RealWaste Image Classification with Transfer Learning

This code implements the InceptionV3 model for classifying images using transfer learning and fine-tuning with Keras.

Objective:

The objective of this project is to replicate the results obtained by pre-trained CNN models like InceptionV3 using transfer learning.

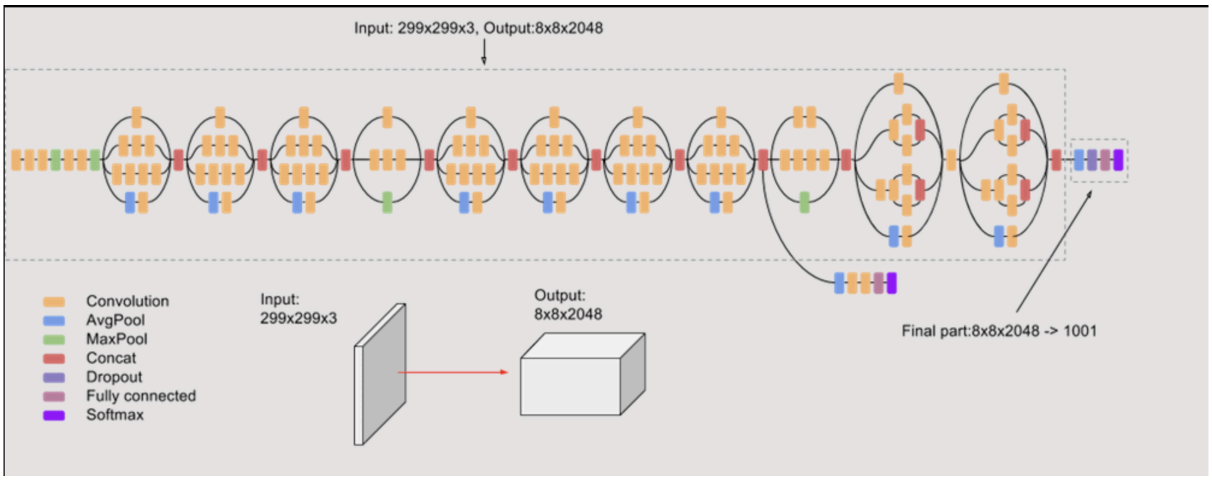


Fig 1. InceptionV3 Architecture

Data Preparation:

- The dataset is divided into ‘train’ and ‘test’ dataset.
- The dataset contains 4752 images belonging to different categories including paper, plastic, vegetation, metal, etc.
- image_size=(524,524): Resizes images to 524x524 pixels.
- shuffle=True: Randomises the order of images in the ‘train’ and ‘test’ datasets.
- seed=123: Fixes the images that will be in the two datasets.
- validation_split=0.2: Divides the original dataset into ‘train’ and ‘test’ in the ratio 80:20.

InceptionV3

- InceptionV3 model is build on the idea that an N*N convolution into a combination of much smaller convolutions of 3*1 and 1*3 etc.

