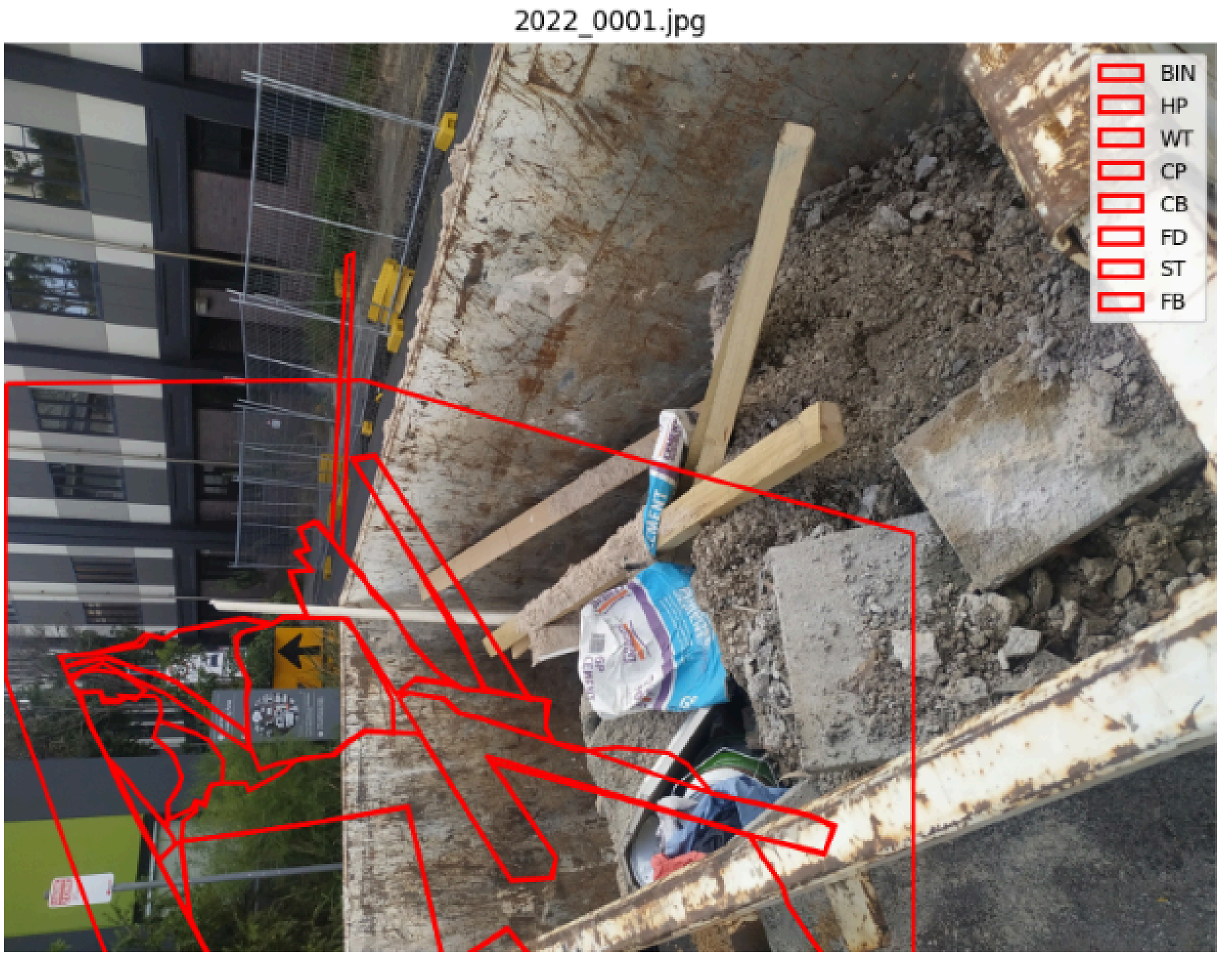
Student Name:	Rabia Riaz
Student ID:	F2023332078
Course Title:	Introduction to Data Science - D2
Assignment Title:	Report on Project: “Deep Learning for Waste Classification in Cluttered Environments”
Date of Submission:	14th July, 2025
Submitted to:	Usama Ahmed
Grade:	
Feedback:	

Introduction

The project is concerned with the application of deep learning to classify and tag various kinds of construction and demolition waste in images. The aim was to develop a model that would be able to look at an image of waste and label each pixel with the appropriate category, wood, plastic, fabric or food waste. The data source of the current project is the scientific paper that was published on the website of Nature in 2025. It also has numerous quality images of actual waste, and annotations that inform us about the location of every kind of waste in the picture.

