

Assessment Task 2

WASTE MANAGEMENT

Jiin Park - 24682054

Hawk He - 24504944

Muhammad Arrazy - 24682559

Rinnie Chen - 14237337

Jayden Gordoun - 13960808



Introduction to the Problem.....	4
Overview of the Project and Methodology.....	5
Overview.....	5
Project Goal.....	5
CNN.....	6
SVM.....	6
MobileNetV2.....	6
Data Collection.....	7
Source of Data.....	7
Dataset Description.....	7
Data Pre-Processing.....	7
Constants (CNN).....	7
Constants (SVM with MobileNetV2).....	7
ImageDataGenerator (CNN and SVM).....	8
Data Generators.....	8
Training & Validation Data (CNN & SVM).....	8
Model Architecture.....	9
CNN.....	9
First Convolutional Block.....	10
Second Convolutional Block.....	10
Third Convolutional Block.....	10
Flatten Layer.....	10
Dense Layers.....	10
Output.....	10
SVM with MobileNetV2.....	11
Feature Extraction in SVM with MobileNetV2.....	11
Model Initialization.....	11
Feature Extraction Process.....	11
SVM Classifier & Training.....	12
Model Evaluation.....	12
Project Management and Teamwork.....	12
Contribution Table.....	13
Results and Findings.....	14
Model Training (CNN).....	14
Model Evaluation.....	14
CNN.....	14
SVM with MobileNetV2.....	14
Classification Report Comparison.....	15
CNN.....	15
SVM with MobileNetV2.....	15
Accuracy and Loss Over Time (CNN).....	16
Confusion Matrix Comparison.....	16
Discussion and Challenges.....	17
Conclusion.....	17

Challenges.....	17
Improvements.....	18
References.....	18
Appendix.....	19

Introduction to the Problem

The global waste crisis is an escalating environmental challenge driven by rapid population growth, urbanization, and consumerism. As the volume of waste produced worldwide surges, the need for effective waste management becomes increasingly critical. Current waste management systems struggle to efficiently sort recyclable and non-recyclable materials, leading to contamination in recycling streams, reduced recycling efficiency, and heightened environmental impact due to increased landfill use and resource depletion.

Improper sorting not only limits the effectiveness of recycling processes but also contributes to pollution and greenhouse gas emissions. In this context, automating waste sorting through technological advancements could play a pivotal role in addressing these issues. Recent developments in artificial intelligence (AI) and machine learning offer promising solutions to tackle this challenge by enabling automated, precise waste classification systems.

Our project leverages machine learning, specifically Convolutional Neural Networks (CNNs) and a hybrid approach combining a pre-trained CNN with a Support Vector Machine (SVM), to create an automated waste sorting system capable of distinguishing various types of waste materials, such as metal, plastic, glass, and paper. We aim to compare the performance of a pure CNN model, which is known for its ability to capture intricate spatial patterns, with a more complex model that combines a pre-trained CNN (MobileNetV2) as a feature extractor and an SVM classifier. The MobileNetV2 model, pre-trained on large image datasets, provides a strong feature representation, while the SVM classifier leverages these features to improve classification performance, particularly in high-dimensional settings. This comparative approach allows us to explore which model offers higher accuracy and efficiency in waste classification.

This project utilizes the TrashNet dataset, a publicly available dataset containing labeled images of diverse waste categories, as the foundation for training and evaluating our models. TrashNet's structured data allows us to assess the models' performance in recognizing and categorizing waste effectively. By implementing and comparing these models, our goal is to develop a system that can generalize well across real-world applications, from industrial-scale recycling facilities to consumer-level sorting solutions. This project contributes to the larger mission of sustainable waste management by improving the accuracy and efficiency of automated waste sorting.

Overview of the Project and Methodology

Overview

Project Goal

The project's objective and goal is to sufficiently identify whether objects within a selected dataset will be able to identify and classify different materials. From deriving this information, the idea is to take this image processing technique and implement it in the real-world to reduce the amount of falsely placed trash to positively influence environmental outcomes.

For this project, the Team experimented with two different machine learning methods in order to categorise and identify items within the dataset. First, the group experimented with using a ResNet-50, a Convolutional Neural Network (CNN) that is known to excel at image classification. However, after several attempts to improve the model, the Team elected to use a hybrid SVM classifier with pre-weighted CNN (MobileNetV2).

