CS230

RecycleNet
Final Report

Nikhil Parab
Stanford University
nik17@stanford.edu

Cameron Tew
Stanford University
ctew@stanford.edu

Veer Shah
Stanford University
veer@stanford.edu

Abstract

Waste production and management pose a large ecological and existential risk to our planet. The use of artificial intelligence (AI) to develop an accurate, low-cost, and lightweight waste classifier garners significant environmental and economic benefits. However, most of the AI research in this area have identified data limitation as a key challenge. In turn, this paper explores to augment the strength of these existing solutions by using Generative Adversarial Network (GAN) to synthesize training data, in order to better capture recyclable waste. A robust data-set will ultimately improve the accuracy and recall of existing deep classifiers, and warrant further research and investment in this area.

1 Introduction

One of the biggest environmental challenges currently threatening our ecosystem is the colossal amount of waste production and its management. In 2017, the EPA estimated total annual municipal-solid-waste (MSW) generation was 267.8 million tons in the United States. Of the recyclable waste (70% of MSW), only 36% was properly recycled [1]. The success or failure of a recycling program is driven primarily by two stages of the recycling life cycle: collection, and processing.

Collection and Processing
While single-stream recycling models (dump everything into one bin) can help reduce the burden to the consumer, additional costs are often passed downstream in the form of capital equipment expenditures, increased manual labor, and increased contamination at the materials recovery facilities (MRFs). These facilities typically use a series of filters for well-defined objects and require manual sorting for the rest of the waste stream. Contamination in a recyclable stream of materials can be as high as 50% [9]. Some recent studies indicate that these impacts often far outweigh the benefits of convenience brought by single-stream models, with costs being as much as 28.5% higher than in a multi-stream model (where the consumer is responsible for sorting) [8].

1.1 Problem Statement and Approach

The use of artificial intelligence (AI) to develop an accurate, low-cost, and lightweight waste classifier clearly can garner significant environmental and economic benefits. Many existing AI solutions suffer from limited, highly-controlled datasets and a lack of entropy in training data (see **Related Work**). Since waste can come in various forms, orientations, lighting conditions, etc, our objective is to augment existing deep classification solutions for recyclable waste by leveraging deep convolutional generative adversarial networks (DCGANs) to synthesize training data of waste items. We evaluate the GAN by comparing the performance of a baseline classification architecture (e.g. trashnet [11])

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

with standard, real-world, training data and compare its performance when trained with a mix of real-world training data and synthesized training data generated by our trained DCGAN model.

2 Related Work

The problem of waste classification is fairly novel and subsequently understudied task. In our literature survey, we found the work done by Yang and Thung [11] to be the most relevant to our task. They have used a CNN based on AlexNet architecture to classify waste images into six categories – glass, paper, cardboard, plastic, metal and trash. However, they were unable to achieve good results and attributed the poor model performance to scarcity of data. This was a useful starting point for defining the direction of our approach. Another relevant work on waste classification by some researchers in Turkey [3] showed significant improvement over the baseline trashnet, achieving 75% - 90% accuracy. However, their best performing models came at a cost of super complex architectures with more than 100 layers, not suitable for deployment in real-time applications. Since our approach is focused on improving the entropy in the limited available training data for improving the classification results, we next directed our research towards some of the literature work focused on using data synthesis approach for improving classifier results. We found some relevant work by a team of Stanford's Deep Learning Class employing a fast R-CNN for waste object detection and classification, comparing a baseline model trained on trashnet's dataset, with GAN-generated collage waste images to augment the trashnet data [7]. Although the team's objective was object detection, the GAN-generated collage images did indeed show improvement to their Precision, Recall, and F1-Score, although their overall loss did increase. It is also unclear whether these measures were corroborated on training, or validation data; indeed, the team was not able to test their model on real data. We also found some interesting work [2] [4] where GANs were used for data augmentation and synthesis for improving results in low-data regime applications like medical imaging [5].

