Exploring Data Augmentation for Trash Classification

Merve Kilic
Electrical and Computer Engineering
A15978640

Abstract

This project investigates the impact of data augmentation techniques on the performance of deep learning models trained on the small TrashNet dataset. Two pre-trained models, ResNet50 and VGG16, were used to classify the trash images. Data augmentation was performed through traditional image transforms and a more experimental method using a Deep Convolutional Generative Adversarial Network (DCGAN) to generate additional training data. The results indicate that while augmentation techniques hold promise for improving performance on small datasets, the quality of the generated data significantly influences model performance. The findings, including that the original dataset-trained model performs best, provide important insights into the challenges and potential solutions when working with small datasets in the field of image classification. Overall, it is found that data augmentation for the TrashNet dataset is not a better approach to potentially be used to improve trash sorting and recycling efforts.

1 Problem and Motivation

Having an automated system that can accurately classify different types of trash based on images can greatly improve the recycling process, allowing for a more sustainable future. Proper trash sorting is critical for environmental sustainability, as it enables the recovery and recycling of valuable materials and reduces the amount of waste sent to landfills. Currently, trash sorting is often done manually, which is a time-consuming and error-prone process. Automated classification can significantly improve the efficiency and accuracy of trash sorting, which is crucial for environmental sustainability and reducing waste. Thus, in this project, I have explored trash data classification using differing data augmentation and pre-trained deep learning models.

2 Methodology

2.1 Dataset

In this project, I use the TrashNet dataset [1], a collection of images focused on waste classification. The dataset consists of a total of 2,527 images of six different types of waste: glass, paper, cardboard, plastic, metal, and trash. Each image is a color image of size 512 x 384. The images are divided into six categories, with each category pertaining to a specific type of waste:

- Glass: 501 images
- Paper: 594 images
- Cardboard: 403 images
- Plastic: 482 images
- Metal: 410 images
- Trash: 137 images

The images in the dataset are taken in different light conditions and each image has a single piece of waste in the center. The waste item can vary in its orientation and scale, which adds to the dataset's complexity and makes it a challenging task for an image classification model. This dataset provides a practical test for models designed to automate waste sorting, a task of increasing importance as we strive for more efficient and effective waste management and recycling processes.

Given the small size of this dataset, one of the primary challenges for models trained on it is generalizing to new, unseen images. As such, data augmentation, a process to artificially increase the size of the dataset by creating modified versions of the images, was used extensively in this project. The following sections outline the methodology and steps taken to implement this.

2.2 Data Preparation

Data preparation for the project involved three different approaches to mitigate the issue of small dataset size, an inherent characteristic of the TrashNet dataset used for the project.

2.2.1 No Augmentation

First, I used a vanilla dataset with no augmentation as a baseline for comparison. I split the data into an 80% training set and a 20% testing set for my experiments.

2.2.2 Traditional Augmentation

Second, the training set was augmented using PyTorch's in-built transforms library. These transformations included random vertical and horizontal flips and rotations, which are standard techniques to increase data diversity and avoid overfitting. The augmented data was then combined with the original training set to create a larger dataset for training. However, only the original dataset was used for testing.

2.2.3 Generative Augmentation

Third, the training set was augmented using DCGAN, a generative network.

A Deep Convolutional Generative Adversarial Network (DCGAN) (architecture shown in figure 1) is a type of Generative Adversarial Network (GAN) that uses convolutional layers in its generator and discriminator components. GANs consist of two neural networks, a generator and a discriminator, which are trained simultaneously. The generator attempts to produce artificial data that is indistinguishable from real data, while the discriminator network attempts to distinguish between real and fake data. Thus, the generator is trying to fool the discriminator, and the discriminator is trying to correctly classify the data as real or fake. As they train, both networks improve their skills until the generator becomes adept at creating data that the discriminator cannot distinguish from real data.

In this project, the DCGAN is used to generate synthetic images of trash for each class, increasing the number of training examples and potentially improving the ability of the model to generalize to unseen data.

