Department of Computer Science and Engineering

CSE 432 – Machine Learning Lab

Final Project Report

Trash Classification using Xception with Fine-Tuning

Submitted to

Mrinmoy Biswas Akash
Lecturer
Department of Computer Science & Engineering

Submitted by

Tahmid Nasif Tamal

Id: 2114951008

Session: Spring 25

Section : 7C1

Submission Date: 07/06/2025

Trash Classification using Xception with Fine-Tuning

Introduction: Garbage segregation is one of the key challenges in environmental management. This project aims to solve that problem using image classification powered by transfer learning. By using a pretrained CNN model (Xception), the system can classify images of trash into specific categories with good accuracy, even with a relatively small dataset.
Github Link: https://github.com/tahmidnasiftamal/trash-classification-xception

Dataset Description:

1. **Dataset Source:**
   https://www.kaggle.com/datasets/asdasdasasdas/garbage-classification
2. **Total Classes:** 6
   Cardboard
   Glass
   Metal
   Paper
   Plastic
   Trash (general/organic/miscellaneous)
3. **Format:**
   -Folder structure with images in subfolders named after the class.
   -Moderate-size dataset, suitable for fast prototyping.
4. **Preprocessing:**
   -All images resized to 150x150 pixels.
   -Image normalization (/255 scaling).
   -Augmentation applied (rotation, zoom, flips, etc.) to reduce overfitting.

Methodology:

1.Model Used: Xception

- A 71-layer deep CNN with depth wise separable convolutions.
- Pretrained on ImageNet.
- Known for high performance with low computation.

2.Transfer Learning Strategy:

Phase 1: Feature Extraction

- Xception base was frozen.
- Custom classification layers (GAP + Dense + Dropout + Dense with softmax) added.
- Trained for 10 epochs using Adam optimizer.

Phase 2: Fine-Tuning

- Unfroze the last 20 layers of Xception.
- Recompiled with lower learning rate (1e-5).
- Trained for 5 more epochs to allow slight adjustment of pretrained weights.

Code:

```
# Data Augmentation
datagen = ImageDataGenerator(
  rescale=1./255,
  validation_split=0.2,
  rotation_range=20,
  zoom_range=0.2,
  width_shift_range=0.2,
  height_shift_range=0.2,
  shear_range=0.2,
  horizontal_flip=True,
  fill_mode='nearest'
)

train_generator = datagen.flow_from_directory(
  dataset_path,
  target_size=IMG_SIZE,
  batch_size=BATCH_SIZE,
  class_mode='categorical',
  subset='training',
  shuffle=True
)

val_generator = datagen.flow_from_directory(
  dataset_path,
  target_size=IMG_SIZE,
  batch_size=BATCH_SIZE,
  class_mode='categorical',
  subset='validation',
  shuffle=True
)
```

```
Found 2024 images belonging to 6 classes.
Found 503 images belonging to 6 classes.
```

```
# Xception Model Setup
base_model = Xception(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
base_model.trainable = False

model = models.Sequential([
  base_model,
  layers.GlobalAveragePooling2D(),
  layers.Dense(256, activation='relu'),
  layers.Dropout(0.5),
  layers.Dense(train_generator.num_classes, activation='softmax')
])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5
83683744/83683744 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

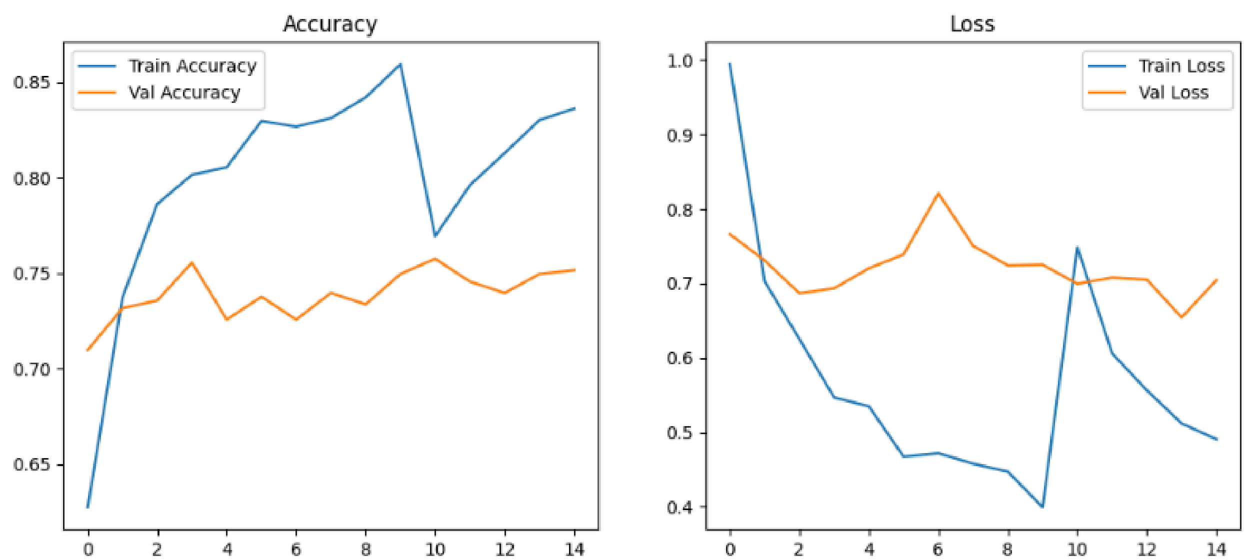
```
# Compile and Train
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
history=model.fit(train_generator,validation_data=val_generator,epochs=10)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call
  self._warn_if_super_not_called()
Epoch 1/10
64/64 ──────────────────── 433s 7s/step - accuracy: 0.5402 - loss: 1.2228 - val_accuracy: 0.7097 - val_loss: 0.7659
Epoch 2/10
64/64 ──────────────────── 274s 4s/step - accuracy: 0.7284 - loss: 0.7192 - val_accuracy: 0.7316 - val_loss: 0.7306
Epoch 3/10
64/64 ──────────────────── 242s 4s/step - accuracy: 0.7912 - loss: 0.6035 - val_accuracy: 0.7356 - val_loss: 0.6866
Epoch 4/10
64/64 ──────────────────── 275s 4s/step - accuracy: 0.7982 - loss: 0.5553 - val_accuracy: 0.7555 - val_loss: 0.6933
Epoch 5/10
64/64 ──────────────────── 280s 4s/step - accuracy: 0.8122 - loss: 0.5493 - val_accuracy: 0.7256 - val_loss: 0.7201
Epoch 6/10
64/64 ──────────────────── 274s 4s/step - accuracy: 0.8220 - loss: 0.4987 - val_accuracy: 0.7376 - val_loss: 0.7387
Epoch 7/10
64/64 ──────────────────── 237s 4s/step - accuracy: 0.8403 - loss: 0.4358 - val_accuracy: 0.7256 - val_loss: 0.8209
Epoch 8/10
64/64 ──────────────────── 240s 4s/step - accuracy: 0.8325 - loss: 0.4430 - val_accuracy: 0.7396 - val_loss: 0.7503
Epoch 9/10
64/64 ──────────────────── 240s 4s/step - accuracy: 0.8522 - loss: 0.4345 - val_accuracy: 0.7336 - val_loss: 0.7239
Epoch 10/10
64/64 ──────────────────── 235s 4s/step - accuracy: 0.8668 - loss: 0.3839 - val_accuracy: 0.7495 - val_loss: 0.7249
```

