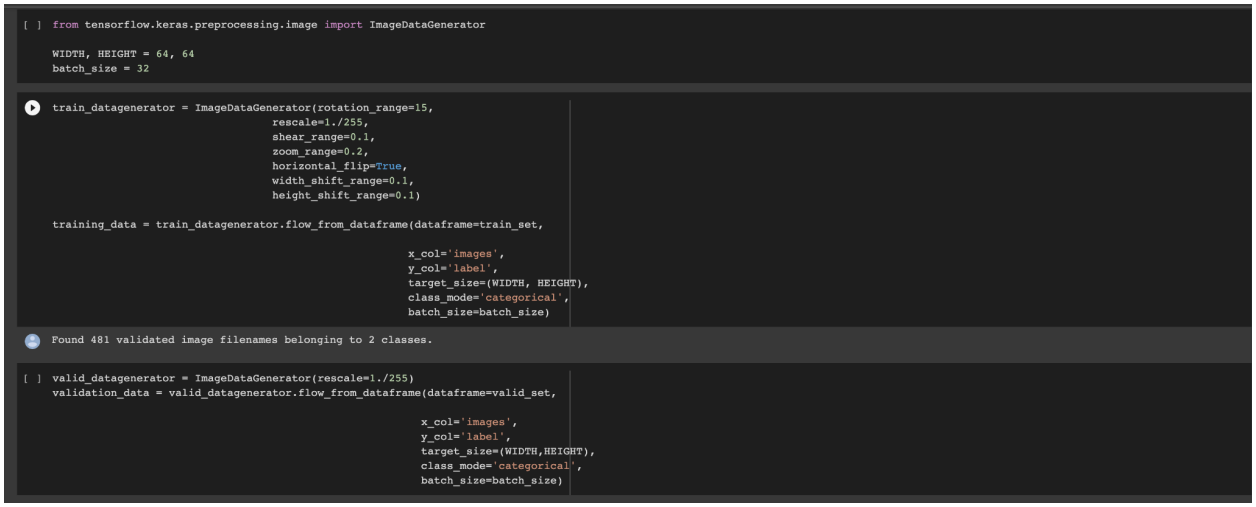


Fig 3. Image data generator

Training and Testing:

This is where the data is fit into the training set. Due to the callback's monitoring of the validation set, the validation set is also passed. If the loss does not improve after two epochs as specified in the EarlyStopping callback, the training process will be terminated by the callback. In this scenario, you can define a large number of epochs.

RESULTS

We got a total of 90 percent accuracy on the Training dataset and 95 percent accuracy on the testing dataset.

```
train set distribution:
 0    71863
1    47981
Name: label, dtype: int64

valid set distribution:
 0    18020
1    11941
Name: label, dtype: int64
```