3 Dataset and Features

We leveraged the trashnet project for training data and as a baseline CNN for evaluating the success of our synthetic recyclable waste training images generated by the DCGAN implementation. The trashnet dataset contains six classes: glass, paper, cardboard, plastic, metal, and trash. There are 2527 images in total; the distribution of these training images is seen below:

Category	Count
Glass	501
Paper	594
Cardboard	403
Plastic	482
Metal	410
Trash	137

Figure 1: Trashnet Recyclable Waste Distribution

As the Github repository for trashnet explains, the pictures were taken by placing the object on a white posterboard and using sunlight and/or room lighting. The pictures have been scaled down to 512 x 384 for consistency. The pictures themselves were taken on one of the following devices: Apple iPhone 7 Plus, Apple iPhone 5S, or Apple iPhone SE.

Figure 2: Trashnet Recyclable Waste (glass, paper, cardboard, plastic, metal, trash)

In order to deal with the small dataset size and the non-uniform distribution of waste classes, we perform custom data synthesis on the labeled data to aid in training the DCGAN. Inspired by ideas

from an article on GAN hacks [10], we have created a data synthesis function, that takes transforms as arguments, and augments the training data by applying a series of composite transformations. Examples of transforms include horizontal/vertical flips, rotations, and color jitter. For each respective recyclable waste category, this can increase our underlying training dataset between 3-5 times (depending on the number of transforms applied).

4 Experiments and Results

We conducted several experiments to both gain an intuition on how the DCGAN works, as well as improve the generator and discriminator loss.

Running the baseline DCGAN model, we observed shortcomings in the Generator, and, reciprocally, the strength of the Discriminator. So, the primary focus of our experiments was to strengthen the Generator to make it better at fooling the Discriminator.

Model: "DCGAN_Generator"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	[(None, 1, 1, 100)]	0
layer1_ConvT (Conv2DTranspos	(None, 4, 4, 512)	819712
layer1_BatchNorm (BatchNorma	(None, 4, 4, 512)	2048
layer1_LeakyRelu (LeakyReLU)	(None, 4, 4, 512)	0
layer2_ConvT (Conv2DTranspos	(None, 8, 8, 256)	2097408
layer2_BatchNorm (BatchNorma	(None, 8, 8, 256)	1024
layer2_LeakyRelu (LeakyReLU)	(None, 8, 8, 256)	0
layer3_ConvT (Conv2DTranspos	(None, 16, 16, 128)	524416
layer3_BatchNorm (BatchNorma	(None, 16, 16, 128)	512
layer3_LeakyRelu (LeakyReLU)	(None, 16, 16, 128)	0
layer4_ConvT (Conv2DTranspos	(None, 32, 32, 64)	131136
layer4_BatchNorm (BatchNorma	(None, 32, 32, 64)	256
layer4_LeakyRelu (LeakyReLU)	(None, 32, 32, 64)	0
layer5_ConvT (Conv2DTranspos	(None, 64, 64, 3)	3075
layer5_tanh (Activation)	(None, 64, 64, 3)	0

Total params: 3,579,587
Trainable params: 3,577,667
Non-trainable params: 1,920

Model: "DCGAN_Generator"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	[(None, 1, 1, 100)]	0
layer1_ConvT (Conv2DTranspos	(None, 4, 4, 512)	819712
layer1_BatchNorm (BatchNorma	(None, 4, 4, 512)	2048
layer1_LeakyRelu (LeakyReLU)	(None, 4, 4, 512)	0
layer2_ConvT (Conv2DTranspos	(None, 8, 8, 512)	4194816
layer2_BatchNorm (BatchNorma	(None, 8, 8, 512)	2048
layer2_LeakyRelu (LeakyReLU)	(None, 8, 8, 512)	0
layer3_ConvT (Conv2DTranspos	(None, 16, 16, 512)	4194816
layer3_BatchNorm (BatchNorma	(None, 16, 16, 512)	2048
layer3_LeakyRelu (LeakyReLU)	(None, 16, 16, 512)	0
layer4_ConvT (Conv2DTranspos	(None, 32, 32, 256)	2097408
layer4_BatchNorm (BatchNorma	(None, 32, 32, 256)	1024
layer4_LeakyRelu (LeakyReLU)	(None, 32, 32, 256)	0
layer5_ConvT (Conv2DTranspos	(None, 32, 32, 128)	295040
layer5_BatchNorm (BatchNorma	(None, 32, 32, 128)	512
layer5_LeakyRelu (LeakyReLU)	(None, 32, 32, 128)	0
layer6_ConvT (Conv2DTranspos	(None, 64, 64, 3)	6147
layer6_tanh (Activation)	(None, 64, 64, 3)	0