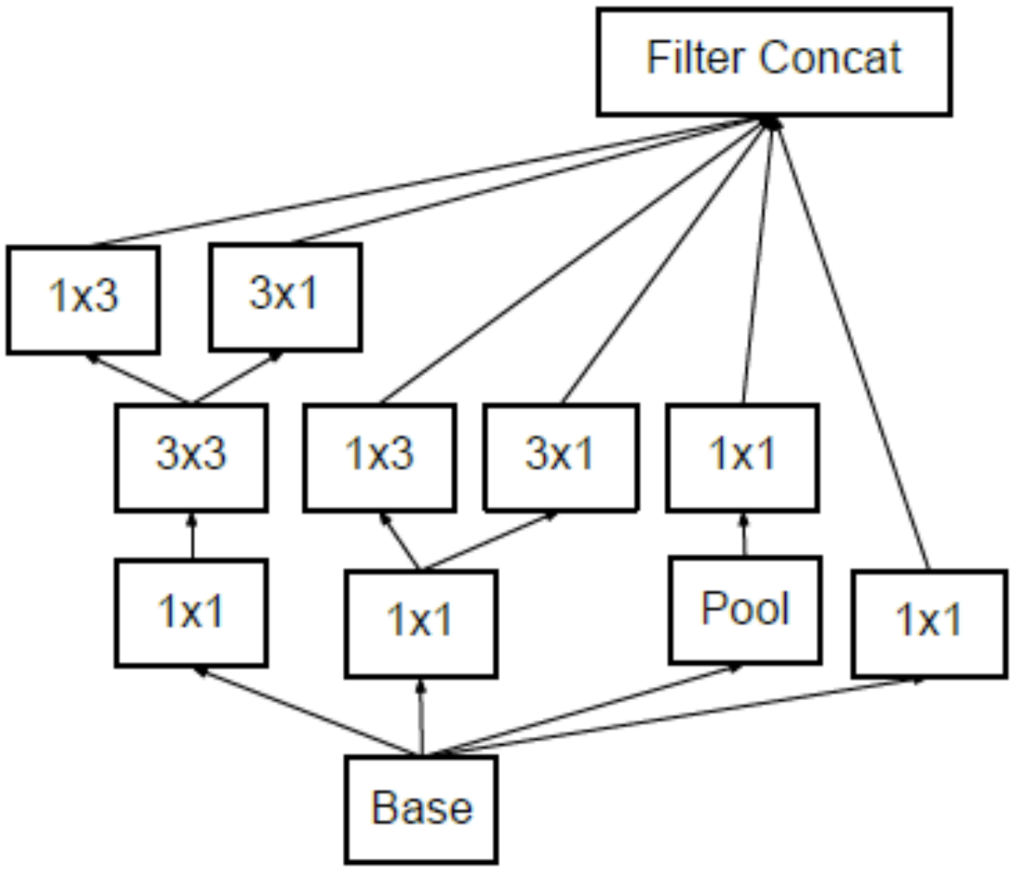


Fig 2. Structure of asymmetric convolutions

- The model is 48 layers deep.
- include_top=False returns a model without the last few fully-connected layers which makes it easier for us to add our own custom fully connected layer.
- weights='imagenet' loads the weights and biases that were used while training the Imagenet dataset. This helps to extract features from the images which are then treated through a fully connected layer.
- The InceptionV3 model uses different types of layers such as Convolution Layer, BatchNormalization, Activation, Maxpooling2D, AvgPooling2D, Mixed layers, etc.

Workflow:

- Our model takes an image of size (1, 524, 524, 3) and rescales it down to range of (-1, 1), as it is required for the imported model as stated in the paper.
- The **Sequential()** function connects the output from the previous layer to the input of the next layer.
- An image can be considered as a 2D array of integers with three channels for Red, Green and Blue.
- The Convolution layer works with a filter/kernel.
- The filters help in feature extraction like edge detection. They do so by taking the average of the dot product with a particular region of the image. This value represents the feature of that particular region in another matrix. The filter move with a stride.
- The activation layer applies a ReLU activation to the input layer.
- The MaxPool2D layer takes the maximum value from the 2D pool of the specific region in the matrix.
- The BatchNorm layer normalizes the values of the array as by subtracting the mean from each index value and then dividing the output with the standard deviation.

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

- In the InceptionV3 model the complex convolutions are broken into simpler convolutions which are then later concatenated into one single output using the mixed layer. **This helps to save a lot of compute.**
- The convolutional layer is then converted into 1D using the Flatten() layer.
- This layer is connected to the Dense layer with 512 nodes followed by ReLU activation.
- Dropout layer is used to regularise the model to prevent overfitting. A dropout of 0.3 is applied which means that 30% of the nodes become inactive at the

time of training so that a particular model does not get more importance than the other.

- Finally the Dense layer is connected with another dense layer containing 9 nodes to classify the given image to one of the classes. Softmax activation is used to convert the output in the form of probabilities.
- The index of the outputs are mapped with the names of the classes.

Compilation:

- The model is then compiled with:
    1. **‘sparse_categorical_crossentropy’**: Suitable for multi-class classification with integer labels.
    2. Adam optimizer with a low learning rate to fine-tune the pre-trained layers. The Adam optimizer takes into consideration both the momentum and adaptive learning rate while updating the weights and biases.
    3. **metrics=[‘Accuracy’]**: Monitors training and validation accuracy.
- The model is trained using the fit function with:
    - **train** dataset for training.
    - **batch_size=32**: Processes images in batches of 32 during training.
    - **validation_data=(test)**: Evaluates the model on the validation set after each epoch.