There are four core goals that the group has set out to achieve in this task - this includes; accurate classification, hybrid model optimisation, automation of waste sorting and capitalising on real-time performance.

The classifiers and self-evaluation for the model will be conducted analysing the accuracy, precision, recall, and F1 score to determine the effectiveness of categorising different trash materials. This will ensure that the model used will be able to perform effectively in the real-world and hence show its value and application. The group will perform different methods of tuning in order to improve results and accuracy effectively. Therefore, an understanding and emphasis placed on this method of evaluation is extremely important for the result of the program.

Additionally, as mentioned previously, the utilisation and implementation of a hybrid model and optimising it to the best of its capability could potentially be a powerful solution to our business case. If managed and created effectively, the hybrid model could be the differentiator of having an extremely powerful and accurate machine learning program that is able to identify and categorise features in a variety of ways. The goal the group is laying out for this outcome is to have a CNN and SVM classifier that is able to distinctively select and differentiate false positives and negatives effectively.

Beyond the scope of the project, the Team can foresee the waste management and sorting project in a real-life application reducing reliance on manual labour to effectively manage waste solutions. If successful in accuracy, the Team believes that this system can be implemented in already installed CCTV in public areas to facilitate sorting and reducing error rates. This system will incentivise the general public to be more conscious when it comes to waste and managing the sorting efforts.

Finally, the Team wants to implement real time results and performance. For the time being, the model only generates findings every 30-45 minutes due to low computational power.

Once the prototype and the systems run at an effective rate, the group will need to be provided funding from external investors in order to facilitate a mechanism that is able to continuously keep up with waste data and classification in real-time. Since there will be high amounts of data due to the nature of it being placed in general public areas that are busy, this is key in order to scale the program effectively.

These manageable and achievable goals will be explored and elaborated further in the sections below through data collection, pre-processing, generators, architecture and results/findings.

CNN

The first type of machine learning practice that will be implemented within the project is a Convolutional Neural Networks (CNNs). A CNN model is a deep-learning practice that is commonly used and implemented for image analysis, identification and classification. The team decided to choose this model because it is able to detect different types of shapes, textures, colours and material-specific patterns very effectively which is more than ideal for the business case at hand. Because of its ability to learn features from image data, the Team understood that this will eliminate the process of manual feature engineering which was important in the decision and selection of this application. However, the most important reason that the group landed on CNN as one of the choices is because of translation invariance. More specifically, this model is able to identify and recognise the position in an image regardless of the orientation it is shown at, which will be extremely common in real-case scenarios and in the dataset that will be analysed.

SVM

Additionally, the group deemed that a Support Vector Machine (SVMs) is able to work in conjunction with CNNs effectively to assist machine learning outcomes of waste management. By grouping together these two methods, the group hopes to achieve greater performance in the model by enhanced feature extraction and classification outcomes through decision boundaries in high dimensional spaces. The SVM will allow the CNN to make quicker decisions since it is computationally lighter, especially for inference. This is instrumental in the ultimate goal of classifying objects and features in real time.

MobileNetV2

Finally, MobileNetV2 is a CNN type architecture that is able to translate and create efficient mobile embedded solutions to the model. It was deemed as a good mixture of computational power and accuracy with accurate classification and semantic segmentation. It is lightweight and is able to power and generate the model in a quick and simplified way whilst keeping the integrity of the algorithms.

Data Collection

Source of Data

For this project, We will be using the TrashNet Dataset available on GitHub. This dataset includes a large collection of trash that was categorized into metal, plastic, glass, paper, cardboard, and non-recyclable trash, which will be used to train and evaluate the machine learning model for fake news detection. The dataset provides a robust starting point for building a model to classify news articles across various topics.

From the dataset, there are 6 separate files categorized based on types of trash. We will load all files and then merge them into the same data file.

Dataset Description

TrashNet consists of 2527 images which include 501 glass, 594 paper, 403 cardboard, 482 plastic, 410 metal, and 137 trash which was already resized into 512 x 384. According to TrashNet Github (2017), The devices used to capture all the images were Apple iPhone 7 Plus, Apple iPhone 5S, and Apple iPhone SE.

Data Pre-Processing

Constants (CNN)

Image Size: 128 x 128

All images were resized to 128x128 to improve learning speeds while keeping the image relatively discernable.

Batch Size: 32

32 Images were processed in a batch to balance speed and performance.

Constants (SVM with MobileNetV2)

Image Size: 224 x 224

All images were resized to 224x224 to improve learning speeds while keeping the image relatively discernable. MobileNetV2 works best with 224x224.

Batch Size: 32

Batch sizes stay the same as CNN

ImageDataGenerator (CNN and SVM)

```
datagen = ImageDataGenerator(  
    rescale=1.0/255.0,  
    validation_split=0.2,  
    rotation_range=20,  
    width_shift_range=0.1,  
    height_shift_range=0.1,  
    shear_range=0.1,  
    zoom_range=0.1,  
    horizontal_flip=True  
)
```

The image pixel values were rescaled from the standard 0-255 to 0-1 for normalization to improve neural networks training. Next, 20% of the images were taken for validation. Lastly, performing augmentation to the image to improve positional, orientation, and other scales. We tried using augmentation due to the fact that our results were poor during the first time we did it.

Data Generators

Training & Validation Data (CNN & SVM)

```
train_generator = datagen.flow_from_directory(  
    data_path,  
    target_size=(IMG_SIZE, IMG_SIZE),  
    batch_size=BATCH_SIZE,  
    class_mode='categorical',  
    subset='training'  
)
```

```
validation_generator = datagen.flow_from_directory(  
    data_path,  
    target_size=(IMG_SIZE, IMG_SIZE),  
    batch_size=BATCH_SIZE,  
    class_mode='categorical',  
    subset='validation'  
)
```

In preparing the data for training, we utilized ImageDataGenerator to preprocess and augment the dataset for both the CNN and SVM models. The flow_from_directory function

allows the model to directly read images from directories, dynamically resizing them to the target dimensions (IMG_SIZE) specified for each model. For CNN, a smaller image size (e.g., 128x128) was used, while for MobileNetV2 with SVM, the target size was set to 224x224 to match MobileNetV2's requirements.

Both training and validation generators were set with a batch size (BATCH_SIZE), enabling efficient loading and training in manageable portions. The class_mode was defined as 'categorical,' fitting the multi-class classification objective, where each image belongs to one of six waste categories. Additionally, categorical cross-entropy was selected as the loss function due to its effectiveness in multi-class classification tasks. For the SVM feature extraction process, shuffling was disabled by setting shuffle=False to maintain a consistent image order, ensuring accurate correspondence between features and labels during extraction. This data generation approach streamlined the preprocessing, ensuring robust model input.

Model Architecture

CNN

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3)),
    MaxPooling2D(2, 2),
    Dropout(0.2),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Dropout(0.2),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Dropout(0.3),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(6, activation='softmax') # 6 classes
])
```

The model uses a Sequential model which stacks different layers on top of each other. The output of the first layer becomes the input for the second one. This model works well with our goal of stacking different convolutional layers on top of each other. The input layer of this model is an image with 3 color channels (RGB) of 128x128 size. The shape would then be (128,128,3). The model uses 3 convolutional blocks to progressively learn features from the images, with each block having a deeper level of understanding to the image.

First Convolutional Block

- Conv2D Layer: Scans patches of the images and extracts features such as edge detection (32 filters).
- MaxPooling2D layer: Downsampling by reducing image size to take the max value in a 2x2 grid.
- Dropout Layer: Randomly prevents overfitting by dropping 20% of the neurons.
- Output: A shape of (64, 64, 32)

Second Convolutional Block

- Conv2D Layer: More filters than before to detect more complex patterns (64 filters).
- MaxPooling2D layer: More downsampling
- Dropout Layer: Randomly prevents overfitting by dropping 20% of the neurons.
- Output: A shape of (32, 32, 64)

Third Convolutional Block

- Conv2D Layer: More filters to detect more patterns (128 filters).
- MaxPooling2D layer: More downsampling
- Dropout Layer: Randomly prevents overfitting by dropping 30% of the neurons.
- Output: A shape of (16, 16, 128)

Flatten Layer

- Flatten: Transforms the 3D input into a 1D output
- Output: A shape of (32768,)

Dense Layers

- Dense Layer: Combines features to form a complex understanding (128 Filters).
- Dropout: Big dropout layer by dropping 50% of the neurons to prevent overfitting in the dense layer.

Output

- Dense Layer: Chooses a category to put it into (6 neurons for 6 different trash categories)
- Softmax: Converts outputs into probabilities and chooses the highest probability

SVM with MobileNetV2

Feature Extraction in SVM with MobileNetV2

Model Initialization