The DCGAN model design in this project consists of a generator and discriminator. The generator model has a convolutional transpose layer that takes a noise vector and upsamples it to produce a tensor of shape 1024x4x4. This is followed by a series of convolutional transpose and batch normalization layers interspersed with ReLU activation functions. Each successive convolutional transpose layer increases the spatial dimension of the tensor while decreasing its depth. The final output is a 3x512x512 tensor representing an image, which is then cropped to a 3x384x512 image in the forward pass to match the TrashNet dataset.

The discriminator takes an image as input and outputs a probability indicating whether the input image is real or synthetic. The model architecture is a sequence of convolutional layers with decreasing spatial dimensions and increasing depth, each followed by a batch normalization layer and a LeakyReLU activation function. The convolutional layers extract hierarchical features from the input image. The final output from the convolutional layers is flattened and passed through a fully connected layer to produce a single output representing the probability that the input image is real. The output is then scaled to be between 0 and 1 with a sigmoid function to provide a probability.

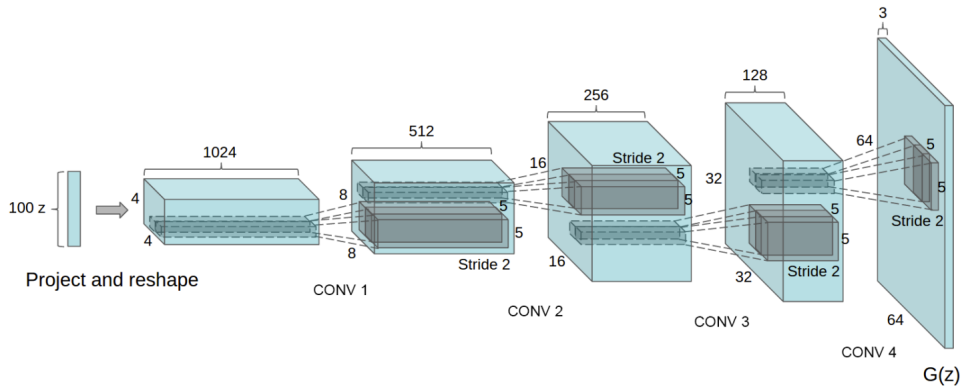

Figure 1: DCGAN Architecture [4]

2.3 Model Training

Two pre-trained Convolutional Neural Network (CNN) models were used for image classification. Both models implemented an encoder-decoder architecture, which has been successful in image classification tasks, thanks to its ability to abstract high-level features through the encoder and then reconstruct the output through the decoder.

Each model was trained on each of the three datasets (vanilla, traditionally augmented, and generatively augmented). The models' performance was then evaluated using the original test set to provide a direct comparison.

By utilizing pre-trained models, the project benefits from the feature extraction capabilities these models have developed on large, diverse image datasets, thereby compensating for the small size of the TrashNet dataset. Through this methodology, the project aims to evaluate the effectiveness of different data augmentation strategies in improving the performance of image classification tasks on small datasets.

2.3.1 ResNet

The first model used a pre-trained ResNet50 [3] model as an encoder. ResNet50 is a 50-layer deep CNN and is trained on more than a million images from the ImageNet [2] dataset. It is thus a powerful model to extract features from the trash images and classify them into respective classes.

In this model, the last fully connected layer of the ResNet50 model, which is usually responsible for the final class predictions, has been removed so that the network serves as a feature extractor. The output of this encoder is a feature map with 2048 channels.

The decoder part of the model is a custom neural network architecture that further processes the feature maps generated by the encoder. It consists of a series of four convolutional layers, each followed by batch normalization and a ReLU activation function. These layers gradually decrease the number of channels from 2048 to 128, keeping the spatial dimensions the same. Each convolutional layer uses a kernel size of 3, a stride of 1, and padding of 1. After the convolutional layers, there is an adaptive average pooling layer that reduces the spatial dimensions to 1x1, making the output invariant to the input size. Finally, the flattened features are fed into a fully connected layer which maps the 128 features to the number of classes in the task. The combination of a pre-trained ResNet50 encoder and a custom decoder makes this model capable of leveraging pre-learned feature representations and fine-tuning them for the specific task of trash classification.