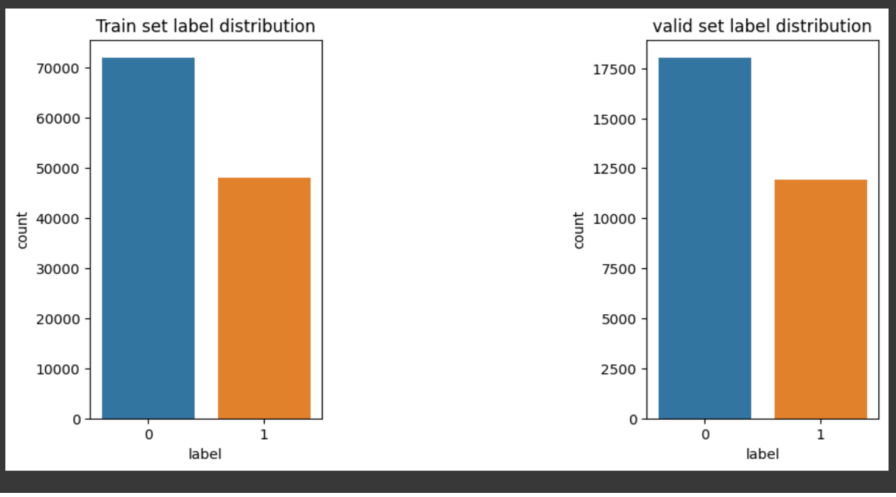


Fig 4. Testing and training dataset distribution.

```
model.summary()

Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 64, 64, 32)        2432

 max_pooling2d_9 (MaxPooling  (None, 32, 32, 32)       0
 2D)

 conv2d_16 (Conv2D)          (None, 32, 32, 64)        18496

 conv2d_17 (Conv2D)          (None, 32, 32, 64)        36928

 max_pooling2d_10 (MaxPoolin  (None, 16, 16, 64)       0
 g2D)

 conv2d_18 (Conv2D)          (None, 16, 16, 128)       73856

 conv2d_19 (Conv2D)          (None, 16, 16, 128)       147584

 max_pooling2d_11 (MaxPoolin  (None, 8, 8, 128)        0
 g2D)

 flatten_3 (Flatten)         (None, 8192)              0

 dense_9 (Dense)             (None, 64)                524352

 dropout_6 (Dropout)         (None, 64)                0

 dense_10 (Dense)            (None, 32)                2080

 dropout_7 (Dropout)         (None, 32)                0

 dense_11 (Dense)            (None, 2)                 66

=================================================================
Total params: 805,794
Trainable params: 805,794
Non-trainable params: 0
_________________________________________________________________
```

Fig 5. Custom CNN model Summary

```
history = model.fit(training_data,
                    epochs=5,
                    validation_data=validation_data,
                    validation_steps=valid_set.shape[0]//batch_size,
                    steps_per_epoch=train_set.shape[0]//batch_size
                    )
```

```
Epoch 1/5
3745/3745 [==============================] - 288s 77ms/step - loss: 0.3159 - accuracy: 0.8702 - val_loss: 0.2173 - val_accuracy: 0.9126
Epoch 2/5
3745/3745 [==============================] - 286s 76ms/step - loss: 0.2354 - accuracy: 0.9078 - val_loss: 0.2040 - val_accuracy: 0.9204
Epoch 3/5
3745/3745 [==============================] - 286s 76ms/step - loss: 0.2156 - accuracy: 0.9164 - val_loss: 0.1865 - val_accuracy: 0.9266
Epoch 4/5
3745/3745 [==============================] - 286s 76ms/step - loss: 0.2004 - accuracy: 0.9220 - val_loss: 0.1787 - val_accuracy: 0.9301
Epoch 5/5
3745/3745 [==============================] - 286s 76ms/step - loss: 0.1888 - accuracy: 0.9277 - val_loss: 0.1729 - val_accuracy: 0.9316
```

Show code

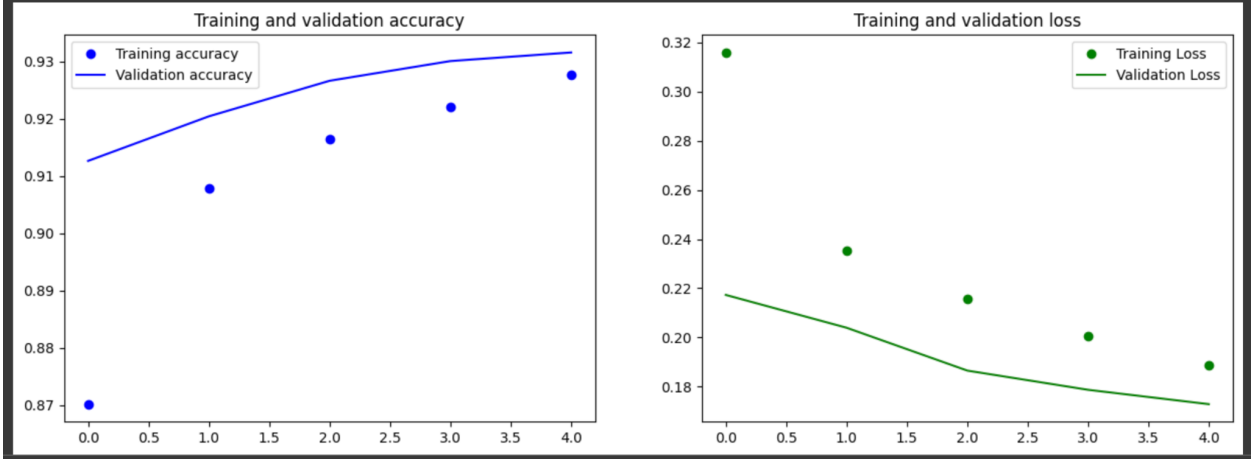


Fig 6. Custom model Epochs, Training and validation accuracy and loss.