Training Performance Visualization

- The code plots the accuracy and loss curves using the data of trained model.
- These plots help to visualise how the model performs during the training phase and helps to identify overfitting and underfitting issues.

Experiment No. 1 - On-Shelf Training

Results:

- The model achieves 80.63% in on-shelf transfer learning.

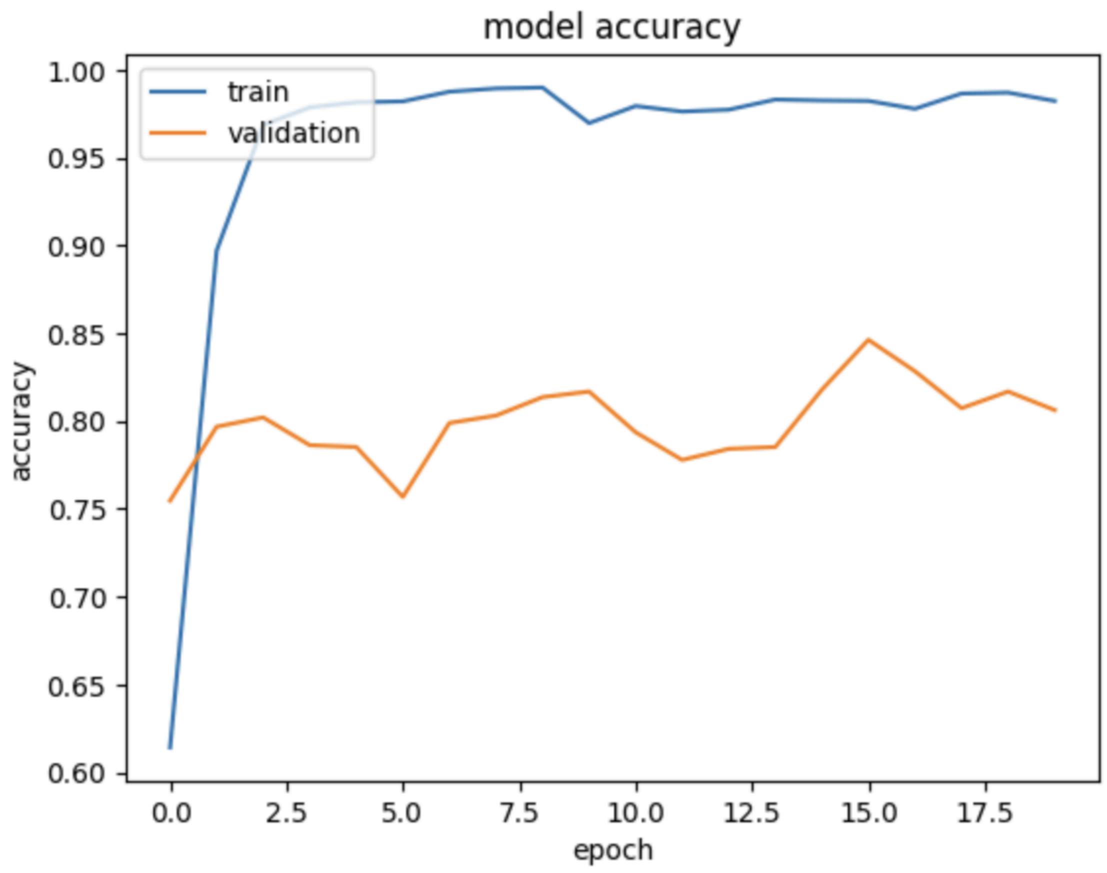


Fig 3. Model accuracy curve for on-shelf training

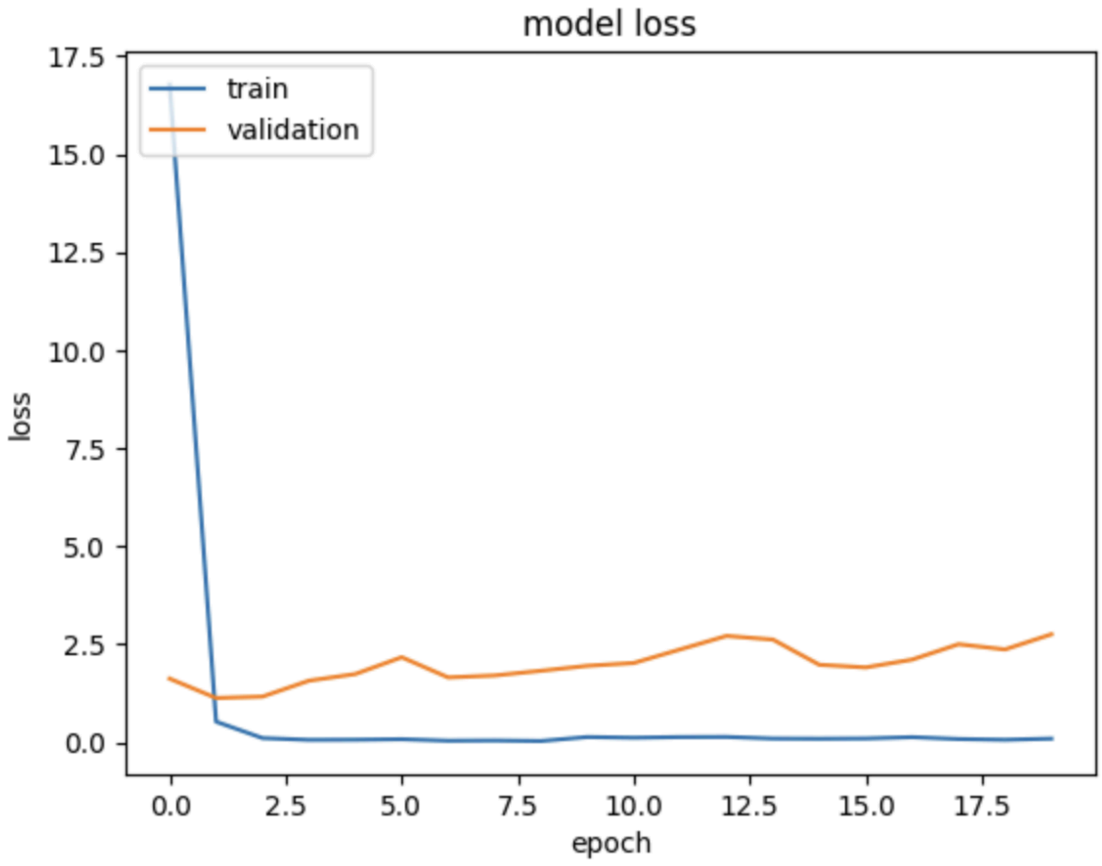


Fig 4. Loss curve for On-shelf training

Confusion Matrix

- The code iterates over the test data to collect predicted labels for each image.
- The confusion matrix visualizes the distribution of correct and incorrect predictions for each waste category.
- This helps identify classes that the model struggles with and allows for further analysis or targeted improvements.
- The confusion matrix helps to calculate the precision and recall values for each class and ultimately the f1 score. The f1 score indicates the performance of the model better than the validation accuracy. The f1 score is effective when the dataset is skewed i.e. the no. of training examples are not uniform for each class.

- **True Positive(TP)**: It means the actual value and also the predicted values are the same. In our case, the actual value is also an apple, and the model prediction is also an apple. If you observe the TP cell, the positive value is the same for Actual and predicted.
- **False Negative(FN)**: This means the actual value is positive. In our case, it is apple, but the model has predicted it as negative, i.e., grapes. So the model has given the wrong prediction. It was supposed to give a positive (apple), but it has given a negative (grape). So whatever the negative output we got is false; hence the name False Negative.
- **False Positive(FP)**: This means the actual value is negative. In our case, it is grapes, but the model has predicted it as positive, i.e., apple. So the model has given the wrong prediction. It was supposed to give a negative (grape), but it has given a positive (apple), so whatever the positive output we got is false, hence the name False Positive.
- **True Negative(TN)**: It means the actual value and also the predicted values are the same. In our case, the actual values are grapes, and the prediction is also Grapes.
- **Precision**: Precision (positive predicted) is the fraction of relevant data among the retrieved data.
  Precision(P) = TP/(TP + FP).
- **Recall**: Recall (sensitivity) is the fraction of relevant data that were retrieved. Both precision and recall are based on relevance.
  Recall(R) = TP/(TP + FN).
- **F1-score**: F1-score is a harmonic mean of Precision and Recall, and so it gives a combined idea about these two metrics. It is maximum when Precision is equal to Recall.
  F1-score = 2*P*R/(P+R).

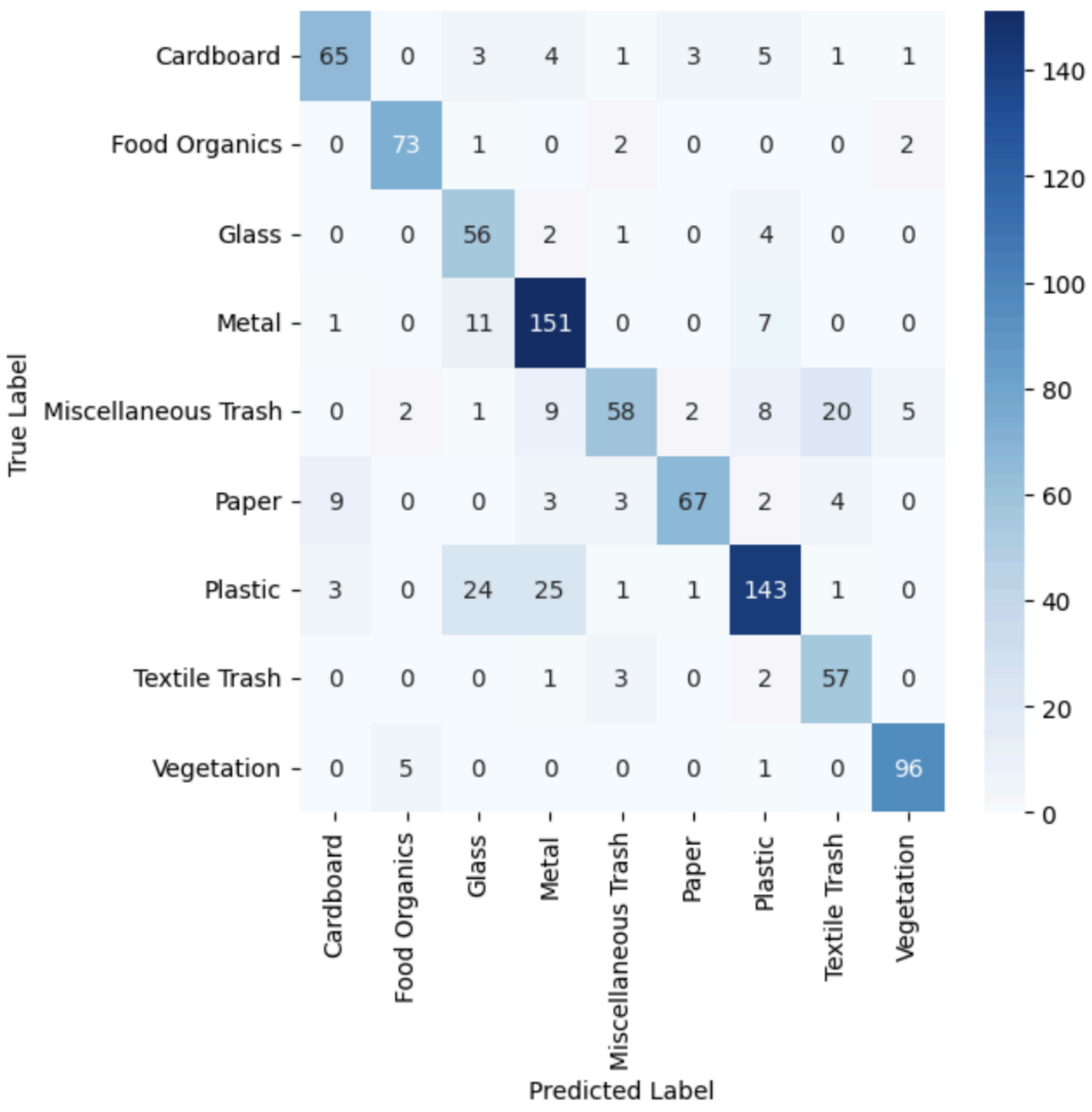


Figure 5. Confusion Matrix for On-shelf training

Classes	Precision	Recall	F1 score
Cardboard	0.83	0.78	0.83
Food Organics	0.91	0.94	0.92
Glass	0.58	0.89	0.70
Metal	0.77	0.89	0.83
Miscellaneous Trash	0.84	0.55	0.67
Paper	0.92	0.76	0.83
Plastic	0.83	0.72	0.77
Textile Trash	0.69	0.90	0.78
Vegetation	0.92	0.94	0.93
Average	0.81	0.82	0.81

Experiment No. 2 - Fine-Tuning

Results:

- Further, the accuracy increases to 87.79% using fine-tuning.

