1 MANU 465 Project: 3D Printing

1.0.1 Authors:

Group 7

Liam Bontkes, 25530163

Stacy Shang

Theophile Calloc'h

Tony Lyu

1.1 1 Project Description

The goal of this project is to create a machine learning model which is capable of identifying defective print layers and classifying 3D print layers as Pass/Fail with 85% or greater accuracy. The machine learning model will use a series of images of print layers (defective and non-defective), interpreted with a machine vision model. To reduce the scope of the project, we will only train the model on simple geometric shapes such as cubes, spheres and pyramids. Additionally, we will only be training the model on 3D prints from a fused filament fabrication printer using polylactic acid filament and a 4mm extruder head.

1.2 2 Libraries

In [1]:
```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

1.3 3 Data Preprocessing

Preprocess the training and test set.

In [2]:
```
from keras.preprocessing.image import ImageDataGenerator

image_data_generator = ImageDataGenerator(rescale=1. / 255,
                                          shear_range=0.2,
                                          zoom_range=0.2,
                                          horizontal_flip=True)

training_set = image_data_generator.flow_from_directory('Dataset/training_set',
                                                        target_size=(64, 64),
                                                        batch_size=32,
                                                        class_mode='binary')

test_set = image_data_generator.flow_from_directory('Dataset/test_set',
                                                    target_size=(64, 64),
                                                    batch_size=32,
                                                    class_mode='binary')
```

```
Found 54 images belonging to 2 classes.
Found 15 images belonging to 2 classes.
```

1.4 4 Building the CNN Model

1.4.1 4.1 Initialize the Model

In [3]:
```
model = tf.keras.models.Sequential()
```

1.4.2 4.2 Add Convolutional Layers

In [4]:
```
# add and pool 1st layer
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', ir
model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=1))
```

In [5]:
```
# add and pool 2nd layer
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=1))
```

1.4.3 4.3 Flatten the Model

In [6]:
```
model.add(tf.keras.layers.Flatten())
```

1.4.4 4.4 Add Connection Layer

In [7]:
```
model.add(tf.keras.layers.Dense(units=256, activation='relu'))
```

1.4.5 4.5 Add Output Layer

In [8]:
```
model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

1.4.6 4.6 Compile the Model

In [9]:
```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

1.5 5 Train the CNN Model

In [11]:
```
model.fit(x=training_set, validation_data=test_set, epochs=50)
```

```
Epoch 45/50
2/2 [==============================] - 2s 1s/step - loss: 5.1523e-06 - accura
cy: 1.0000 - val_loss: 2.5886e-05 - val_accuracy: 1.0000
Epoch 46/50
2/2 [==============================] - 2s 1s/step - loss: 5.7465e-06 - accura
cy: 1.0000 - val_loss: 1.8941e-05 - val_accuracy: 1.0000
Epoch 47/50
2/2 [==============================] - 2s 1s/step - loss: 5.8345e-06 - accura
cy: 1.0000 - val_loss: 2.0567e-05 - val_accuracy: 1.0000
Epoch 48/50
2/2 [==============================] - 2s 1s/step - loss: 4.9002e-06 - accura
cy: 1.0000 - val_loss: 3.0487e-05 - val_accuracy: 1.0000
Epoch 49/50
2/2 [==============================] - 2s 1s/step - loss: 5.4936e-06 - accura
cy: 1.0000 - val_loss: 2.4372e-05 - val_accuracy: 1.0000
Epoch 50/50
2/2 [==============================] - 2s 1s/step - loss: 6.9247e-06 - accura
cy: 1.0000 - val_loss: 1.9716e-05 - val_accuracy: 1.0000
```

Out[11]:
```
<keras.callbacks.History at 0x2336de60ee0>
```

1.6 6 Make a Prediction

In [13]:
```
from keras.preprocessing import image

prediction_image = image.load_img('Dataset/Single_Prediction/Print_1.jpg', target
prediction_image = image.img_to_array(prediction_image)
prediction_image = np.expand_dims(prediction_image, axis=0)

result = model.predict(prediction_image)

if result[0][0] == 1:
    prediction = 'Part is OK.'
else:
    prediction = 'Part is defective!'
print(prediction)
```

```
Part is defective!
```

In [ ]: