SUMMER RESEARCH INTERNSHIP PROGRAM 2024

Automated Waste Classification System

Project code : IP0HG0000006

By

Pankaj (22110177)

Under the guidance of

Prof. Hari Ganesh

Indian Institute of Technology, Gandhinagar

ABSTRACT

The project aims to develop an automated waste classification system that classifies municipal solid waste (MSW) into different classes, so that it can be treated accordingly. In this project waste is classified into one of these six categories - paper, plastic, metal, glass, organic and trash. Deep learning and computer vision techniques are used to achieve the aim. A pre-trained EfficientNet model was used for the task. The pre-trained model was fine tuned using transfer learning to give output as one of the six classes. After employing the model it was tested on a real world test dataset created by us. To train the model, a diverse dataset was used which was made up of a combination of 5 different garbage/waste datasets available online and a total of around 50000 images. This model is intended to be deployed on a conveyor belt system in which waste is put on the belt and a camera captures an image of the waste, which then is classified into one of the categories. Then using a tool such as an airgun can be used to push the waste into the desired bin for the waste. In this report details about methodology, implementation and results are presented.

Acknowledgements

I would like to express my sincere gratitude to professor Hari Ganesh, professor Prachi Thareja and professor Manisha Padala, without them this project would not have been possible. Their guidance,feedback and constant support made this project possible. I would also like to acknowledge staff and faculty members at IIT Gandhinagar for their invaluable support throughout the project.
Finally I would also like to thank my peers and friends for their encouragement and support. Special thanks to my parents whose belief and understanding helped me a lot. I show my gratitude to everyone who contributed to the project in any way.

Introduction

Recycling Municipal Solid Waste (MSW) is a challenging task in today's world, given the increasing amount of waste by growing population and industries. In order to do so in a sustainable manner classification of waste is essential. By efficiently managing waste we can reduce pollution and greenhouse gas, and not only that it also helps in conservation of natural resources. Each category of waste needs to be treated differently for an eco-friendly atmosphere. Traditionally manual sorting is used to segregate waste, in which workers sort the waste physically but it has many downsides. Biggest is the risk of health of the workers classifying the waste as it may expose them to hazardous substances, also it takes a huge amount of time and money to do so. Given these challenges it is the need of the hour to develop an efficient and safer method to segregate the waste. If we can somehow automate this process then these difficulties can be overcomed. This project aims to overcome these challenges by using automation as a solution to this problem. The main objective of the project is to develop a system that classifies waste into different categories based on their disposal method using techniques such as computer vision and image recognition.

We will be Primarily using Deep Learning Neural Network Models for image recognition. To achieve our aim we will employ Deep Learning Neural Networks based on convolutional neural networks (CNNs). Architecture such as EfficientNet-B0, which is a model trained on Image-Net dataset and contains 1000 classes of objects, can be used as a pre-trained model. Further we will be using transfer learning to fine tune our model according to waste classification and classify the waste into the intended classes finally. Other pre-trained models like Xception, DenseNet, ResNet, MobileNet can be used for the task but the final choice will be based on which model provides best accuracy . Techniques such as ensemble learning will be employed to further improve the accuracy of our model.To practically implement this, we intend to use a conveyor belt that will carry the waste. A camera will capture the image first and the image will be analyzed by our trained model. On the basis of the prediction given by our model that image will be classified into one of the categories of waste. Air guns will be used to physically push the waste into its designated bin. This approach not only helps make the waste management process efficient and safe but also minimizes health risks of the workers at a much lower cost. It aligns with the environmental sustainability goals, and creates a path for more eco-friendly garbage disposal methods .

Transfer Learning :
Transfer learning is a technique in which a model created for one task is applied to do another task, serving as the foundation for a new model. It frequently uses less data and computer power by utilizing the knowledge from the first task to enhance learning and performance on the new challenge. In this project it uses the knowledge of pre-trained models which were made for image processing.

Fine-tuning :
In machine learning, the process of fine-tuning involves slightly modifying the parameters of a previously trained model in order to adapt it to a new, related task. In order to let the model adapt and learn according to the new task while keeping the helpful features and representations learnt from the original task, this usually entails training the pre-trained model on the new task-specific data for a few epochs with a smaller learning rate. In this project this is used to modify the pre-trained model according to need i.e waste classification.