```
import tensorflow as tf
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D()
])
```

We initialized a MobileNetV2 model, pre-trained on ImageNet, a large-scale image dataset. Using a pre-trained model on ImageNet allows us to benefit from the robust feature extraction capability that MobileNetV2 offers, as it has already learned to recognize a wide variety of visual patterns.

The `include_top=False` argument was set to exclude the final classification layers. By removing these layers, we enable MobileNetV2 to output feature vectors without producing final class predictions, allowing the model to generalize across new categories in our dataset.

After excluding the top layer, we added a GlobalAveragePooling2D layer. This layer compresses the spatial dimensions into a 1D feature vector of 1280 dimensions, preserving important spatial information while reducing data complexity. This dimensional reduction makes it easier for the SVM classifier to work with the features while retaining essential information for classification.

Feature Extraction Process

```
def extract_features(generator):
    features = model.predict(generator)
    labels = generator.classes
    return features, labels

train_features, train_labels = extract_features(train_generator)
val_features, val_labels = extract_features(validation_generator)
```

The function `extract_features` was created to automate the process of passing images through the MobileNetV2 model and capturing the resulting feature vectors. This function takes in an image generator (either training or validation) and produces a set of features and corresponding labels.

The `train_features` and `val_features` arrays contain the 1280-dimensional feature vectors for each image in the training and validation sets, respectively. These features represent the output of the GlobalAveragePooling2D layer, providing a concise and effective representation of each image's content.

Utilizing MobileNetV2 as a feature extractor enables us to successfully transfer the insights gained from the extensive ImageNet dataset to our particular application. The 1280-dimensional vectors contain important data on the structure, textures, and patterns of each image, making them a valuable input for the SVM classifier. This method provides a notable benefit compared to starting a model from the beginning, particularly when data is scarce, as it enables us to use pre-existing features that have proven to be effective with diverse visual data.

SVM Classifier & Training

```
svm_model = make_pipeline(StandardScaler(), SVC(kernel='rbf', C=1.0, gamma='auto'))  
  
svm_model.fit(train_features, train_labels)
```

After extracting all the features we need, we create a SVM classifier. SVM works best with standardized data, so we used StandardScaler in a pipeline. An SVM with an RBF kernel is trained on these standardized features to classify them into predefined categories.

Model Evaluation

```
1 # Predict on validation set  
2 val_predictions = svm_model.predict(val_features)  
3  
4 # Generate a classification report  
5 from sklearn.metrics import classification_report  
6 class_labels = list(validation_generator.class_indices.keys())  
7 print(classification_report(val_labels, val_predictions, target_names=class_labels))
```

After training, the model was evaluated on the validation set. The predicted labels (val_predictions) were compared against the true labels (val_labels), and the accuracy and other metrics were calculated.

Project Management and Teamwork

Our team collaborated effectively to complete both the coding and reporting aspects of this project. Each member contributed by taking responsibility for specific sections, ensuring that all tasks were covered efficiently. We organized regular check-ins to discuss progress, address challenges, and provide support to one another.

Through clear communication and division of tasks, each team member was able to focus on their assigned responsibilities while maintaining a cohesive approach to the project as a whole. This teamwork allowed us to efficiently complete the project, ensuring that each section was well-developed and integrated into the final report.

Contribution Table

Team Member Name	Description of Task
Jiin Park [24682054]	<p>Report</p> <ul style="list-style-type: none">• Data Generators - Training & Validation Data (CNN & SVM)• SVM with MobileNetV2 - Feature Extraction in SVM with MobileNetV2• Model Evaluation• Classification Report Comparison• Confusion Matrix Comparison <p>Bit of Coding Presentation slides Edit Presentation Video</p>
Muhammad Arrazy [24682559]	<p>SVM and CNN model design and code architecture Report (Some parts of Project Methodology, Results, and Discussion and Challenges) Presentation slides and design</p>
Hawk He [24504944]	<p>Report Editing Coding Presentation slides</p>
Jayden Gordoun [13960808]	<p>Report Editing Formatting Presentation Slides</p>
Rinne Chen[14237337]	<p>Report Editing Presentation Slides</p>

Results and Findings

Model Training (CNN)