2.3.2 VGG

The second model used a pre-trained VGG16 [5] model as an encoder. It was also pre-trained on ImageNet and is a common model used for image segmentation.

In this model, the encoder consists of only the feature extraction part of VGG16, which is a series of convolutional and pooling layers that convert the input image into a set of feature maps.

The decoder is a custom-made architecture designed to take the feature maps produced by the VGG16 encoder and generate class predictions. It comprises a series of four convolutional layers, each accompanied by batch normalization and a ReLU activation function. The convolutional layers use a kernel size of 3 with padding of 1 and progressively decrease the number of channels from 512 down to 64. Following these layers, an adaptive average pooling layer reduces the spatial dimensions to 1x1, producing feature vectors that are independent of the input size. The features are then flattened and passed through a fully connected layer that maps the 64 features to the number of classes for the task. By combining the pre-trained VGG16 encoder with the custom decoder, the model harnesses the power of learned feature representations from VGG16 while fine-tuning them for the specific task of trash classification.

3 Results

My experimentation and evaluation with different models and datasets yielded interesting results, summarized in Table 1. The no augmentation, traditionally augmented, and generatively augmented loss plots are shown in figures 2, 3, and 4 respectively. The loss plots for the DCGAN models for each class are shown in figure 5.

Initially, the models were trained and tested using the original TrashNet dataset (Vanilla). The ResNet model exhibited a strong performance with an accuracy of 77.27%. In contrast, the VGG model achieved an accuracy of 36.96% under the same conditions, showing that the ResNet model was superior in classifying the trash items correctly on the original dataset.

I then explored the use of data augmentation through PyTorch transforms (Transform). Interestingly, both models showed a significant decrease in performance. The ResNet model accuracy dropped to 21.15% while the VGG model had an accuracy of 18.77%. These results suggest that the additional variability introduced by the transforms may have made the classification task more challenging for these models.

In my final experiment, I utilized a DCGAN to generate additional training images for each class (DCGAN). Once again, the performance of the models decreased, with the ResNet model reaching an accuracy of 17.00% and the VGG model reaching an accuracy of 34.98%. While the VGG model's performance improved compared to the transform experiment, it was still below its performance on the vanilla dataset.

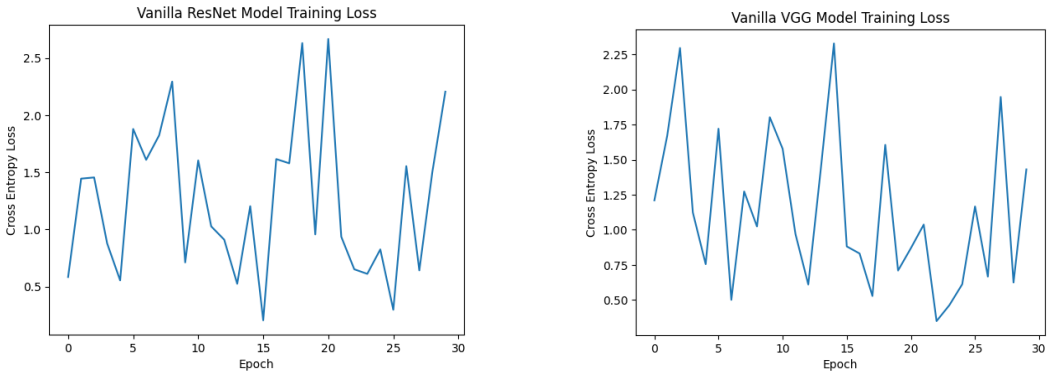

Figure 2: Vanilla Data Training Loss Plots

4 Discussion

This project's classification experiments provide key insights into the application of pre-trained deep learning models and data augmentation techniques on small datasets like the TrashNet. The striking difference in performance between the models, particularly when using the original dataset, reflects