Total params: 11,615,619
Trainable params: 11,611,779
Non-trainable params: 3,840

Figure 3: Generator Mode Architecture Changes: Baseline DCGAN vs RecycleNet DCGAN

Note that we first modified the model from the baseline inspired by Pytorch's DCGAN tutorial [6]. First, we changed the real/fake hard labels in our Discriminator pipeline to soft labels as well as added some noise by randomly flipping real/fake labels which helped in boosting the Generator. After doing some literature study on GAN activation functions, we found recommendations for using LeakyReLU for both Generator and Discriminator. Indeed, switching Generator activation functions to LeakyReLU showed some improvements.

The more substantive experiments dealt with three hyperparameters: number of network layers, the number of channels in each layer, and the effect of using different deconvolution filter kernels.

The baseline DCGAN Generator (5-layer) network has decreasing number of channels (step : 2) in each layer i.e.: 512→256→128→64→3. Changing the architecture to use large number of channels in the initial layers i.e.: 512→512→512→128→3 helped the network learn more features and showed improvements in the Generator output.

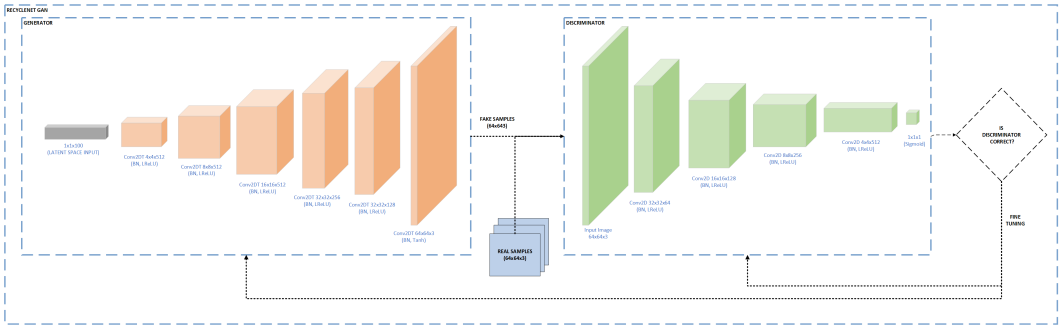


Figure 4: RecycleNet DCGAN Model

Next, we tried increasing number of layers by introducing 3x3 deconvolution filters (k: 3x3, s:1, p:1) in the Generator network. We tried multiple permutations and combinations like introducing 3x3 layer at the start, alternating 3x3 and 4x4 filter layers throughout the network and adding 3x3 layers at the end.

Based on the experiments, we noticed significant improvements on adding 3x3 deconvolution layers only at the later part of the network compared to all other combinations. The results matched our following intuitions – a 3x3 deconvolution layer at the start of Generator network does not help because the initial layers is still learning features and outputs input noise dominant, but an additional 3x3 layer at the end provides better results because at higher layers, the network is learning complex features and the output is close to the generated image.