```
Epoch 40/50
64/64 ————— 110s 2s/step - accuracy: 0.7146 - loss: 0.8024 - val_accuracy: 0.5865 - val_loss: 1.1725
Epoch 41/50
64/64 ————— 103s 2s/step - accuracy: 0.7486 - loss: 0.7206 - val_accuracy: 0.6243 - val_loss: 1.1577
Epoch 42/50
64/64 ————— 141s 2s/step - accuracy: 0.7426 - loss: 0.7423 - val_accuracy: 0.5805 - val_loss: 1.2279
Epoch 43/50
64/64 ————— 139s 2s/step - accuracy: 0.7195 - loss: 0.7711 - val_accuracy: 0.6123 - val_loss: 1.1369
Epoch 44/50
64/64 ————— 143s 2s/step - accuracy: 0.7241 - loss: 0.7943 - val_accuracy: 0.6103 - val_loss: 1.1907
Epoch 45/50
64/64 ————— 153s 2s/step - accuracy: 0.7472 - loss: 0.7124 - val_accuracy: 0.6203 - val_loss: 1.1359
Epoch 46/50
64/64 ————— 103s 2s/step - accuracy: 0.7487 - loss: 0.7227 - val_accuracy: 0.6223 - val_loss: 1.1334
Epoch 47/50
64/64 ————— 102s 2s/step - accuracy: 0.7265 - loss: 0.7208 - val_accuracy: 0.5845 - val_loss: 1.1487
Epoch 48/50
64/64 ————— 155s 2s/step - accuracy: 0.7398 - loss: 0.7040 - val_accuracy: 0.6183 - val_loss: 1.2060
Epoch 49/50
64/64 ————— 133s 2s/step - accuracy: 0.7396 - loss: 0.6955 - val_accuracy: 0.6223 - val_loss: 1.1642
Epoch 50/50
64/64 ————— 104s 2s/step - accuracy: 0.7251 - loss: 0.7236 - val_accuracy: 0.6501 - val_loss: 1.1509
```

The model was trained for 50 epochs using the training data. The function calculates the training loss and accuracy and the validation loss and accuracy.

Model Evaluation

CNN

```
▶ val_loss, val_accuracy = model.evaluate(validation_generator)
print(f"Validation Accuracy: {val_accuracy * 100:.2f}%")
print(f"Validation Loss: {val_loss:.4f}")

↔ 16/16 ————— 11s 686ms/step - accuracy: 0.6524 - loss: 1.1566
Validation Accuracy: 64.41%
Validation Loss: 1.1407
```

SVM with MobileNetV2

```
svm_accuracy = accuracy_score(val_labels, val_predictions)
print(f"SVM Accuracy: {svm_accuracy * 100:.2f}%")