Methodology

Any type of waste is classified into one of these 6 classes - metal, paper, plastic, organic, glass and trash. Trash contains things like electronic wastes, thermocols, rubber etc. The model used for classification was based on the EfficientNet model which is a pre-trained model on ImageNet dataset. Now this model was fine tuned and an additional layer with activation “softmax” was added to it. All the layers of the model were frozen except for the fine tuned ones. This approach ensures that only the parameters of the additional layer were updated during training, allowing the model to specialize in garbage classification tasks. To train the model on a robust dataset, some garbage datasets available online were taken and were combined to form a big dataset. The classes present in those datasets were mapped into the desired 6 classes. The dataset was divided into three parts - 80% for the training set , 10% for the validation set and remaining 10% for the test set. Then, the fine tuned model was trained on the training set . And along with that by the help of the validation set the most optimal parameters required for the newly created layer in our model were taken. Real-time input on the model's accuracy was given via the validation set, which also directed modifications to maximize performance. After completing the training phase the model’s accuracy was assessed using the test set, which contained unseen images. In addition to the publicly available datasets, 100 images were captured

around the campus using a smartphone to create a test dataset. These images included all six categories of waste and were incorporated into the dataset to provide a real-world testing scenario. This additional dataset was used to test the model's accuracy and robustness in a practical setting, simulating real-world conditions. This assessment gave a clear picture of the model's capacity to classify new garbage photos accurately and in a general way.

There are several other popular models available that can be used as a pre-trained model, such as Xception, DenseNet, ResNet, MobileNet. However after evaluating accuracies of the model on different pre-trained models it was concluded that Efficient Net performed the best among all for this task as it was giving highest accuracy among these models. An ensemble method was used to improve the classification accuracy even more. This involved utilizing a majority vote mechanism to aggregate the predictions from many models. The ensemble approach sought to increase the system's overall robustness and classification performance by integrating the advantages of many models.Overall, this methodology achieved accurate and dependable garbage classification by utilizing cutting-edge transfer learning and fine-tuning algorithms along with a solid and well-curated dataset.

Implementation

1. Data Collection and Preprocessing:

- **Data sources:** Aggregated various publicly available garbage datasets for training, testing the model and collected 100 images around the campus using a smartphone for creating a test dataset.
- **Data mapping:** Mapped the classes in the collected datasets to the six target categories: metal, paper, plastic, organic, glass, and trash.
- **Data split:** Split the dataset into training (80%), validation (10%), and test (10%) sets to ensure robust training and evaluation.

Below is the code snippet for initial setup which includes importing necessary libraries, arranging datasets and mapping them accordingly.

```
import numpy as np
import pandas as pd
import random
import os
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow.keras as keras
import tensorflow as tf
import re
from PIL import Image
from keras.layers import Input, Dense, Flatten, GlobalAveragePooling2D, Lambda
from keras.models import Sequential
from keras.preprocessing import image
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.utils import class_weight
from keras.saving import register_keras_serializable
from keras.applications import EfficientNetB0  # Import EfficientNet model

# Define constants
IMAGE_WIDTH = 320
IMAGE_HEIGHT = 320
IMAGE_SIZE = (IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS = 3

# Define the categories
categories = {0: 'paper', 1: 'plastic', 2: 'metal', 3: 'glass', 4: 'organic', 5: 'trash'}

# Define class mappings for each dataset
class_maps = {
    "garbage_classification_12_classes": {
        'paper': 'paper', 'cardboard': 'paper', 'plastic': 'plastic', 'metal': 'metal',
        'brown-glass': 'glass', 'green-glass': 'glass', 'white-glass': 'glass',
        'biological': 'organic', 'clothes': 'trash', 'shoes': 'trash', 'battery': 'trash', 'trash':
'trash'},
    "garbage_classification_v2": {
        'paper': 'paper', 'cardboard': 'paper', 'plastic': 'plastic', 'metal': 'metal', 'glass':
'glass'},
    "trashbox_limited": {
        'paper': 'paper', 'cardboard': 'paper', 'plastic': 'plastic', 'metal': 'metal', 'glass':
'glass'},
    "trashnet":{
        'cardboard':'paper','glass':'glass','metal':'metal','paper':'paper','plastic':'plastic'},
    "waste_dataset": {
        'organic': 'organic'}
}
```

```python
# Define paths to datasets
dataset_paths = {
    "garbage_classification_12_classes": "../input/garbage-classification/garbage_classification/",
    "garbage_classification_v2": "../input/garbage-classification-v2/",
    "trashbox_limited": "../input/trashbox-limited/trashbox_limited/",
    "trashnet": "../input/trashnet/dataset-resized/",
}

# Function to add class name prefix to filenames
def add_class_name_prefix(df, col_name):
    df[col_name] = df[col_name].apply(lambda x: x[:re.search("\d", x).start()] + '/' + x)
    return df

# Create lists to store filenames and categories
filenames_list = []
categories_list = []

# Process each dataset
for dataset_name, base_path in dataset_paths.items():
    class_map = class_maps[dataset_name]
    for category in class_map:
        full_path = base_path + category
        filenames = [full_path + '/' + fname for fname in os.listdir(full_path)]
        filenames_list.extend(filenames)
        categories_list.extend([class_map[category]] * len(filenames))

# Create a DataFrame
df = pd.DataFrame({
    'filename': filenames_list,
    'category': categories_list
})

# Shuffle the dataframe
df = df.sample(frac=1).reset_index(drop=True)
```

