Recyclable Materials Classification Using Deep Learning

Casey Duncan
Colorado School of Mines
1500 Illinois St, Golden, CO 80401
caseyduncan@mymail.mines.edu

Tyler Thune
Colorado School of Mines
1500 Illinois St, Golden, CO 80401
tthune@mymail.mines.edu

Abstract

This comparative analysis measures the relative efficacy of a variety of neural network architectures coupled with various pre-processing and data augmentation techniques when applied to a material classification problem. Using standard three channel (RGB) images these networks will be tasked with classifying various recyclable or non-recyclable materials. This study will focus primarily on proven Deep Learning Network architectures and their relative classification accuracy when working with a relatively small dataset (2500 base images before data augmentation techniques) as well as the efficacy of using data augmentation techniques such as geometric image transforms in improving network prediction accuracy.

1. Introduction

In recent years advances in deep neural networks have provided never before possible advances in innumerable fields. Automation and robotics are among the first to really benefit the most from these advances as it affords greater ability, capacity, and autonomy to computerized systems. Image-based object identification and classification flourished over the last decade with the seemingly endless improvements emerging from the machine-learning community. This work aims to explore the effectiveness of modern neural network architectures and machine learning techniques when applied to classifying various municipal solid waste (recycle-able) materials. The proposed networks will determine a material's classification from among five general recycle-able material categories as well as account for non-recyclable waste items bound for the landfill. The goal of designing an effective classifier specialized to this task is to ultimately provide a model that can help drive better ecological stewardship, either through helping consumers provide a purer recycling stream or empowering automated systems to better sort recyclables.

2. Previous Work

Image detection and classification remain two of the most active research areas in machine learning, algorithms, and computer vision. As computing hardware has grown more powerful and more compact, machine learning has grown to dominate more conventional bespoke methods of image processing. For nearly three decades increasingly more complex and computationally expensive models and techniques have been developed, each front runner that emerged dethroned the current best in accuracy or flexibility, and often times both. Most of the most performant techniques that have emerged involve training a neural network to read in images and provide desired knowledge about previously unprocessed images. The most successful of these networks have proven to generally be some form of Deep Neural Network.

LeCun et al [4] presented a paper just prior to the turn of the century that leveraged multi-layer neural nets that were trained using gradient descent and back-propagation as a means of classifying individual handwritten characters into the appropriate category. The network proposed for this task was dubbed LeNet-5 and was, at the time, one of the most performant methods discovered for handling this single character classification task with an error rate below 1% on the Modified NIST dataset. This work is often considered one of the seminal papers on deep learning applied to computer vision. LeNet-5 sparked a revolution in both the machine learning and computer vision worlds as it unlocked many of the key principles that would eventually lead to practical Optical Character Recognition and incredibly powerful models capable of classifying images into over 1000 categories with more than 85% accuracy.

While LeNet-5 reigned supreme for a period and began the neural network revolution it wasn't until Krizhevsky et al [3] introduced what is widely regarded as the first truly deep neural network of repute that image inference accuracy was able to make the leap beyond approximatley 40% accuracy on the ImageNet dataset. The proposed architecture, AlexNet was among the first prominent networks to leverage dropout and Rectified Linear Unit (ReLU) activation functions. AlexNet was also among the first networks to introduce max-pooling layers and data augmentation to help improve overall performance. It is also notable that the relatively simple data augmentations (image transformations, such as horizontal reflection, small translations, etc.) used not only effectively increased the data set size significantly, but also helped prevent overfitting which remains one of the most important and technically challenging aspects of neural networks design.

Following on the heels of what could be called the Deep Learning revolution increasingly deep networks were, for a time, found to only increase inference accuracy up to a point. At this point, the infamous vanishing (or exploding) gradients problem dominated networks over a certain depth. This barrier was finally broken when He et al [2] introduced Residual Networks (ReNets). The ResNet architecture overcame this fundamental limitation of deep neural networks by reformulating the problem as a layering of residual functions that prevented the continual multiplication intrinsic to neural networks of all sizes from losing valuable learning at all layers. This "deep residual learning" framework quickly established itself as the dominant deep neural network architecture and remains the basis of much of contemporary research in this area.

