Long Huynh Report: TrashNet

DATS 6303
Prof. Amir Jafari

1. Introduction

I focused on improving the training phase by addressing the persistent issue of overfitting observed in earlier versions developed by my team members. My contributions involved building the latest version of the code, enhancing model performance, and creating a user-friendly interface to demonstrate our results. My main goals were to reduce overfitting, improve generalization on the small and imbalanced TrashNet dataset (approximately 2,500 images across 6 classes), and provide a robust evaluation process to ensure reliable performance.

2. Identifying the Problem: Overfitting in Previous Versions

The earlier versions of our TrashNet project, specifically the baseline and version 5, suffered from overfitting. The baseline model, which used a custom CNN, lacked a validation set, making it impossible to monitor generalization to unseen data. This often leads to overfitting, especially on a small dataset like TrashNet. Version 5 improved by adding a validation set, using ResNet-18, and applying basic data augmentation (flips and rotations). However, it still lacked techniques to fully control overfitting, such as early stopping or class weighting for the imbalanced dataset. Additionally, version 5 only experimented with a single model, limiting our ability to find the best architecture for the task.

I implemented several key changes in the training phase:

- **Switch to TensorFlow/Keras with Multiple Pre-trained Models**:
  I transitioned the framework from PyTorch to TensorFlow/Keras to leverage its support for advanced pre-trained models and fine-tuning workflows. Instead of using a single model like ResNet-18, I experimented with seven pre-trained models, including EfficientNetV2S, MobileNetV2, and ResNet152V2. These models, pre-trained on ImageNet, are better suited for small datasets because they provide robust feature extraction, reducing the risk of overfitting by leveraging learned features from a large dataset.
- **Dropout for Regularization**:
  I added a dropout layer with a rate of 0.5 after the dense layer in the model architecture. During training, dropout randomly disables 50% of the neurons in each iteration, preventing the model from relying on specific neurons or over-specializing to patterns in the training data, such as a particular lighting condition in ‘glass’ images. This forces the model to distribute its learning across all neurons, improving generalization and reducing overfitting, which is critical for a small dataset like TrashNet.
- **Enhanced Data Augmentation**:
  I expanded the data augmentation beyond version 5’s basic flips and rotations to include zoom, shear, and brightness adjustments. These augmentations increase the diversity of the

training data, making the model more robust to variations and helping it generalize better, thus mitigating overfitting on the limited dataset.

- **Early Stopping**:
  I introduced early stopping with a patience of 5 epochs, monitoring validation loss. This technique halts training when the model stops improving on the validation set, ensuring it learns general patterns rather than memorizing the training data. Early stopping was crucial to prevent overfitting, especially given the small dataset size, where the risk of memorization is high.
- **Class Weighting for Imbalanced Data**:
  The TrashNet dataset is imbalanced, with 'paper' having 594 images and 'trash' only 137. Without addressing this, the model would overfit to majority classes, neglecting minority ones. I implemented class weighting using a balanced strategy, where initial weights are computed as:

$$weight\ i\ = \frac{total\ samples}{number\ of\ classes \times samples\ in\ class\ i}$$

  I then adjusted these weights with an alpha factor of 0.2 using the formula:

$$adjusted\ weight\ \ i\ = 1.0\ +\ \alpha\times(weight\ i\ -\ 1.0)$$

  This adjustment moderates the weighting strength, ensuring the model focuses more on underrepresented classes like 'trash' without overly neglecting majority classes, thus improving generalization and reducing overfitting to majority classes.

- **Two-Phase Training with Fine-Tuning**:
  I implemented a two-phase training approach: initial training with frozen base layers for 10 epochs, followed by fine-tuning the top 20 layers for another 10 epochs with a lower learning rate. This allows the model to adapt pre-trained weights to the TrashNet dataset while the lower learning rate during fine-tuning prevents drastic changes that could lead to overfitting.

3. Conclusion

I focused on improving the training phase by addressing the persistent issue of overfitting observed in earlier versions developed by my team members. My contributions involved building the latest version of the code, enhancing model performance, and creating a user-friendly interface to demonstrate our results. My main goals were to reduce overfitting, improve generalization on the small and imbalanced TrashNet dataset (approximately 2,500 images across 6 classes), and provide a robust evaluation process to ensure reliable performance.