2. Model Selection and Fine-Tuning:

- **Base Model:** Selected EfficientNet model pre-trained on the ImageNet dataset for its high performance and efficiency.
- **Fine-Tuning:** Added a new layer with a "softmax" activation function to the pre-trained EfficientNet model. All other layers were frozen to retain the learned features.

- **Hyperparameter Optimization:** Utilized the validation set to fine-tune the hyperparameters and optimize the performance of the newly added layer.

```
# Define the model
efficientnet_layer = EfficientNetB0(include_top=False, input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT,
IMAGE_CHANNELS), weights='imagenet')
efficientnet_layer.trainable = False

model = Sequential()
model.add(keras.Input(shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS)))

# Custom layer for preprocessing
@register_keras_serializable()
def efficientnet_preprocessing(img):
    return tf.keras.applications.efficientnet.preprocess_input(img)

model.add(Lambda(efficientnet_preprocessing))
model.add(efficientnet_layer)
model.add(GlobalAveragePooling2D())
model.add(Dense(len(categories), activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])
model.summary()

# Callbacks
early_stop = EarlyStopping(patience=2, verbose=1, monitor='val_categorical_accuracy', mode='max',
min_delta=0.001, restore_best_weights=True)
callbacks = [early_stop]
```

3. Training:

- **Training Setup:** Trained the fine-tuned EfficientNet model on the training set.
- **Validation:** Used the validation set to monitor the model's performance and prevent overfitting by early stopping and adjusting hyperparameters as needed.
- **Class imbalance:** Class imbalance in the dataset is handled by giving weights to each class according to the imbalance.

```python
# Split data
train_df, validate_df = train_test_split(df, test_size=0.2, random_state=42)
validate_df, test_df = train_test_split(validate_df, test_size=0.5, random_state=42)

train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)
test_df = test_df.reset_index(drop=True)
total_train = train_df.shape[0]
total_validate = validate_df.shape[0]
print('Train size =', total_train, 'Validate size =', total_validate, 'Test size =', test_df.shape[0])

# Compute class weights manually
class_weights = {}
total_samples = len(train_df)
for category in categories.values():
    class_count = len(train_df[train_df['category'] == category])
    class_weight = total_samples / (len(categories) * class_count)
    class_weights[category] = class_weight
class_weights_dict = class_weights

# Create data generators with class weights
def create_data_generator(dataframe, batch_size=64):
    datagen = ImageDataGenerator()
    generator = datagen.flow_from_dataframe(
        dataframe,
        x_col='filename',
        y_col='category',
        target_size=IMAGE_SIZE,
        class_mode='categorical',
        batch_size=batch_size,
        shuffle=True,
        class_weight=class_weights_dict
    )
    return generator

batch_size = 64

# Create combined generators
train_generator = create_data_generator(train_df, batch_size)
validation_generator = create_data_generator(validate_df, batch_size)

# Training
EPOCHS = 20
history = model.fit(
    train_generator,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=total_validate // batch_size,
    steps_per_epoch=total_train // batch_size,
    callbacks=callbacks
)
```

