Program code in python language:

```
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint

# Paths
DATA_DIR = "dataset"
MODEL_PATH = "model/waste_classifier.h5"
IMG_SIZE = 224
BATCH_SIZE = 32
EPOCHS = 10

# Load and preprocess dataset
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    horizontal_flip=True,
    zoom_range=0.2
)

train_gen = train_datagen.flow_from_directory(
    DATA_DIR,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

val_gen = train_datagen.flow_from_directory(
    DATA_DIR,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)

# Load MobileNetV2 base model
base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(IMG_SIZE, IMG_SIZE, 3))
base_model.trainable = False  # freeze base model

# Add classification head
x = base_model.output
```

```
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
output = Dense(train_gen.num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)

model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(train_gen, epochs=EPOCHS, validation_data=val_gen)

# Save model
os.makedirs("model", exist_ok=True)
model.save(MODEL_PATH)

print("✅ Model trained and saved at:", MODEL_PATH)
```

Zipcode:

```
import zipfile
import os

def zip_project_folder(folder_path, output_zip):
    # Create a ZIP file from the folder
    with zipfile.ZipFile(output_zip, 'w', zipfile.ZIP_DEFLATED) as zipf:
        for root, dirs, files in os.walk(folder_path):
            for file in files:
                file_path = os.path.join(root, file)
                arcname = os.path.relpath(file_path, folder_path)
                zipf.write(file_path, arcname)
    print(f"Project zipped successfully as {output_zip}")

# Example usage:
folder_to_zip = 'CleanTech_Project'  # Folder containing your project files
output_zip_file = 'CT-WM-TL-2025.zip'  # Desired name of the zip file

zip_project_folder(folder_to_zip, output_zip_file)
```