Alison Quach & Kiana Ren Deliverable 2

1. Problem statement: Restate the initial project that you proposed in deliverable one in 2 - 3 sentences.

The proposed project aims to develop a waste sorting system using CNN machine learning algorithm to classify garbage into categories. By integrating an Arduino-based hardware setup, the system will automatically detect and classify waste items, directing them to their respective bins based on the classification results.

2. Data Preprocessing: Confirm the dataset you are working with. State any changes from the initial dataset you chose. Discuss the content of the dataset (number of samples, labels, etc.). Describe and justify your data preprocessing methods (did you delete or modify any data? If so, why?).

We are using this dataset from Kaggle: Garbage Classification (12 classes) (kaggle.com)
There are 12 classes. 945 files under battery, biological 985, brown-glass 607, cardboard 891, clothes 5325, green-glass 629, metal 769, paper 1050, plastic 865, shoes 1977, trash 697, white-glass 775. Total of 15515 files. The bin won't have enough place to fit all 12 categories, so we will combine the glasses together. We will normalize the size of the data.

3. Machine learning model:

Framework used: Pytorch. https://www.kaggle.com/code/yarinbnyamin/garbage-classification-efficientnet-b0-90-acc The model architecture is based on EfficientNet, a family of CNN. EfficientNet consists of multiple blocks, each containing multiple layers. The number of layers varies depending on the depth of the network. It scalse up the network in terms of depth, width, and resolution simultaneously to achieve better performance while maintaining efficiency. In the provided code, the EfficientNet backbone is followed by a fully connected layer (Linear layer) with the number of output neurons equal to the number of classes in the classification task. This fully connected layer is responsible for mapping the features extracted by the backbone to the class probabilities.

The dataset is split into training and validation sets using a 80-20 split. Data augmentation techniques such as RandomHorizontalFlip, ColorJitter, and RandomAffine are applied to the training data to augment the dataset and improve the model's robustness. The Adam optimizer is used for optimization. Learning rate scheduling is implemented using a StepLR scheduler, which reduces the learning rate by a factor after a certain number of epochs. This helps in stabilizing and speeding up the training process.

Learning rate is set to 3e-5. The batch size is set to 64, which balances the trade-off between computational efficiency and model convergence. The number of epochs is set to 20.

The model's performance is evaluated using the validation set during training. For each epoch, the validation loss and accuracy are computed and logged. This allows monitoring the model's performance on unseen data and helps in detecting overfitting or underfitting.

4. Preliminary results:

The evaluation metric used in the model is Multiclass Accuracy, which calculates the accuracy across all classes in the classification task.

Training Progress: The model was trained for 20 epochs. From the training progress logs, it can be observed that both training and validation accuracies steadily increase over epochs, indicating that the model is learning and generalizing well.

Overfitting/Underfitting: There are no clear signs of overfitting or underfitting based on the provided logs. Both training and validation accuracies increase without significant gaps between them, suggesting that the model is not overfitting or underfitting severely. However, further analysis with additional metrics and visualizations can provide more insights.

Feasibility of the Project: The preliminary results show that the model is capable of learning the task of garbage classification to a significant extent. With a validation accuracy of approximately 89%, the model demonstrates promising performance. However, further evaluation, including testing on unseen data and comparison with other models, is necessary to assess its feasibility comprehensively.

5. Next steps: Discuss your next steps. Describe the pros/cons of your approach and future work. Will you be altering your model? For example, will you be fine-tuning it?

Increase Data Loaders' Workers: The logs suggest that the data loaders may be a bottleneck due to fewer workers. Increasing the number of workers in data loaders can improve performance.

Hyperparameter Tuning: Fine-tuning hyperparameters such as learning rate, batch size, and augmentation techniques could potentially enhance model performance.