Group 14 Progress Report: Municipal Waste Image Classifier

Sunny Yao, Aswin Kuganesan, Third Author
{yaoh24,macid2,macid3}@mcmaster.ca

1 Introduction

The rapid increase in municipal solid waste has become a major global issue, contributing to landfill overflow, environmental pollution, and the depletion of natural resources. Effective waste management plays a vital role in addressing these challenges by promoting recycling, conserving resources, and reducing the overall environmental footprint. However, traditional methods of waste sorting are often manual, time-consuming, and error-prone, limiting the efficiency and scalability of recycling systems.

This report presents the development of a Municipal Waste Image Classification system that applies deep learning techniques to automate the identification and categorization of waste materials. The system is designed to classify common waste types such as plastic, paper, glass, metal, and organic matter based on image data. By automating the classification process, the project aims to improve recycling efficiency, streamline waste management operations, and support environmental sustainability efforts.

The main challenge of this project lies in the variability and complexity of real-world waste images. Factors such as lighting conditions, object overlap, background noise, and material degradation can make accurate classification difficult. Overcoming these challenges requires the use of robust machine learning models capable of handling diverse input data while maintaining high accuracy and efficiency. The success of this project would demonstrate the potential of artificial intelligence to transform waste management practices and contribute to the creation of more sustainable and environmentally responsible cities.

2 Related Work

Image classification for waste management has gained significant attention in recent research, with Convolutional Neural Networks (CNNs) proving particularly effective due to their ability to automatically learn hierarchical feature representations from image data. Wang et al. (2019) developed a waste classification system using transfer learning with pre-trained models like VGG16 and ResNet50, achieving over 90% accuracy on multi-class waste datasets. Their work demonstrated the effectiveness of deep features for distinguishing between waste categories including plastic, paper, and metal. Yang and Thung (2016) created one of the first comprehensive waste classification datasets and implemented a custom CNN architecture for six waste categories similar to our work. They reported challenges with distinguishing visually similar materials and emphasized the importance of data augmentation and proper preprocessing. Awe et al. (2020) proposed an ensemble approach combining multiple CNN architectures, highlighting the difficulty in classifying categories that share visual similarities, such as different types of plastics or metal objects. Kumar et al. (2021) investigated lightweight CNN architectures for edge device deployment, balancing model complexity with inference speed for practical real-time sorting applications. Our approach builds upon these works by implementing a custom CNN architecture with progressive feature extraction through multiple convolutional blocks, incorporating global average pooling to reduce overfitting, and utilizing dropout for regularization. While our dataset is currently limited in size, the model architecture follows proven design principles from the literature while being optimized for the six-category waste classification task.

3 Dataset

The dataset used is the Garbage Images Dataset from Kaggle. The dataset contains different types of garbage in folders, which allows the model to get trained on the different types of waste. The dataset is separated into cardboard, glass, metal, paper, plastic and trash. Several preprocessing operations were employed, which involved traversing each subfolder in the GarbageDataset class and collecting the paths for all images along with their corresponding class labels.

A list of tuples was created to store the path and label for all images that had a valid image format of jpg, jpeg or png, and the images were all loaded in RGB format. Using PyTorch's torchvision library, the transforms module resized the images to 224 by 224, which is ideal for image processing for the neural network while also maintaining image quality.

The transforms module also converted the images to a tensor, which normalizes the images to a [0.0, 1.0] range. One percent of the dataset is then used using an 80 percent training and 20 percent testing split, since one percent of the dataset is sufficient. The dataset was already annotated since the folders included the garbage type, so matching the images to their class label did not require any manual annotation. Each garbage type was assigned to an integer value, which was mapped to each image path.

4 Features

The dataset used in this project consists of labeled images of municipal waste categorized into classes such as plastic, paper, glass, metal, and organic waste. Each image is organized within a directory structure, where each subfolder represents one waste category. A custom GarbageDataset class was developed to efficiently load the data by reading image paths and their corresponding labels, returning (image, label) pairs suitable for use in PyTorch data loaders.

Before being fed into the model, all images undergo a series of preprocessing steps using torchvision.transforms. These include resizing the images to a uniform size, converting them into tensors, and normalizing pixel values to ensure consistent scale and distribution across the dataset. This preprocessing helps improve training stability and ensures that the model can handle variations in lighting, background, and image quality.