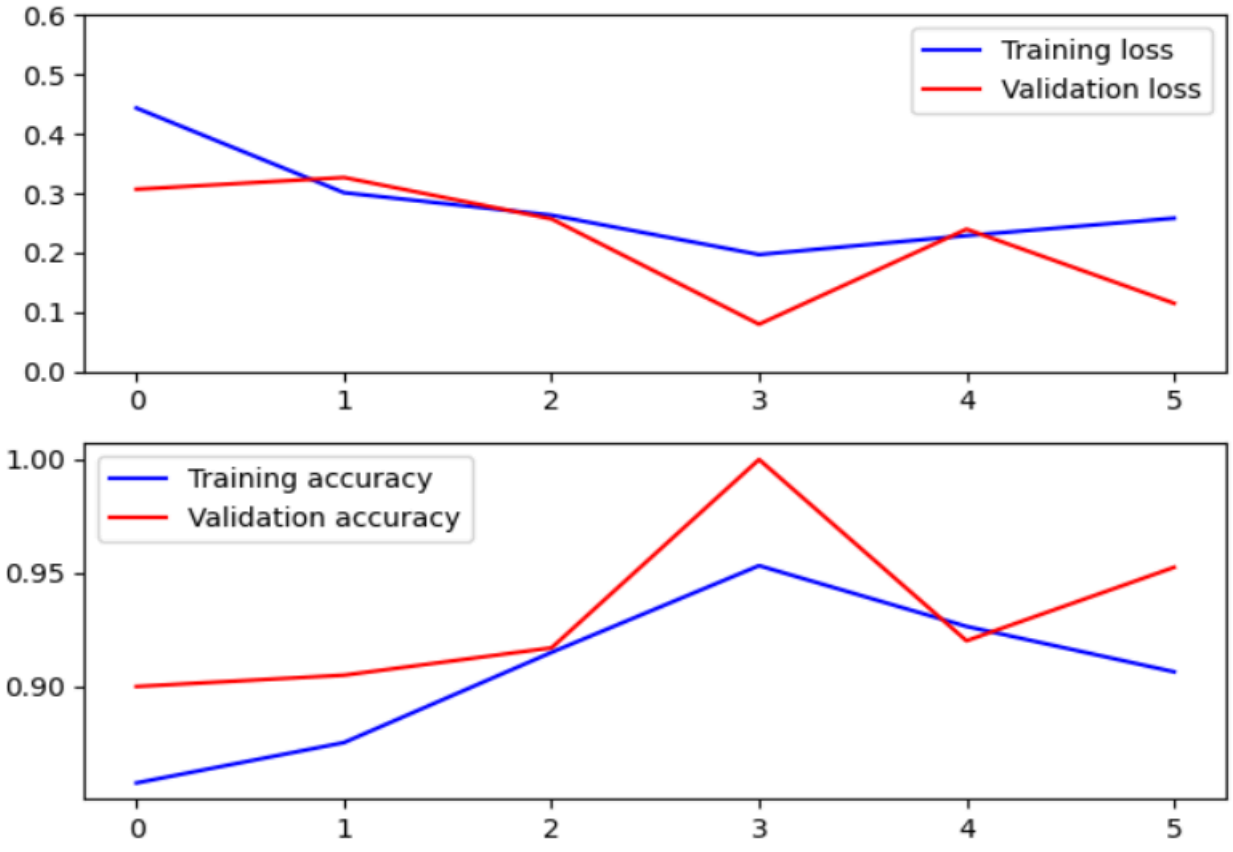
4. Testing and Evaluation:

- **Testing:** Evaluated the model's accuracy using the test set, which contained unseen data.
- **Model Evaluation:** The model is evaluated on the test set using the evaluate method. The number of evaluation steps corresponds to the number of samples in the test set, ensuring that each image is processed.

```
# Evaluation
test_datagen = ImageDataGenerator()

test_generator = test_datagen.flow_from_dataframe(
    dataframe=test_df,
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    color_mode="rgb",
    class_mode="categorical",
    batch_size=1,
    shuffle=False
)

filenames = test_generator.filenames
nb_samples = len(filenames)

results = model.evaluate(test_generator, steps=nb_samples)
accuracy = results[1]
print('Accuracy on test set =', round((accuracy * 100), 2),
'%')
```

Results and Discussion

The accuracy of this model on the training set came out to be around 92-93 %, which showed that the model is performing well. But despite this we were not sure how it would perform in the actual scenarios. So, we decided to test the model on the images taken by us from our surroundings. We took around 100 diverse images, including paper, glass, metal, organic, plastic, trash. Then we created a test dataset from those images by labeling them into their classes. When this dataset was tested on the model the accuracy came out to be around 70 %, which was significantly lower than what we got on the original test set. There can be many factors that may have contributed to this low accuracy including the image quality,lighting etc. It was noticed that it was regularly misclassifying few objects such as thermocol,wood .So a major factor can be the unseen images. There may be some objects whose images were not in the training dataset, so the model didn't get trained on them which led to misclassifying them. We can make out dataset much more robust by adding more diverse images in it which will make the model better. Here are the plots of Training, validation losses and accuracies against number of epochs (training iterations).



Metrics such as precision, recall and f1-scores are important to judge any classification algorithm/model. Here are the class wise results of these on our model.

	precision	recall	f1-score	support
glass	0.91	0.93	0.92	878
metal	0.89	0.90	0.90	555
organic	0.93	0.94	0.94	90
paper	0.94	0.97	0.96	1185
plastic	0.88	0.80	0.84	587
trash	0.97	0.96	0.96	887
accuracy			0.93	4182
macro avg	0.92	0.92	0.92	4182
weighted avg	0.93	0.93	0.93	4182

There are a lot of future aspects and room for improvement in this area. We can create a much better dataset that takes real life images into account. Additionally we can improve the model by using data argumentation, ensemble learning and advanced object detection techniques. By these efforts we can establish an eco-friendly and efficient way to waste management on a large scale.

References

[1] M. O. Rahman, A. Hussain, E. Scavino, H. Basri, and M. A. Hannan, “Intelligent computer vision system for segregating recyclable waste papers,” *Expert Systems with Applications*, vol. 38, no. 8, pp. 10398–10407, Aug. 2011, doi: https://doi.org/10.1016/j.eswa.2011.02.112.

[2] “Recyclable Waste Classification Using Computer Vision And Deep Learning | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/document/9499291

[3] M. Malik *et al.*, “Waste Classification for Sustainable Development Using Image Recognition with Deep Learning Neural Network Models,” *Sustainability*, vol. 14, no. 12, p. 7222, Jun. 2022, doi: https://doi.org/10.3390/su14127222.

[4] J. Hu and B. Zhang, “Application Research of Automatic Garbage Sorting Based on TensorFlow and OpenCV,” *Journal of Physics: Conference Series*, vol. 1883, no. 1, p. 012169, Apr. 2021, doi: https://doi.org/10.1088/1742-6596/1883/1/012169.

[5] “Garbage Classification (keras + Transfer Learning),” *kaggle.com*. https://www.kaggle.com/code/mostafaabla/garbage-classification-keras-transfer-learning

[6] “Garbage Classification (12 classes),” *www.kaggle.com*. https://www.kaggle.com/datasets/mostafaabla/garbage-classification

[7] “Garbage Dataset,” *www.kaggle.com*. https://www.kaggle.com/datasets/sumn2u/garbage-classification-v2

[8] BaoBaBu, “TrashNet,” *Kaggle.com*, 2022. https://www.kaggle.com/datasets/lzt2423654259/trashnet (accessed Jul. 03, 2024).

[9] ddeheij, “trashbox-limited,” *Kaggle.com*, 2022. https://www.kaggle.com/datasets/ddeheij/trashbox-limited (accessed Jul. 03, 2024).

[10] “Waste Classification data,” *www.kaggle.com*. https://www.kaggle.com/datasets/techsash/waste-classification-data/data