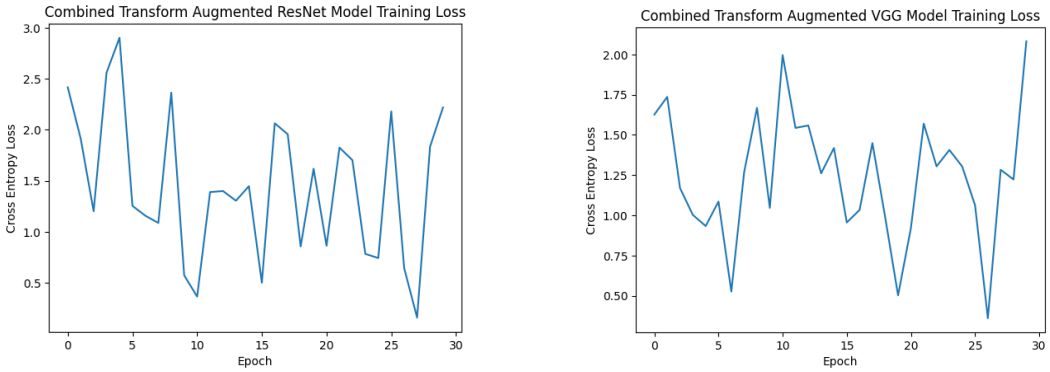

Figure 3: Transform Augmented Data Training Loss Plots

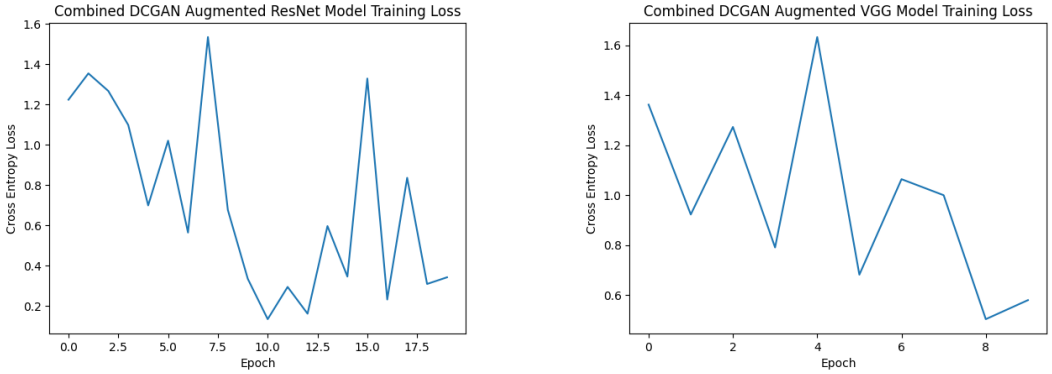

Figure 4: DCGAN Augmented Data Training Loss Plots

the power of deeper and more complex architectures such as the ResNet50. However, these results should be taken in the context of the dataset's nature, as the simplicity of TrashNet might have favored the ResNet50 model.

The significant performance decrease observed when using the transformed and DCGAN augmented datasets warrants a deeper discussion. Typically, data augmentation methods, including basic transformations such as flipping and rotating, have been found to improve the performance of deep learning models by providing more varied examples during training. This increased variety often helps models generalize better to unseen data. However, in this case, both models' performance declined when trained on the transformed dataset, suggesting that the added variability may have introduced complexities that were difficult for the models to handle, given the already challenging small size of the dataset.

My findings in the DCGAN experiment are particularly intriguing. While the idea of using GANs to artificially expand small datasets is promising, the results reveal potential obstacles when applying this method to a dataset as small and as specific as TrashNet. As the GANs were trained on the limited dataset, the models struggled to generate new, meaningful images. Most of the images produced by the DCGAN were essentially noise and failed to represent distinct, recognizable classes of trash. This is further depicted when observing the DCGAN generator and discriminator loss plots for each class shown in figure 5. The behavior of the plots seems to converge initially, but then diverge greatly, showing that they are unable to learn meaningful aspects of the data. This problem is reminiscent of overfitting, where a model learns the training data too well, including its noise and outliers, resulting in poor generalization to new data.