The model itself is a Convolutional Neural Network, which is particularly well-suited for image classification tasks. Unlike traditional approaches that rely on manual feature engineering, the CNN automatically learns to extract hierarchical visual features directly from raw image data. Early convolutional layers capture low-level features such as edges, colors, and textures, while deeper layers identify higher-level patterns like shapes and object structures. The network then uses these learned embeddings to classify each image into its corresponding waste category.

All feature extraction and classification are performed within this single end-to-end neural network pipeline. By leveraging the CNN's ability to learn rich spatial representations, the system avoids manual feature selection and instead learns robust, generalizable features directly from data. This approach enhances classification accuracy and makes the model more adaptable to real-world waste sorting applications, where visual variability is common.

5 Implementation

Our waste classification model is implemented as a PyTorch neural network comprising four convolutional blocks followed by a classification head. The architecture is designed to progressively extract increasingly complex features from input images while maintaining computational efficiency.

The model accepts RGB images of size 224×224 pixels (3 input channels) and processes them through four sequential convolutional blocks:

Block 1 consists of two 3×3 convolutional layers with 64 filters each, followed by ReLU activations and 2×2 max pooling. This block captures low-level features such as edges and simple textures. The use of padding = 1 preserves spatial dimensions before pooling.

Block 2 doubles the feature depth to 128 channels through two convolutional layers, again followed by ReLU activations and max pooling. This block learns mid-level representations that combine basic shapes and textures typical of different waste materials.

Block 3 increases feature depth to 256 channels with a single 3×3 convolution and max pooling, enabling the network to detect higher-order visual patterns related to object composition and structure.

Block 4 expands the feature space to 512 channels through a 3×3 convolutional layer followed by ReLU activation and 2×2 max pooling. This final block consolidates high-level abstractions and provides rich, discriminative features for classification. Including this block improves the model's capacity to handle the visual diversity in waste imagery without excessive overfitting.

Following the convolutional stages, we employ Global Average Pooling (AdaptiveAvgPool2d) rather than standard flattening. This reduces the number of parameters substantially by averaging each feature map into a single scalar, improving generalization on limited data.

The classification head consists of:

- A flattening layer to convert 2D features to 1D
- A fully connected layer reducing 256 features to 256 hidden units with ReLU activation
- A dropout layer (p=0.5) for regularization during training
- A final linear layer mapping to 6 output classes (cardboard, glass, metal, paper, plastic, trash)

The model is trained using the Adam optimizer with a learning rate of 0.001 and Cross-Entropy loss function, which is appropriate for multi-class classification. We employ stratified train-test splitting (80/20) to ensure balanced class representation, with a subset of 1% of the full dataset (139 samples) used for computational efficiency during development.

Data preprocessing includes resizing all images to 224×224 pixels and conversion to tensors. The batch size is set to 32 for both training and testing. Our training loop implements 15 epochs with validation after each epoch to monitor performance.

The implementation uses PyTorch 2.9.0 and is designed for easy extension to GPU acceleration, though current development is CPU-based. The modular architecture allows for straightforward modifications such as adjusting the number of filters, adding batch normalization, or implementing different pooling strategies.

6 Results and Evaluation

The dataset was divided using an 80/20 split, with 80 percent being used for training and 20 percent being used for testing. One percent of the dataset was used, and the model was trained for 15 epochs using cross-entropy loss to determine the difference between the predicted class and the true label.

Using this methodology, the training loss for the first epoch was 1.57, and the accuracy was 57.03 percent. The testing loss was 0.22 for the first epoch, and the accuracy was 100 percent. However,

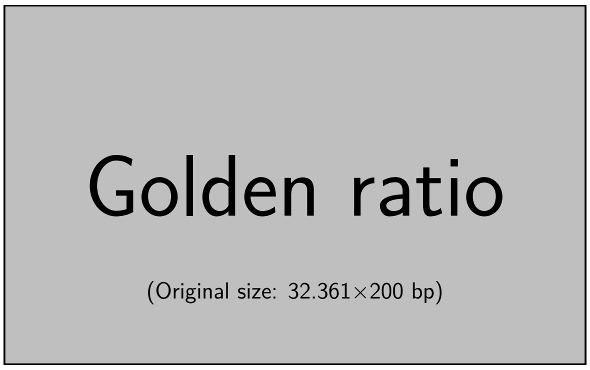


Figure 1: A figure with a caption that runs for more than one line. Example image is usually available through the `mwe` package without even mentioning it in the preamble.

in the second epoch and afterwards, the training loss and testing loss are 0, and the training and testing accuracy are 100 percent.