SVM Accuracy: 72.76%
```

We evaluated the models by comparing the performance of an SVM using MobileNetV2 feature extraction and a standalone CNN model. After the training session, the CNN model achieved a validation loss of 1.1407 and a validation accuracy of 64.41%. Even with the relatively high validation loss suggesting potential issues when applied to new data, the outcome demonstrates decent classification accuracy. Moreover, the CNN performed


effectively in initial epochs but reached a plateau due to its inability to capture detailed features specific to waste categories, potentially indicating overfitting.

On the contrary, the SVM classifier, utilizing 1280-dimensional feature vectors extracted from MobileNetV2, showed an impressive validation accuracy of 72.76%. This outcome shows an enhancement of over 8% compared to the CNN model. The significant boost in precision demonstrates the effectiveness of MobileNetV2's pre-trained features in likely delivering a more dependable depiction of the image content. The SVM achieved more precise classification thanks to the inclusive input provided by these high-dimensional features. Moreover, the SVM's outstanding performance is bettered by its utilization of the RBF kernel, which is ideal for handling complex decision boundaries in feature space.

This comparison shows that the hybrid method of merging MobileNetV2 with SVM is highly efficient for complex classification tasks such as waste sorting. In the future, research could investigate more feature extraction layers or adjust MobileNetV2 to improve model accuracy and generalization.

Classification Report Comparison

CNN

 16/16

10s 595ms/step

	precision	recall	f1-score	support
cardboard	0.11	0.07	0.09	80
glass	0.23	0.35	0.28	100
metal	0.17	0.18	0.18	82
paper	0.30	0.35	0.32	118
plastic	0.26	0.11	0.16	96
trash	0.04	0.04	0.04	27
accuracy			0.22	503
macro avg	0.18	0.18	0.18	503
weighted avg	0.21	0.22	0.21	503

SVM with MobileNetV2

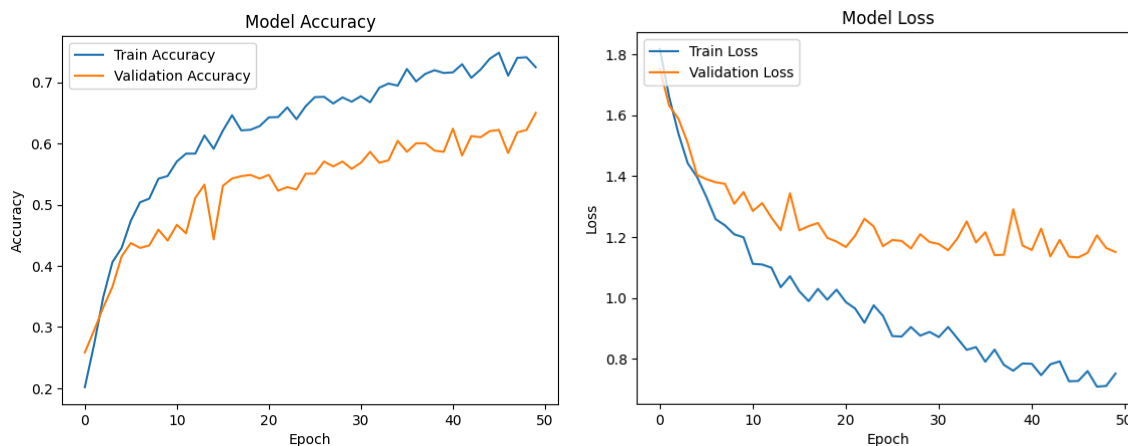
	precision	recall	f1-score	support
cardboard	0.93	0.66	0.77	80
glass	0.64	0.79	0.71	100
metal	0.76	0.83	0.80	82
paper	0.71	0.93	0.81	118
plastic	0.71	0.51	0.59	96
trash	0.64	0.26	0.37	27
accuracy			0.73	503
macro avg	0.73	0.66	0.67	503
weighted avg	0.74	0.73	0.72	503

The classification report shows an in-depth analysis of how well each model performed in every waste category. The CNN model working independently produced poor outcomes,

achieving an overall accuracy of 22% and a weighted average F1-score of just 0.21. The CNN had difficulty distinguishing features for each category, leading to low precision, recall, and F1 scores for all categories. An example is the F1-scores of 0.09 and 0.04 obtained by the cardboard and trash categories, respectively, indicating that the CNN model struggled to distinguish between the categories effectively. The model's inadequate feature extraction capacity, compared to more advanced architectures, may be the reason for its poor performance, leading to significant misclassifications.

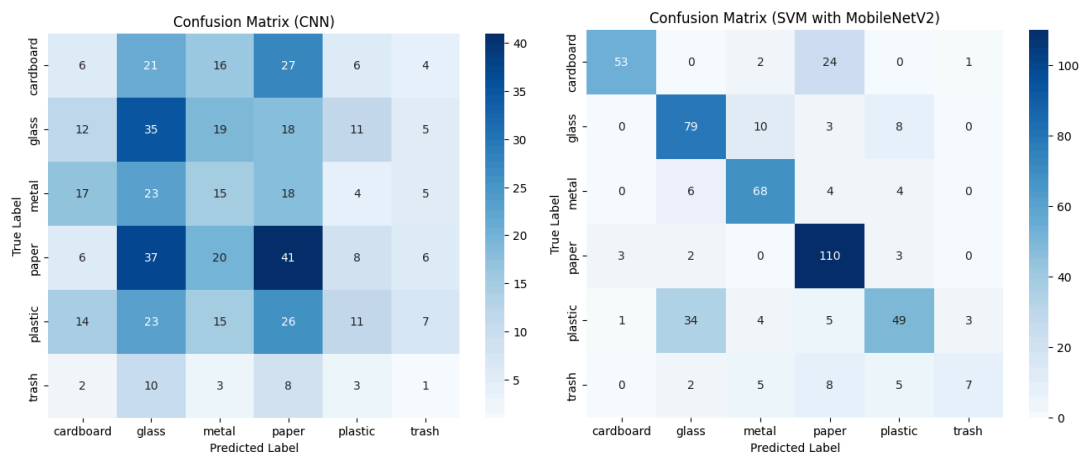
On the other hand, the SVM using the MobileNetV2 model showed a significant enhancement in classification accuracy, achieving 73% accuracy and a weighted average F1-score of 0.72. The MobileNetV2 model showed strong performance in the metal (F1-score of 0.80) and paper (F1-score of 0.81) categories, demonstrating the efficacy of its pre-trained features in capturing image content. Even with this enhancement, the SVM still struggled with the trash category, obtaining a lower F1 score of 0.37, suggesting that additional adjustments may be necessary to tackle class imbalances or feature overlaps.

Accuracy and Loss Over Time (CNN)



The two graphs shown above showcase the accuracy and loss over 50 epochs. The model accuracy and validation accuracy seem to be learning well at the start but the model was not learning very well past epoch 10-20. While training loss keeps decreasing, validation loss stagnates. This means that the model isn't really good at data it hasn't seen before, which means there might be a case of overfitting. While we have tried to prevent this with dropout layers, It seems there is need for improvement.

Confusion Matrix Comparison



While both models faced challenges with specific categories, the SVM with MobileNetV2 outperformed the standalone CNN, particularly in achieving a balanced precision-recall across classes.

The SVM model's ability to leverage MobileNetV2's pre-trained features from ImageNet was likely a key advantage, as these features provided a stronger foundation for distinguishing visual patterns in high-dimensional spaces, making the hybrid approach more effective in this classification task.

Future improvements could include further fine-tuning MobileNetV2 for waste categories specifically or exploring deeper architectures within CNN models to capture intricate patterns.

Discussion and Challenges

Conclusion

In this report CNN and SVM with MobileNetV2 was used to create a model to categorize different types of trash into recyclable categories and trash. The dataset was preprocessed to turn the images into image data and make the model more efficient, followed by augmenting and normalizing.