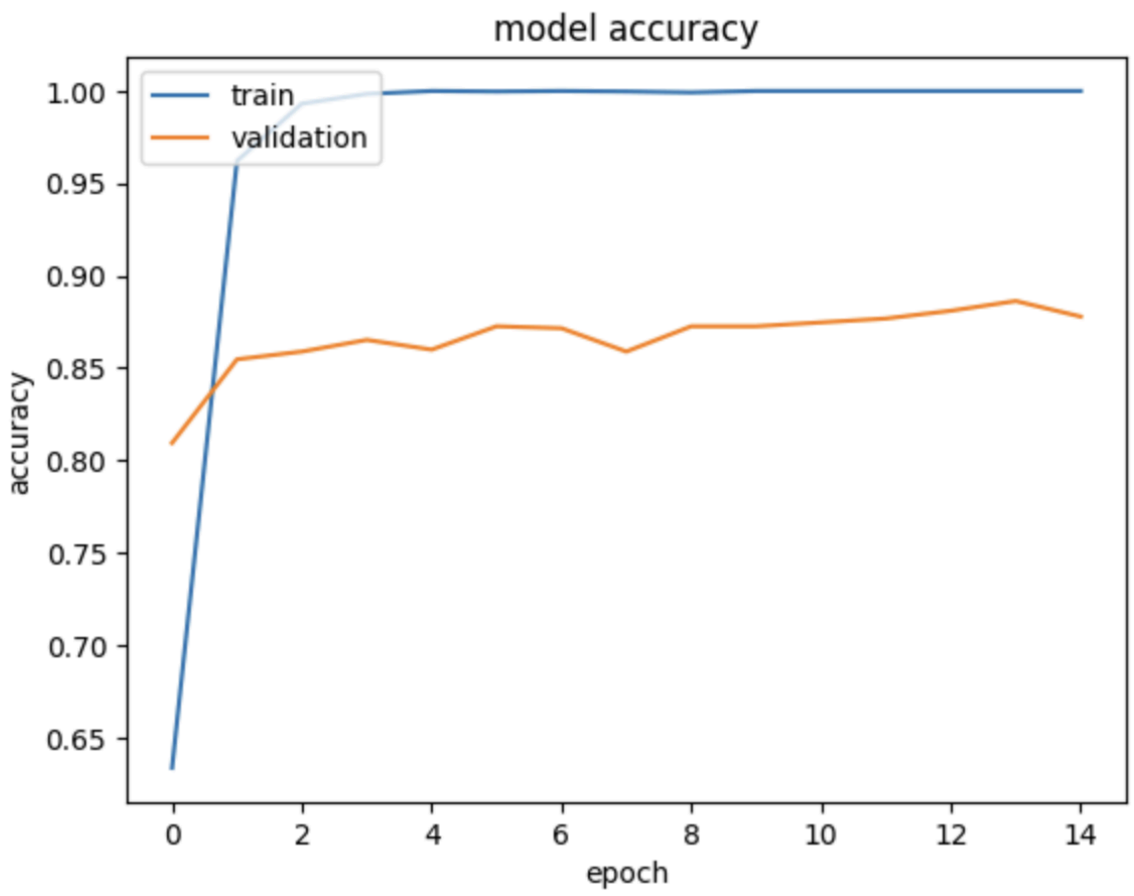


Fig 6. Model accuracy curve for fine-tuned training

- The model achieves training accuracy of 100% within 5 epochs with a learning rate of 2e-4.
- The validation accuracy increases gradually throughout the training from 80.95% to 87.79%.