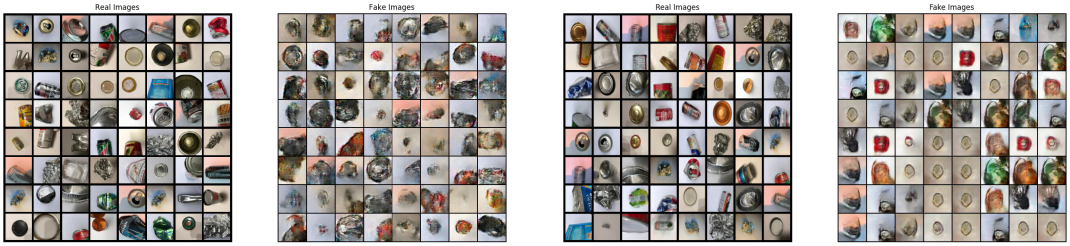


Figure 5: GAN Results: Baseline DCGAN vs RecycleNet DCGAN

From Figure 5, we observe that the 3x3 deconvolution kernel helps the layer map learned features into closer neighborhood as opposed to larger neighborhood (4x4 kernel), resulting in much sharper images and better Generator output image quality. As we can see there are some of the images are repeated in the grid of generated image. We suspect this is just random sampling issue but there is a possibility of GAN mode collapse here. This needs to be further investigated and can be an interesting topic for future work.

Alternating 4x4 and 3x3 layer throughout resulted in too deep network and problem of vanishing gradient.

Based on the above experimental results and learning, we finalized on 6-layer generator architecture (RecycleNet DCGAN) with large number of channels at the start, an additional 3x3 deconvolution fiter at the end and leakyReLU activation functions.

Once the network architecture was finalized, we let the RecycleNet DCGAN train for longer iterations by using a manual threshold (based on empirical evidence) to help control when the DCGAN was sufficiently trained and could begin generating synthetic data for a given waste classification class. In total, we generated 500 synthetic metal class images to supplement the trashnet training data, in an effort to boost the validation and test accuracy of trashnet's worst-performing classification class. The synthesized data was then indexed into the trashnet training dataset and evaluated against trashnet's baseline performance on the train/validation/test sets. These results are shown below:

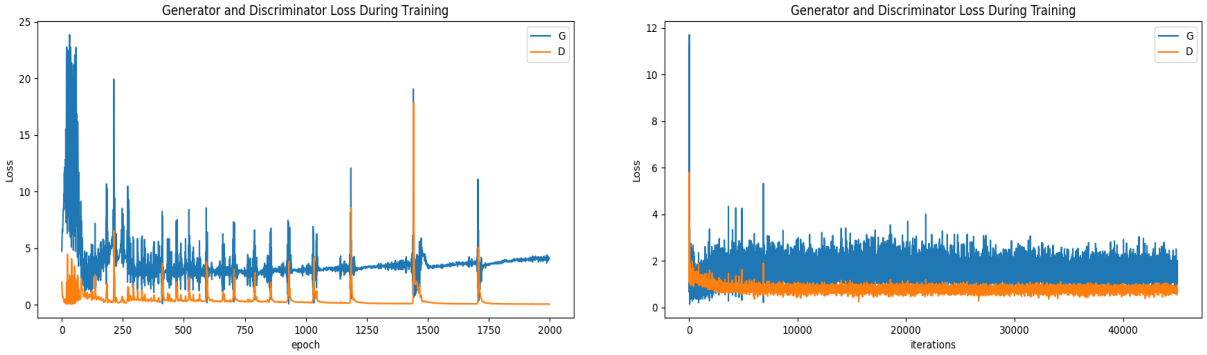


Figure 6: Loss Plot: Baseline DCGAN vs RecycleNet DCGAN

		Predicted Class												
		Trashnet Baseline							Synthetic Metal						
		G	Pa	C	Pl	M	T		G	Pa	C	Pl	M	T	
Actual Class	G	50	6	4	14	3	5	60.976%	48	2	2	12	15	3	58.537%
	Pa	9	85	2	5	6	1	78.704%	8	73	2	5	19	1	67.593%
	C	1	3	56	5	3	2	80.000%	3	1	54	3	8	1	77.143%
	Pl	9	7	5	49	4	0	66.216%	13	3	5	42	9	2	56.757%
	M	9	7	6	5	37	4	54.412%	11	5	2	0	48	2	70.588%
	T	10	1	1	3	2	12	41.379%	12	0	1	2	4	10	34.483%