As it stands, there are countless network architectures and optimizations aimed at improving neural networks for nearly any dataset imaginable. In general, larger networks tend to be more performant than their smaller counterparts. This increase in size is often accompanied by a significant increase in computational complexity. And while many researchers are continuing to improve network performance and generalizability some of the most cutting edge research currently focuses on optimizing these networks for both inference accuracy and number of operations required to make an inference. Tan et al [5] are among these researchers and their work presents a framework they refer to as EfficientNets. These EfficientNets leverage a peculiarity in network scaling. The team recognized that instead of only increasing one dimension or randomly increasing dimensionality to increase the accuracy, that networks could be formulaically scaled to great effect. They discovered that when scaling a network there is a methodologically consistent way to scale that optimizes both inference efficiency as well as accuracy, and conveniently the scaling should increase by a constant ratio, leading to what they call the compound scaling method. These ratios can be determined by a grid search of optimal parameters on the smaller model and simply scaled up. The authors provide the clearest explanation, "For example, if we want to use 2N times more computational resources, then we can simply increase the network depth by α^N, width by β^N, and image size by γ^N, where α, β, and γ are constant coefficients determined by a small grid search on the original small model." The resulting scaling scheme provides a family of "EfficientNets" for any given underlying architecture that provides significant improvements in both network size and inference time while providing similar or improved prediction accuracy.

3. Approach

As noted, most modern research and resulting technologies in the world of optical image processing rely on neural networks for image classification and object identification as they have proven to be among the most effective and accurate techniques when applied. The current state of the art often relies heavily on Deep Neural Networks and often Deep Convolutional Neural Neural Networks (DCNNs) to achieve the truly impressive results that have become commonplace in recent years.

The objective of this investigation is to provide insight on the efficacy of neural networks in classifying a variety of recyclables and distinguishing which general material category an item belongs to based entirely on relatively unstructured photographs of the objects. To this end, six classes are defined for all possible municipal solid waste items: Cardboard, Glass, Metal, Paper Plastic, and Trash. Given the impressive results demonstrated by both proven and emerging neural network research, we decided to leverage a transfer learning approach using a few popular, modern network architectures to benchmark and explore the accuracy we can achieve using modern techniques on our dataset. In order to both effectively train and evaluate each network's performancemwe appropriated a canonical approach of splitting the dataset into training, validation, and test sets.

3.1. Data Selection

We began by searching for a dataset of recyclable materials sorted into multiple classes, and we found the Garbage Classification dataset [1] on Kaggle, one of the largest public data platforms. This dataset consists of 2,527 images sorted into the following 6 classes:

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

See Figure 1 for example images for each class.