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 64, 64, 3)]       0

 block1_conv1 (Conv2D)       (None, 64, 64, 64)        1792

 block1_conv2 (Conv2D)       (None, 64, 64, 64)        36928

 block1_pool (MaxPooling2D)  (None, 32, 32, 64)        0

 block2_conv1 (Conv2D)       (None, 32, 32, 128)       73856

 block2_conv2 (Conv2D)       (None, 32, 32, 128)       147584

 block2_pool (MaxPooling2D)  (None, 16, 16, 128)       0

 block3_conv1 (Conv2D)       (None, 16, 16, 256)       295168

 block3_conv2 (Conv2D)       (None, 16, 16, 256)       590080

 block3_conv3 (Conv2D)       (None, 16, 16, 256)       590080

 block3_pool (MaxPooling2D)  (None, 8, 8, 256)         0

 block4_conv1 (Conv2D)       (None, 8, 8, 512)         1180160

 block4_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block4_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 block5_conv1 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv2 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv3 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

fig 7. VGG16 model summary

```
Epoch 1/2
3745/3745 [==============================] - 290s 77ms/step - loss: 0.1440 - accuracy: 0.9470 - val_loss: 0.1516 - val_accuracy: 0.9395
Epoch 2/2
3745/3745 [==============================] - 286s 76ms/step - loss: 0.1068 - accuracy: 0.9613 - val_loss: 0.0968 - val_accuracy: 0.9647
```

Show code

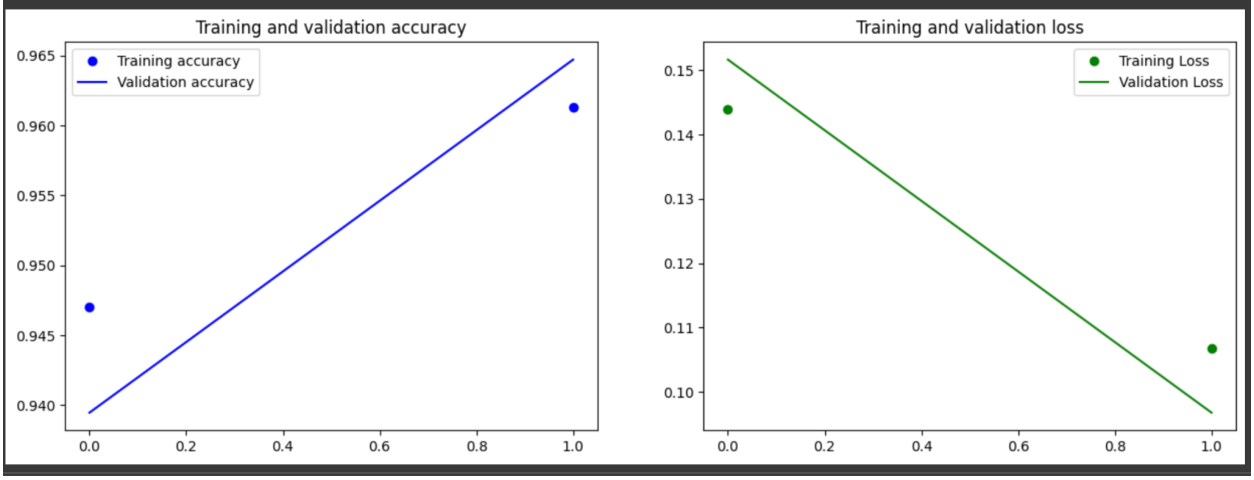


fig 8. VGG16 model Epochs, Training and validation accuracy and loss

```
fig, axs = plt.subplots(nrows=len(test_generator.filenames), figsize=(6, 1.5*len(test_generator.filenames)))
for i in range(len(test_generator.filenames)):
    filename = test_generator.filenames[i]
    prediction_Custom = prediction[i][0]
    label = 'biodegradable' if prediction_Custom >= 0.5 else 'non-biodegradable'
    img_path = os.path.join('/content/df1', filename)
    img = Image.open(img_path)
    axs[i].imshow(img)
    axs[i].axis('off')
    axs[i].set_title(f"Prediction: {label} (score: {float(prediction_Custom):.2f})")
plt.tight_layout()
plt.show()
```