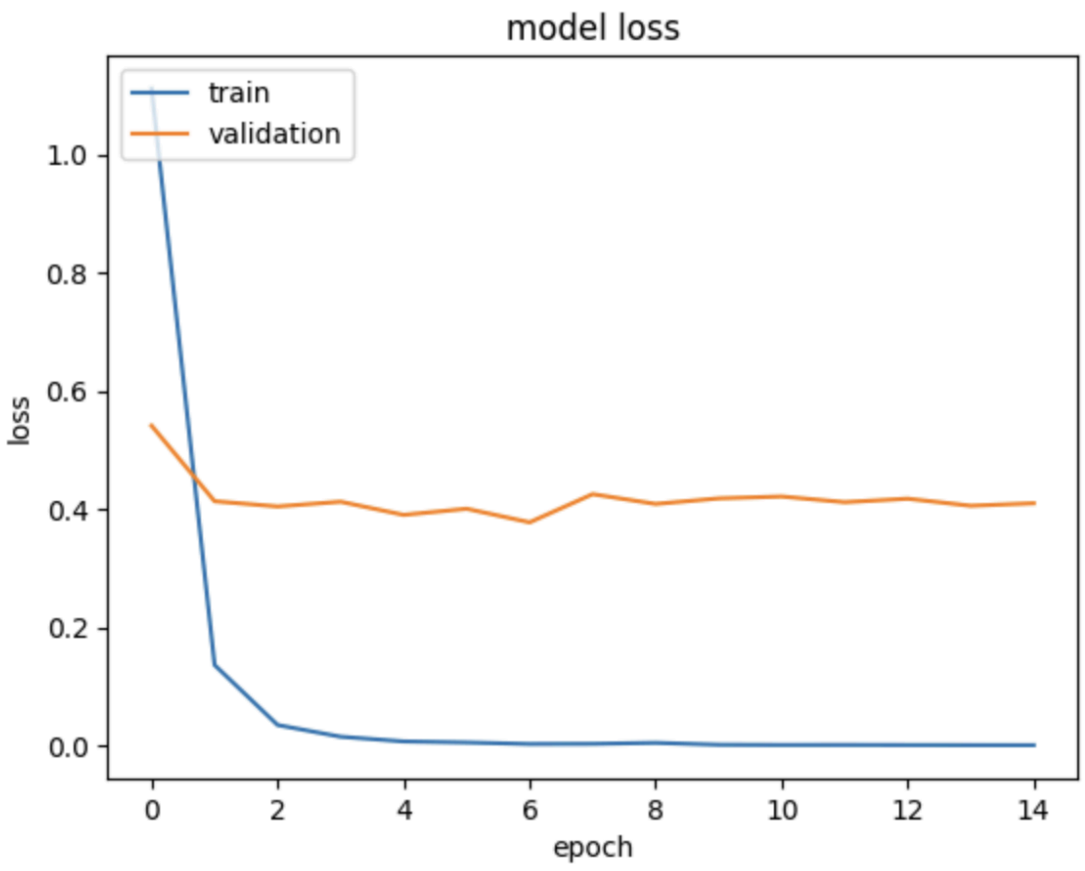
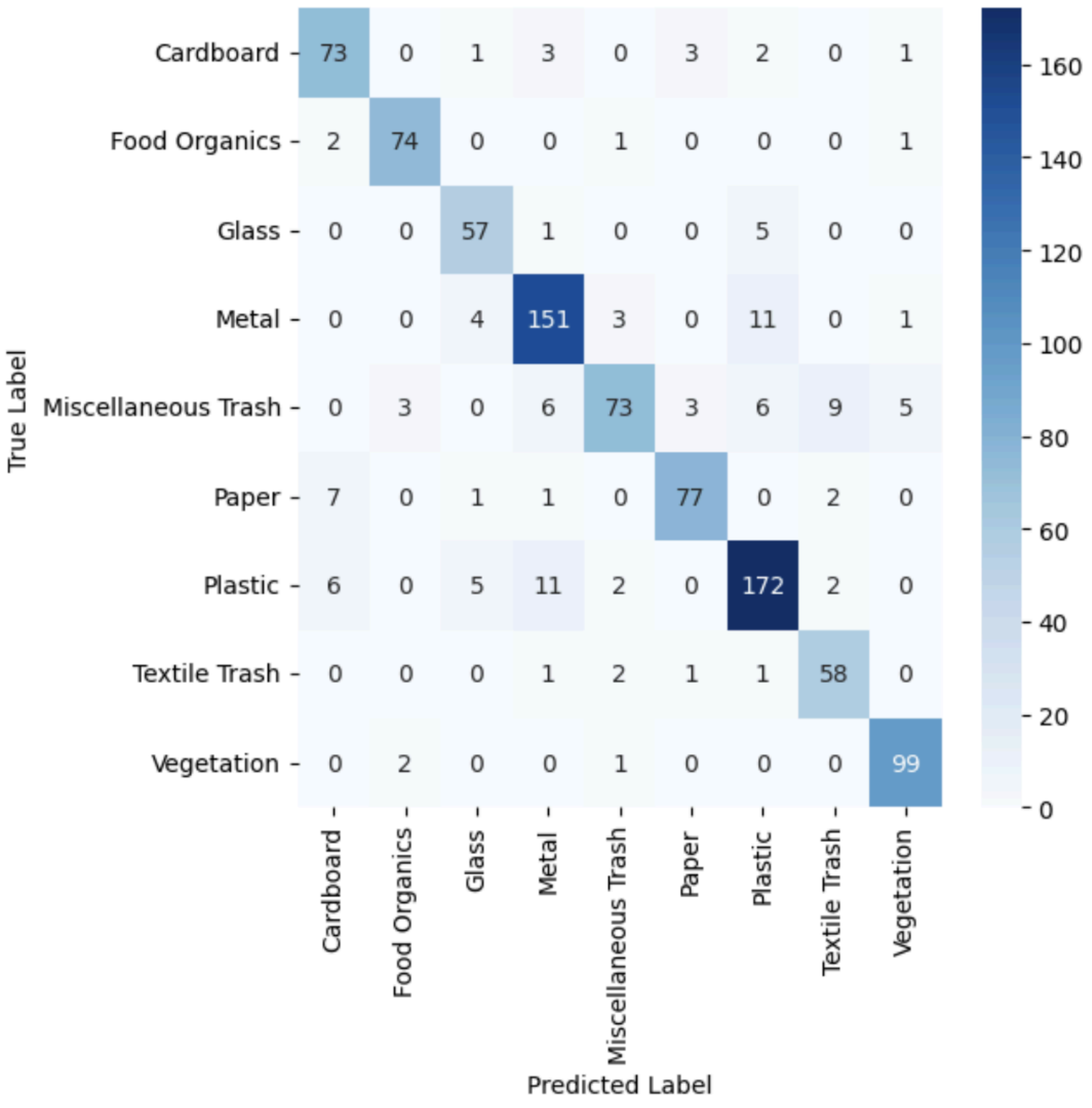