Although this demonstrates that the model is able to classify the waste properly, this also demonstrates that the model is overfitting to the dataset. The baseline is the neural network trained on the garbage dataset, which should be adjusted to account for overfitting, and the baseline performance is 100 percent, which is high due to overfitting.

7 Feedback and Plans

For the remainder of the project, our primary focus will be on improving the accuracy, robustness, and generalization ability of our CNN-based waste classification model. We will also explore other CNN based models to look at their techniques and see if we can incorporate them into our own model.

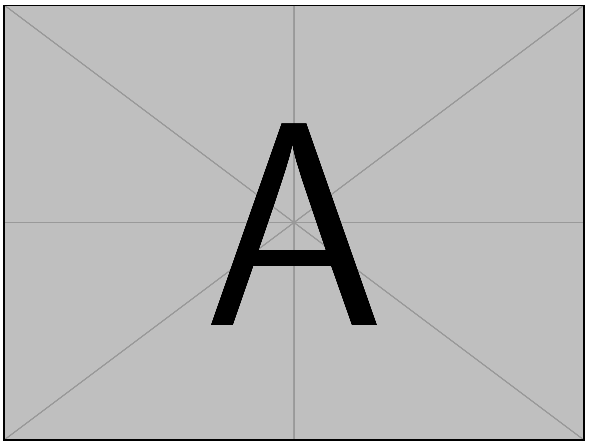
8 Template Notes

8.1 Tables and figures

See Table 1 for an example of a table and its caption. See Figure 1 for an example of a figure and its caption.

8.2 Citations

Table 1 shows the syntax supported by the style files. We encourage you to use the natbib styles. You can use the command `\citet` (cite in text) to get "author (year)" citations, like this citation to a paper by Gusfield (1997). You can use the command `\citep` (cite in parentheses) to get "(author, year)" citations (Gusfield, 1997). You can use the command `\citealp` (alternative cite without parentheses) to get "author, year" citations, which



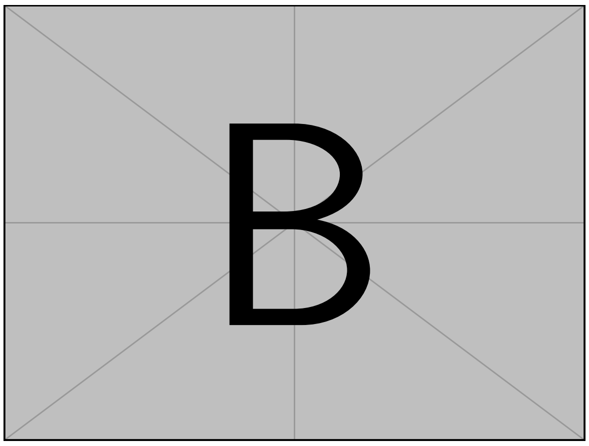


Figure 2: A minimal working example to demonstrate how to place two images side-by-side.

Output	natbib command	ACL only command
(Gusfield, 1997)	`\citep`	
Gusfield, 1997	`\citealp`	
Gusfield (1997)	`\citet`	
(1997)	`\citeyearpar`	
Gusfield's (1997)		`\citeposs`

Table 1: Citation commands supported by the style file.

is useful for using citations within parentheses (e.g. Gusfield, 1997).

8.3 References

Many websites where you can find academic papers also allow you to export a bib file for citation or bib formatted entry. Copy this into the `custom.bib` and you will be able to cite the paper in the LaTeX. You can remove the example entries.

8.4 Equations

An example equation is shown below:

$$A = \pi r^2 \tag{1}$$

Labels for equation numbers, sections, subsections, figures and tables are all defined with the `\label{label}` command and cross references to them are made with the `\ref{label}` command. This an example cross-reference to Equation 1. You can also write equations inline, like this: $A = \pi r^2$.

Team Contributions

Write in this section a few sentences describing the contributions of each team member. What did each member work on? Refer to item 7.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.