Therefore, the generated noisy data likely contributed to the decreased performance of both the ResNet and VGG models in the DCGAN experiment. Although the VGG model showed some improvement compared to its performance on the transformed dataset, the results were still far below those achieved with the vanilla dataset.

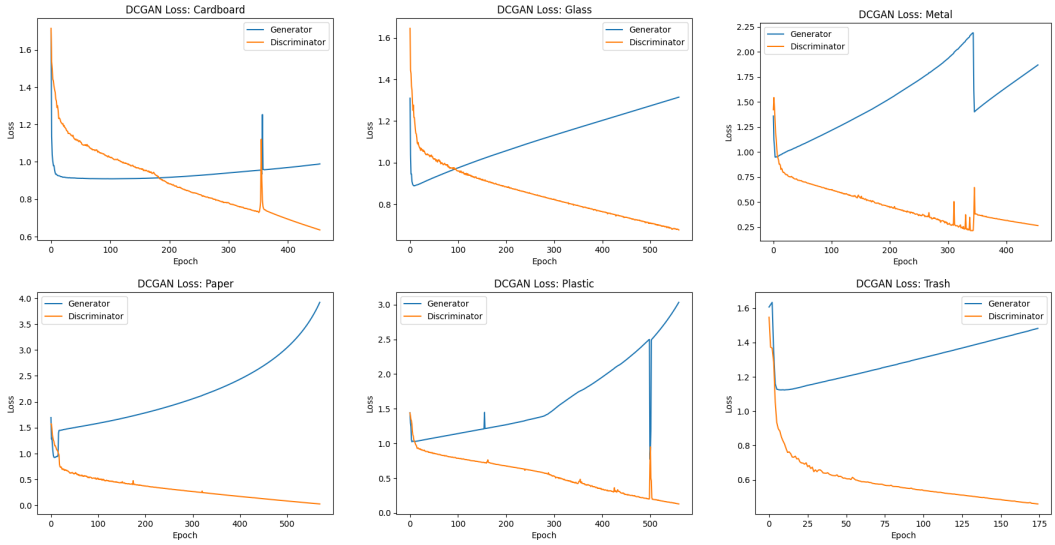

Figure 5: DCGAN Training Loss Plots

Table 1: Model Performances

Data	Model	Accuracy
Vanilla	ResNet	77.27%
Vanilla	VGG	36.96%
Transform	ResNet	21.15%
Transform	VGG	18.77%
DCGAN	ResNet	17.00%
DCGAN	VGG	34.98%

These results underline the difficulty of working with small datasets and highlight the need for careful application and evaluation of data augmentation techniques. Future work could consider alternative augmentation techniques, varying the complexity of the deep learning model, or incorporating other domain-specific knowledge to improve upon these results.

5 Conclusion

This study examined the impact of data augmentation techniques on the classification performance of deep learning models trained on a small, specific dataset, TrashNet. Using pre-trained models with different architectures, ResNet50 and VGG16, my findings demonstrate the challenges and complexities of augmenting small datasets for improved model performance. While the use of a GAN to generate additional training data was a novel approach, the quality of generated images emerged as a crucial factor affecting the final model performance. These insights not only shed light on the intricacies of working with small datasets but also highlight the potential and limitations of current data augmentation techniques. Future research can further explore the delicate balance between dataset size, augmentation methods, and model complexity to optimize the performance of deep learning models in similar scenarios.

References

[1] Rahmi Arda Aral et al. “Classification of TrashNet Dataset Based on Deep Learning Models”. In: *2018 IEEE International Conference on Big Data (Big Data)*. 2018, pp. 2058–2062. DOI: 10.1109/BigData.2018.8622212.

[2] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[3] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[4] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[5] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].