Figure 7: Trashnet Confusion Matrix

From Figure 7, we see that supplementing the training data with the DCGAN synthetic metal waste images helped boost the accuracy of metal classification by 16%. This did, however, have the side effect of *decreasing* accuracy among all other classes of recyclable waste, with the largest decreases in accuracy occurring in the "paper" and "plastic" classification tasks, respectively. We postulate that this is a result of the synthetic metal data used in this exercise not being orthogonal-enough to the structural appearance of paper (e.g. crumpled, in a ball, in a pile, etc.), nor orthogonal-enough to the "shine" present in some plastic images. By supplementing the other classes with accurate synthetic data, as well as updating the RecycleGAN's loss function to perhaps take into account relative dissimilarity to other generated waste images, we think similar performance improvements are achievable among the other classes.

5 Conclusion and Future Work

In this paper, we tried to explore how classification models can be improved for applications where data limitations were identified as key challenge with the help of Generative Adversarial Networks. We targeted the most under-performing class (metal) and observed significant boost in the classification results by feeding in data generated from our RecycleNet DCGAN model. Based on our preliminary results achieved thus far, we see several key avenues for future work. Firstly, we would like to enrich our dataset for all classes using GAN and observe the implications on the classifier results. Next, we would like to experiment with some of the state-of-the-art pre-trained multi-class classification models like ResNet, MobileNet and use GAN synthesized data to train these models using transfer learning approach and evaluate the results. Also, we would like to explore several DCGAN architectures and GAN transfer learning approaches. We envision end use-case of our classifier model to be deployed on a mobile or embedded platform once we achieve good real-time classification performance and generalizability.

6 Contributions

All team members contributed equally to the direction of the project through weekly meetings and frequent check-ins. VS worked on dataloader section, hyperparamter search, running experiments and final project video. CT laid the foundational framework for the project, got the baseline trashnet up and running, performed hyperperparameter search, tuning and served as our AWS master. NP worked on implementing baseline model, final model architecture changes, hyperparameter tuning and running experiments. All the team members contributed equally to the result/error analysis, milestone/project reports, slides and the codebase. Our sincere thanks to our project mentor Shahab Mousavi for all the the support and guidance throughout the project. We thoroughly enjoyed our weekly discussions with him.

References

[1] United States Environmental Protection Agency (EPA). *Advancing Sustainable Materials Management: 2017 Fact Sheet*. 2017 (published in 2019). URL: https://www.epa.gov/sites/production/files/2019-11/documents/2017_facts_and_figures_fact_sheet_final.pdf.

[2] Antreas Antoniou, Amos Storkey, and Harrison Edwards. "DATA AUGMENTATION GENERATIVE ADVERSARIAL NETWORKS". In: *arXiv:1711.04340v3* (2018).

[3] Cenk Bircanoğlu et al. "RecycleNet: Intelligent Waste Sorting Using Deep Neural Networks". In: *2018 Innovations in Intelligent Systems and Applications (INISTA)* (2018). DOI: 10.1109/INISTA.2018.8466276. URL: https://ieeexplore.ieee.org/document/8466276.

[4] Christopher Bowles et al. "GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks". In: *CVPR* (2018). URL: https://arxiv.org/abs/1810.10863.

[5] Maayan Frid-Adar et al. "Synthetic Data Augmentation using GAN for Improved Liver Lesion Classification". In: *arXiv:1801.02385v1* (2018).

[6] Nathan Inkawhich. "DCGAN TUTORIAL". In: (2017). URL: https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html.

[7] Hrushikesh N. Kulkarni and Nandini Kannamangalam Sundara Raman. "Waste Object Detection and Classification". In: (2019). URL: http://cs230.stanford.edu/projects_fall_2019/reports/26262187.pdf.

[8] Calvin Lakhan. "A Comparison of Single and Multi-Stream Recycling Systems in Ontario, Canada". In: *Resources* 4 (June 2015), pp. 384–397. DOI: 10.3390/resources4020384.

[9] Waste Management. *The Changing Waste Stream*. 2014. URL: https://www.epa.gov/sites/production/files/2015-09/documents/changng_wste_stream.pdf.