The CNN model demonstrated undesirable performance with only 22% accuracy in the classification report. From our testing, this has been a very unexpected behavior as CNN is commonly used for machine learning operations involving images. However, in our testing, the model was able to learn during the training scenario earning a somewhat usable 64% accuracy. However, when trying to use the model on unseen data, the model was unable to perform well. In the confusion matrix, it is seen that the model struggles with classifying them and seems to randomly classify them.

Our MobileNetV2 model with an SVM classifier was able to perform very well in tests and is able to perform with a 73% accuracy in tests. This comes with no surprise as MobileNetV2 used pre-trained weights from ImageNet which would have been a strong support base for the model. The classification report boasts strong performance for the model as f-1 scores were high with the exception of the trash category. This outlier of the trash category seems

to be an issue of the amount of data it has, as while the data for every other category exceeded 400 images, the trash category seems to have less than half of that.

In summary, the SVM classifier with MobileNetV2 was undoubtedly better than our CNN model. With strong performance in all categories except trash, SVM would be the best use for our scenario. With further tuning, such as adding more data to the trash category, the model's performance could be further optimized. This model serves as a good start in the attempt to solve waste issues by helping recycling.

Challenges

There are a lot of challenges in this assignment such as time constraints. However some of the technical challenges the team faced in this project are:

- **Unusable first CNN model**
Our team tried to only use CNN for this project and was unable to make a working model. This was a significant challenge for us as we spent quite a lot of time troubleshooting that. Our fix for this scenario was to instead make a comparison paper of our first CNN model and an SVM model that has pre-trained MobileNetV2 weights. This was very successful and we changed our project direction to that instead.
- **Troubleshooting code**
Our team had some problems along the way in designing code, as we first wanted to create a RNN + CNN hybrid instead of using MobileNetV2 with SVM. However, this was unsuccessful due to problems during the code and the team decided on using MobileNetV2 instead.

Improvements

A lot of improvements can be made to this project, especially the data and code. Some of them include:

- **Adding More Data & Variety**
There is not enough data from the trash category, which makes our model perform poorly on categorizing that specific issue. Adding more image data to that category could help our data perform much better in those scenarios. Further, a more diverse dataset can allow the model to train and learn on different outcomes and orientations of the different objects and textures of waste.
- **Trying to use other pre-trained weights**
Other pre-trained models such as ResNet could maybe be better in our scenario. Our testing has only involved MobileNetV2, which means other models could perform better but are just untested. Whilst MobileNetV2 is a good starter architecture to utilise in a CNN model, it is relatively weak and lacks much processing power to create a scalable and sustainable solution.
- **Better Data & Creating More Noise**
Our dataset was using a dataset with the images of trash in good conditions (good lighting and white background). A dataset that uses different background and lighting could be added to improve model performance. Creating more noise and disruption in the dataset will allow for better outcomes and training of the model. Additionally,

this will create a more realistic real-life dataset that would be encountered when analysing CCTV footage with blurred images and poor quality shots.

References

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. <https://doi.org/10.1109/cvpr.2018.00474>

Yang, M., Thung, G. (2017). TrashNet: Garbage Classification Dataset. *GitHub repository*. Retrieved from <https://github.com/garythung/trashnet>

Appendix

Preprocessing and CNN architecture

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

IMG_SIZE = 128
BATCH_SIZE = 32

data_path = '/content/drive/MyDrive/dataset-resized'

datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)

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

validation_generator = datagen.flow_from_directory(
    data_path,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)

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

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3)),
    MaxPooling2D(2, 2),
    Dropout(0.2),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Dropout(0.2),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Dropout(0.3),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(6, activation='softmax')
])

/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a
super().__init__(activity_regularizer=activity_regularizer, **kwargs)

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

history = model.fit(
    train_generator,
    epochs=50,
    validation_data=validation_generator
)
```