```
# Fine-tune
base_model.trainable = True
for layer in base_model.layers[:-20]:
  layer.trainable = False

model.compile(optimizer=tf.keras.optimizers.Adam(1e-5),loss='categorical_crossentropy',metrics=['accuracy'])
fine_tune_history = model.fit(train_generator,validation_data=val_generator,epochs=5)
```

```
Epoch 1/5
64/64 ──────────────────── 331s 5s/step - accuracy: 0.7488 - loss: 0.8024 - val_accuracy: 0.7575 - val_loss: 0.6994
Epoch 2/5
64/64 ──────────────────── 380s 5s/step - accuracy: 0.7811 - loss: 0.6208 - val_accuracy: 0.7455 - val_loss: 0.7077
Epoch 3/5
64/64 ──────────────────── 319s 5s/step - accuracy: 0.8091 - loss: 0.5637 - val_accuracy: 0.7396 - val_loss: 0.7050
Epoch 4/5
64/64 ──────────────────── 321s 5s/step - accuracy: 0.8396 - loss: 0.4990 - val_accuracy: 0.7495 - val_loss: 0.6542
Epoch 5/5
64/64 ──────────────────── 315s 5s/step - accuracy: 0.8410 - loss: 0.4894 - val_accuracy: 0.7515 - val_loss: 0.7042
```

```
[ ]  #Visualization
     acc = history.history['accuracy'] + fine_tune_history.history['accuracy']
     val_acc = history.history['val_accuracy'] + fine_tune_history.history['val_accuracy']
     loss = history.history['loss'] + fine_tune_history.history['loss']
     val_loss = history.history['val_loss'] + fine_tune_history.history['val_loss']

     plt.figure(figsize=(12,5))
     plt.subplot(1,2,1)
     plt.plot(acc,label='Train Accuracy')
     plt.plot(val_acc,label='Val Accuracy')
     plt.legend()
     plt.title('Accuracy')

     plt.subplot(1,2,2)
     plt.plot(loss,label='Train Loss')
     plt.plot(val_loss,label='Val Loss')
     plt.legend()
     plt.title('Loss')
     plt.show()
```



```python
# Path to your test image
img_path = '/content/drive/MyDrive/Trash_Classification/Test.jpg'

# Load and preprocess
img = load_img(img_path, target_size=IMG_SIZE)
img_array = img_to_array(img) / 255.0
img_array = np.expand_dims(img_array, axis=0)

# Predict
predictions = model.predict(img_array)

# Map prediction to class label
class_indices = train_generator.class_indices
class_labels = dict((v, k) for k, v in class_indices.items())
predicted_class = class_labels[np.argmax(predictions)]
print("Predicted class:", predicted_class)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 118ms/step
Predicted class: metal
```

Result: Accuracy & Loss trends after 15 epochs (10 initial + 5 fine-tuning):
-Training Accuracy: Improved steadily and stabilized.
-Validation Accuracy: Peaked around 85–88% depending on random seed and augmentations.
-Loss Curve: Decreased with smooth convergence and low overfitting.