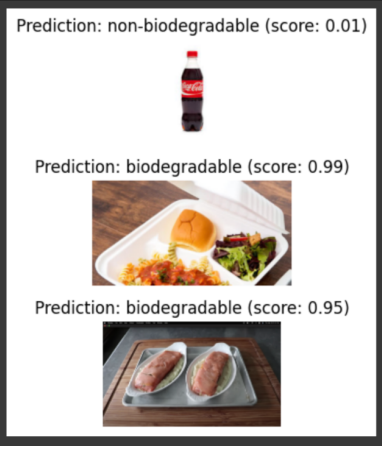


fig 9. Custom CNN model predictions

```
fig, axs = plt.subplots(nrows=len(test_generator.filenames), figsize=(6, 1.5*len(test_generator.filenames)))
for i in range(len(test_generator.filenames)):
    filename = test_generator.filenames[i]
    prediction_Custom = prediction1[i][0]
    label = 'biodegradable' if prediction_Custom >= 0.5 else 'non-biodegradable'
    img_path = os.path.join('/content/df1', filename)
    img = Image.open(img_path)
    axs[i].imshow(img)
    axs[i].axis('off')
    axs[i].set_title(f"Prediction: {label} (score: {float(prediction_Custom):.2f})")
plt.tight_layout()
plt.show()
```

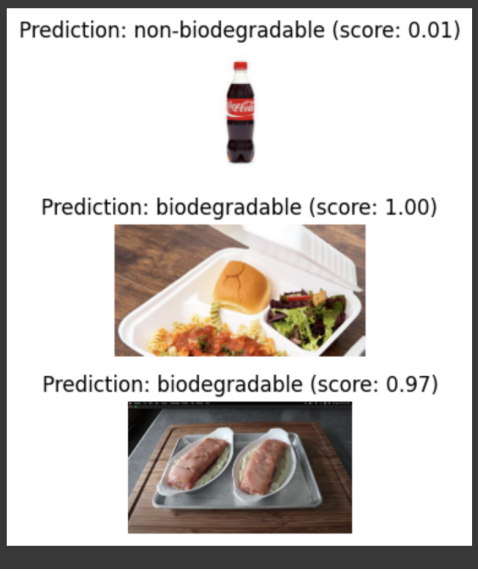


fig 10. VGG16 model predictions

IMPACT OF THE PROJECT

If the model is used in AI-powered trash collectors or classifications, we can save millions of trash from getting landfilled. If the bio and non-biodegradable materials are mixed with each other, it's hard to classify and recycle the non-degradable. So if they are classified properly, we can save tons of trash from going to landfills.

REFERENCE

1.) Hanbal, I. F., Ingosan, J. S., Oyam, N. A. A., & Hu, Y. (2020). Classifying Wastes Using Random Forests, Gaussian Naïve Bayes, Support Vector Machine and Multilayer Perceptron. *IOP Conference Series: Materials Science and Engineering*, *803*(1), 012017. https://doi.org/10.1088/1757-899x/803/1/012017

2.) Atik, I. (2022). Analysis of Biodegradable and Non-Biodegradable Materials Using Selected Deep Learning Algorithms. *International Journal of Computer (IJC)*, *43*(1),

48–59. https://ijcjournal.org/index.php/InternationalJournalOfComputer/article/view/1939/712

3.) Bhansali, R. (2021, October 11). *An Overview of Non-Biodegradable Waste Management and Solutions*. Organica Biotech. https://organicabiotech.com/an-overview-of-non-biodegradable-waste-management-and-solutions/

4.) whoisslimshady. (2021, March 14). *Introduction to Convolutional Neural Network With Tensorflow and Keras*. Geek Culture. https://medium.com/geekculture/introduction-to-convolutional-neural-network-with-tensorflow-and-keras-cb52cdc66eaf

5.) Derrick Mwiti, (2021, February 23). *How to build CNN in TensorFlow: examples, code and notebooks | cnvrg.io*. https://cnvrg.io/cnn-tensorflow/

6.) *What is Non-Biodegradable Waste: Define, advantages and disadvantages*. (2020, November 25). OkCredit Blogs - Business Ideas, Tips, Government Schemes & More. https://okcredit.in/blog/what-is-non-biodegradable-waste/

The link for the data set and git hub repo is:

1.) Data set link: Project_training set, Test data
2.) Git hub Link: Git Hub