Fig 7. Loss curve for fine-tuned training

- The validation loss remains almost constant at around 0.4 while the training loss decreases to as low as 4.83e-4.

Confusion Matrix



True Label \ Predicted Label	Cardboard	Food Organics	Glass	Metal	Miscellaneous Trash	Paper	Plastic	Textile Trash	Vegetation
Cardboard	73	0	1	3	0	3	2	0	1
Food Organics	2	74	0	0	1	0	0	0	1
Glass	0	0	57	1	0	0	5	0	0
Metal	0	0	4	151	3	0	11	0	1
Miscellaneous Trash	0	3	0	6	73	3	6	9	5
Paper	7	0	1	1	0	77	0	2	0
Plastic	6	0	5	11	2	0	172	2	0
Textile Trash	0	0	0	1	2	1	1	58	0
Vegetation	0	2	0	0	1	0	0	0	99

Figure 8. Confusion Matrix for fine-tuning

Classes	Precision	Recall	F1 score
Cardboard	0.83	0.88	0.85
Food Organics	0.94	0.95	0.94
Glass	0.84	0.90	0.87
Metal	0.87	0.89	0.88
Miscellaneous Trash	0.89	0.70	0.78
Paper	0.92	0.88	0.90
Plastic	0.87	0.87	0.87
Textile Trash	0.82	0.92	0.87
Vegetation	0.83	0.97	0.95
Average	0.88	0.88	0.88

Overall

This code demonstrates a common approach for real-waste image classification using transfer learning and fine-tuning. It leverages a pre-trained InceptionV3 model and fine-tunes its final layers for the specific waste classification task. The code also includes data preparation, model compilation, training, performance visualization, and evaluation using a confusion matrix.