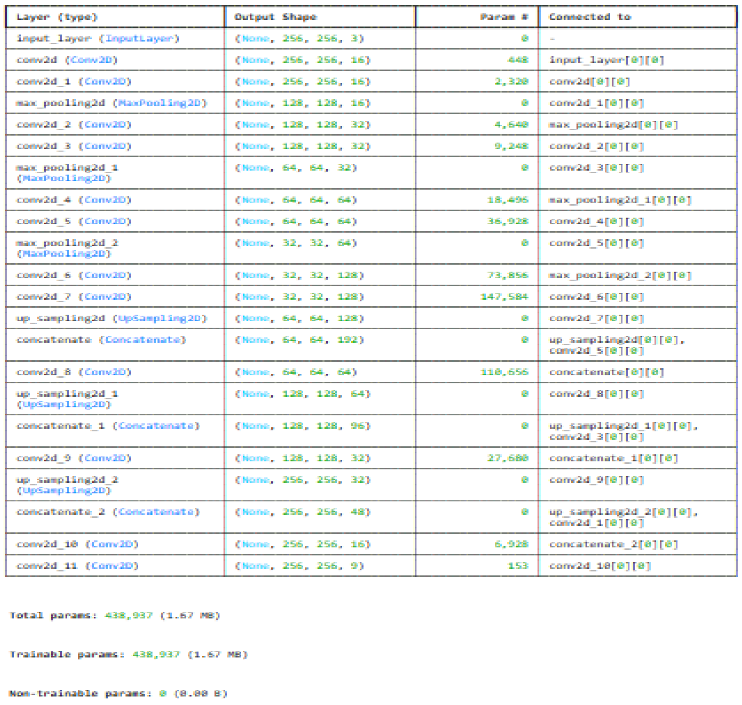


Dataset & Annotations

With every image, there was a JSON file that contained labeled polygons of the objects in the picture. We applied these JSON files to generate mask images, indicating the right class of every pixel. The waste items in the dataset are wastes bins, hard plastics, wood, cardboard, concrete, food wastes, stones and fabrics.

Model Architecture

In order to complete this project we constructed a U-Net model with tensorflow and Keras. U-Net is a unique form of neural network and is effective in image segmentation. The input images to our model are 256 by 256 with 3 color channels and its output is a prediction mask of the same size. It has multiple layers which shrink the image followed by an expansion of the image so that it can learn general shapes as well as minute details.