[10] Chintala S. et al. *How to Train a GAN? Tips and tricks to make GANs work*. 2016. URL: https://github.com/soumith/ganhacks.

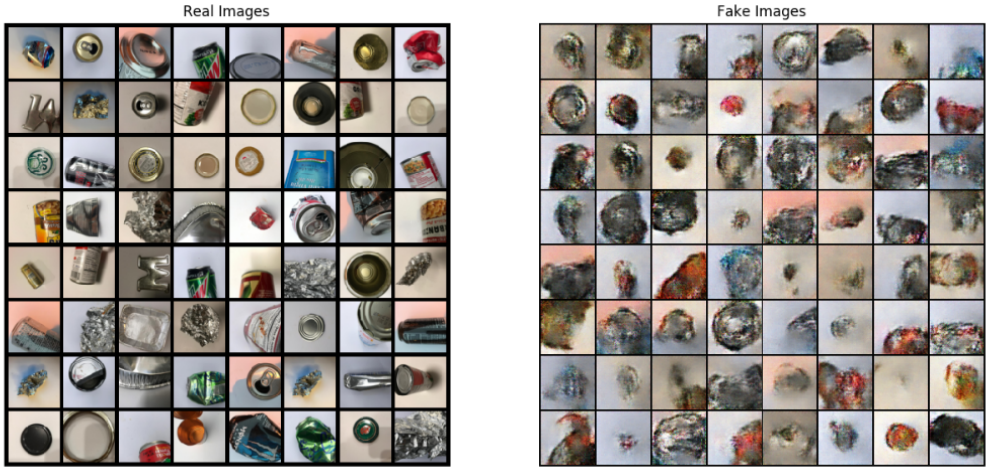
[11] G. Thung and M. Yang. 2016. URL: https://github.com/garythung/trashnet.

Appendices

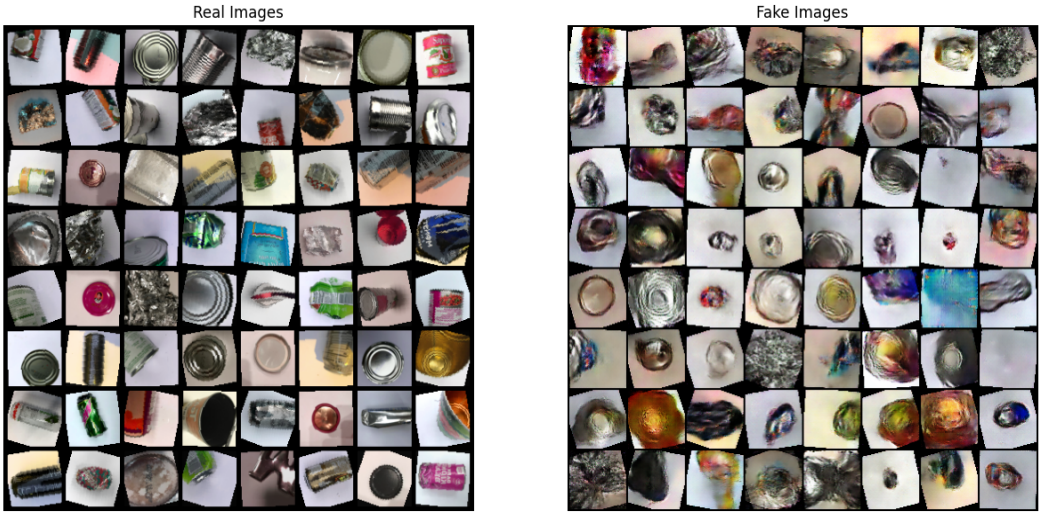
A Fake-Real Comparisons

As we can see below, the Milestone 1 fake images had less defined features and had fewer orientations, making them less believable as images of metal trash.