CNN Accuracy

```
[ ] model.save('model50Epoch.keras')
```

```
[ ] val_loss, val_accuracy = model.evaluate(validation_generator)
    print(f"Validation Accuracy: {val_accuracy * 100:.2f}%")
    print(f"Validation Loss: {val_loss:.4f}")
```

➡ 16/16 ————— 11s 686ms/step - accuracy: 0.6524 - loss: 1.1566
Validation Accuracy: 64.41%
Validation Loss: 1.1407

```
▶ import matplotlib.pyplot as plt
```

```
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='upper left')
plt.show()
```

```
# Plot training & validation loss values
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper left')
plt.show()
```

SVM Code

```
[ ] import tensorflow as tf
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D()
])
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_9406464/9406464 0s 0us/step

```
[ ] def extract_features(generator):
    features = model.predict(generator)
    labels = generator.classes
    return features, labels

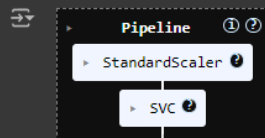
train_features, train_labels = extract_features(train_generator)
val_features, val_labels = extract_features(validation_generator)
```

/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class `self._warn_if_super_not_called()`
64/64 635s 10s/step
16/16 178s 12s/step

```
[ ] from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

svm_model = make_pipeline(StandardScaler(), SVC(kernel='rbf', C=1.0, gamma='auto'))

svm_model.fit(train_features, train_labels)
```



```
[ ] val_predictions = svm_model.predict(val_features)

from sklearn.metrics import classification_report
class_labels = list(validation_generator.class_indices.keys())
print(classification_report(val_labels, val_predictions, target_names=class_labels))
```