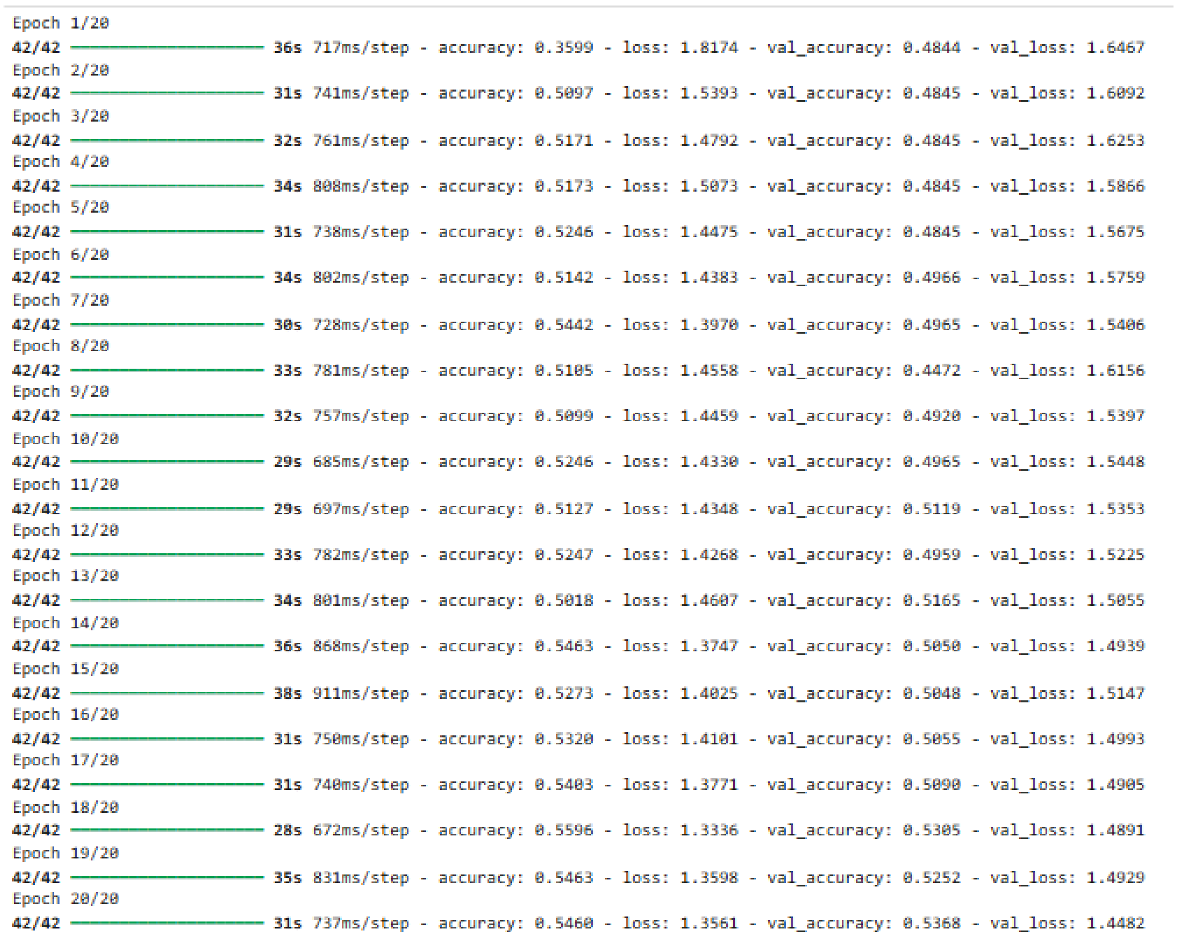


Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 256, 256, 3)	0	-
conv2d (Conv2D)	(None, 256, 256, 16)	448	input_layer[0][0]
conv2d_1 (Conv2D)	(None, 256, 256, 16)	2,320	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 128, 128, 16)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 128, 128, 32)	4,640	max_pooling2d[0][0]
conv2d_3 (Conv2D)	(None, 128, 128, 32)	9,248	conv2d_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 32)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None, 64, 64, 64)	18,496	max_pooling2d_1[0][0]
conv2d_5 (Conv2D)	(None, 64, 64, 64)	36,928	conv2d_4[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 64)	0	conv2d_5[0][0]
conv2d_6 (Conv2D)	(None, 32, 32, 128)	73,856	max_pooling2d_2[0][0]
conv2d_7 (Conv2D)	(None, 32, 32, 128)	147,584	conv2d_6[0][0]
up_sampling2d (UpSampling2D)	(None, 64, 64, 128)	0	conv2d_7[0][0]
concatenate (Concatenate)	(None, 64, 64, 192)	0	up_sampling2d[0][0], conv2d_5[0][0]
conv2d_8 (Conv2D)	(None, 64, 64, 64)	110,656	concatenate[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 128, 128, 64)	0	conv2d_8[0][0]
concatenate_1 (Concatenate)	(None, 128, 128, 96)	0	up_sampling2d_1[0][0], conv2d_3[0][0]
conv2d_9 (Conv2D)	(None, 128, 128, 32)	27,680	concatenate_1[0][0]
up_sampling2d_2 (UpSampling2D)	(None, 256, 256, 32)	0	conv2d_9[0][0]
concatenate_2 (Concatenate)	(None, 256, 256, 48)	0	up_sampling2d_2[0][0], conv2d_1[0][0]
conv2d_10 (Conv2D)	(None, 256, 256, 16)	6,928	concatenate_2[0][0]
conv2d_11 (Conv2D)	(None, 256, 256, 9)	153	conv2d_10[0][0]

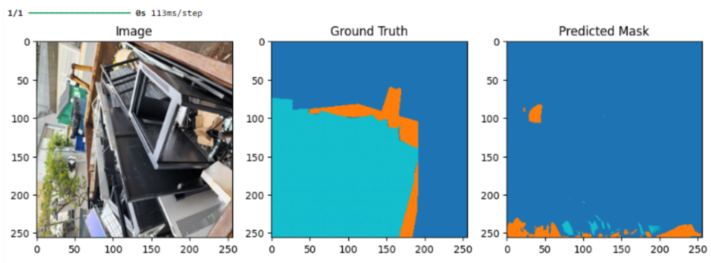
```
Total params: 438,937 (1.67 MB)

Trainable params: 438,937 (1.67 MB)

Non-trainable params: 0 (0.00 B)
```