Milestone 1 GAN Images



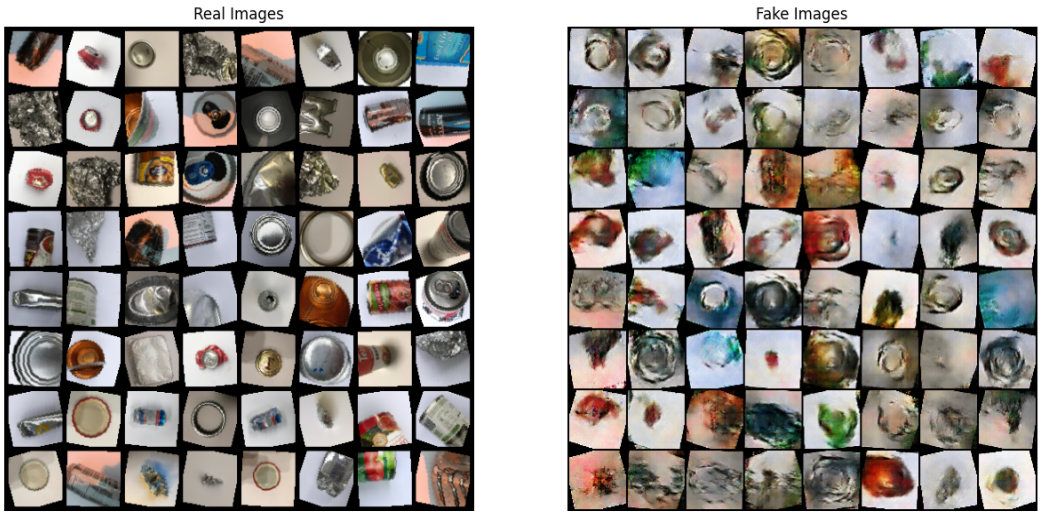
Milestone 2 GAN Images



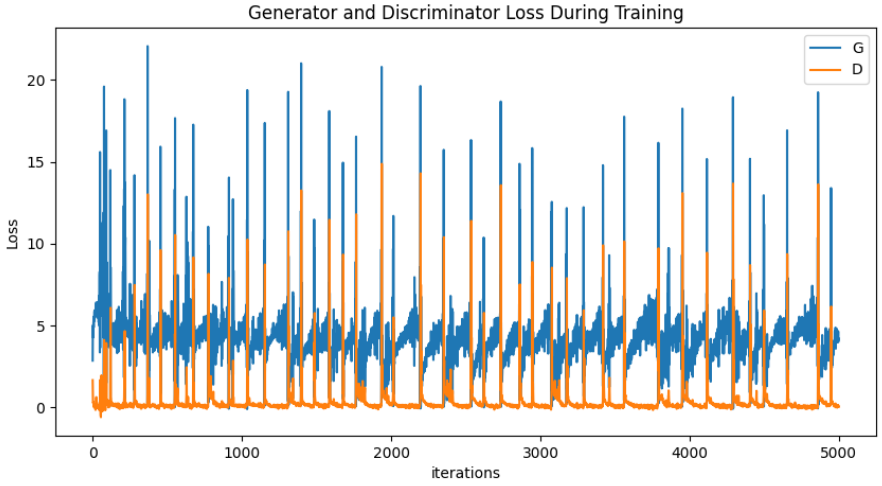
B 2 x 2 Deconv Filter Results

The baseline DCGAN uses 4x4 deconvolution filter kernel with stride of 2 and pad of 1. Switching to a 2x2 deconvolution filter kernel resulted in a drop in the GAN performance. As we can see below, the fake images (when the model had a 2x2 deconvolution filter) were extremely noisy, without many distinguishable features to the eye, which the plotted loss corroborates with the high Generator loss.

2x2 Deconv Filter Images



2x2 Deconv Filter GAN Loss



C Reduced Input Dataset

When feeding training data into the GAN architecture, we initially apply several transforms, including rotations, flips, crops, etc. Applying these transformations sequentially increases our training dataset 5 fold, and our training time skyrockets to nearly 36 hours. Thus, we attempted to significantly reduce the transforms applied in this experiment to 2, with sub par qualitative results as seen below.

Reduced Input Dataset: Real-Fake Comparison