Training Process

We used 20 rounds also known as the epochs to train the model. In training we applied 80 percent of the images to learning and 20 percent to testing. The model was capable of achieving a validation accuracy of approximately 53.7 percent and a validation loss of approximately 1.45. Another metric that we also computed was the so-called mean IoU, and this value turned out to be approximately 14.9 percent. This implies that the model has partial knowledge on the task at hand, but it can be better.



```
Epoch 1/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 36s 717ms/step - accuracy: 0.3599 - loss: 1.8174 - val_accuracy: 0.4844 - val_loss: 1.6467
Epoch 2/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 31s 741ms/step - accuracy: 0.5097 - loss: 1.5393 - val_accuracy: 0.4845 - val_loss: 1.6092
Epoch 3/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 32s 761ms/step - accuracy: 0.5171 - loss: 1.4792 - val_accuracy: 0.4845 - val_loss: 1.6253
Epoch 4/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 34s 808ms/step - accuracy: 0.5173 - loss: 1.5073 - val_accuracy: 0.4845 - val_loss: 1.5866
Epoch 5/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 31s 738ms/step - accuracy: 0.5246 - loss: 1.4475 - val_accuracy: 0.4845 - val_loss: 1.5675
Epoch 6/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 34s 802ms/step - accuracy: 0.5142 - loss: 1.4383 - val_accuracy: 0.4966 - val_loss: 1.5759
Epoch 7/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 30s 728ms/step - accuracy: 0.5442 - loss: 1.3970 - val_accuracy: 0.4965 - val_loss: 1.5406
Epoch 8/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 33s 781ms/step - accuracy: 0.5105 - loss: 1.4558 - val_accuracy: 0.4472 - val_loss: 1.6156
Epoch 9/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 32s 757ms/step - accuracy: 0.5099 - loss: 1.4459 - val_accuracy: 0.4920 - val_loss: 1.5397
Epoch 10/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 29s 685ms/step - accuracy: 0.5246 - loss: 1.4330 - val_accuracy: 0.4965 - val_loss: 1.5448
Epoch 11/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 29s 697ms/step - accuracy: 0.5127 - loss: 1.4348 - val_accuracy: 0.5119 - val_loss: 1.5353
Epoch 12/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 33s 782ms/step - accuracy: 0.5247 - loss: 1.4268 - val_accuracy: 0.4959 - val_loss: 1.5225
Epoch 13/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 34s 801ms/step - accuracy: 0.5018 - loss: 1.4607 - val_accuracy: 0.5165 - val_loss: 1.5055
Epoch 14/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 36s 868ms/step - accuracy: 0.5463 - loss: 1.3747 - val_accuracy: 0.5050 - val_loss: 1.4939
Epoch 15/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 38s 911ms/step - accuracy: 0.5273 - loss: 1.4025 - val_accuracy: 0.5048 - val_loss: 1.5147
Epoch 16/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 31s 750ms/step - accuracy: 0.5320 - loss: 1.4101 - val_accuracy: 0.5055 - val_loss: 1.4993
Epoch 17/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 31s 740ms/step - accuracy: 0.5403 - loss: 1.3771 - val_accuracy: 0.5090 - val_loss: 1.4905
Epoch 18/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 28s 672ms/step - accuracy: 0.5596 - loss: 1.3336 - val_accuracy: 0.5305 - val_loss: 1.4891
Epoch 19/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 35s 831ms/step - accuracy: 0.5463 - loss: 1.3598 - val_accuracy: 0.5252 - val_loss: 1.4929
Epoch 20/20
42/42 ━━━━━━━━━━━━━━━━━━━━ 31s 737ms/step - accuracy: 0.5460 - loss: 1.3561 - val_accuracy: 0.5368 - val_loss: 1.4482
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
```

Results and Observations

The model worked well with large objects that were well visible but failed to work with small, overlapping, or unclear objects. The unbalanced nature of the number of different classes in the dataset was one of the challenges. There were some forms of waste that were very prevalent compared to others. The other issue was the proper installation of TensorFlow, and we resolved it with the help of the Anaconda environment.

Future Improvements

In order to enhance this project in the future, we can augment the training data to generate additional training data and experiment with superior models such as utilizing ResNet as the encoder and better loss functions such as Dice Loss. Although the results are not ideal, we managed to develop a functional segmentation system that will be helpful in the sorting of waste materials with the help of artificial intelligence.

Final Output

We also saved the trained model in a .keras file, the predicted masks in images and also the IoU scores to be evaluated. The project shows that deep learning is beneficial in environmental activities such as sorting waste and holds much potential to be improved.

GitHub Repo

https://github.com/Rabia4111/IDS-FP

Annotations and dataset files are under the Files folder in the root directory.