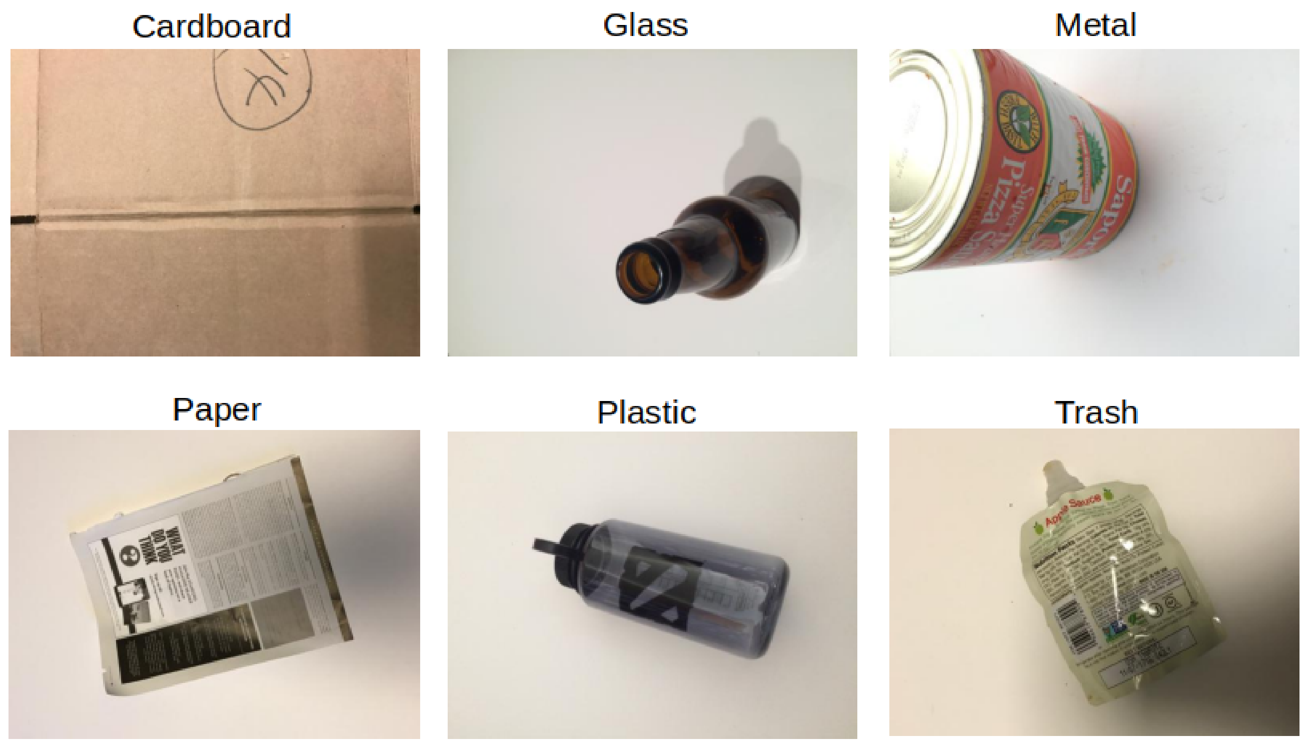

Figure 1: Recyclable Classes

3.2. Data Pre-Processing

In order to help reduce bias in our data, improve model resilience, and multiply the quantity and variety of available images, we developed a module that allows the user to selectively apply image transformations throughout the dataset. As this method preserves the original data instance while also adding additional transformed instances, these are more appropriately called data augmentations. The currently supported transformations include:

- Flip image horizontally
- Flip image vertically
- Rotate image ± 30 degrees
- Add noise to image
- Add blur to image

See Figure 2 for examples of each transformation type. It should be noted that if an augmentation type is selected by the user, it will be performed to an image randomly at every epoch. If the augmentation for horizontal flipping was enabled, there would be a 50% chance of each image in the training set actually undergoing that transformation. If the augmentation for rotation were enabled each image in the training set would have a 50% chance of being rotated by some randomly selected angle between -30 degrees and +30 degrees. The remainder of the transformations follow this paradigm save for the image blur augmentation. In this instance an image would have a 5% chance of having the blur transforme applied to it.

In addition to these five different augmentation types, every 3 channel RGB image (training, validation, and test) was down-sampled to be of size 256×341 pixels. These smaller images were then normalized to have a mean and standard deviation of 0.5. In doing so, we hoped that it would help us develop a more robust network without having to manually gather and label several thousand additional images to add to the dataset. Using the selected preprocessed dataset and augmentations, we were be able to read the images into a format that would be able to be used for training a Convolutional Neural Network (CNN) in Pytorch.

3.3. Model Architectures

During research and literature review, we found that the ResNet [2] architectures had promising results for a wide range of image classification problems. Therefore, we elected to leverage a Transfer Learning approach by using pre-trained ResNet models as a basis of our classifier. Aside from saving us the computation time of training a full ResNet from scratch this approach had the added benefit

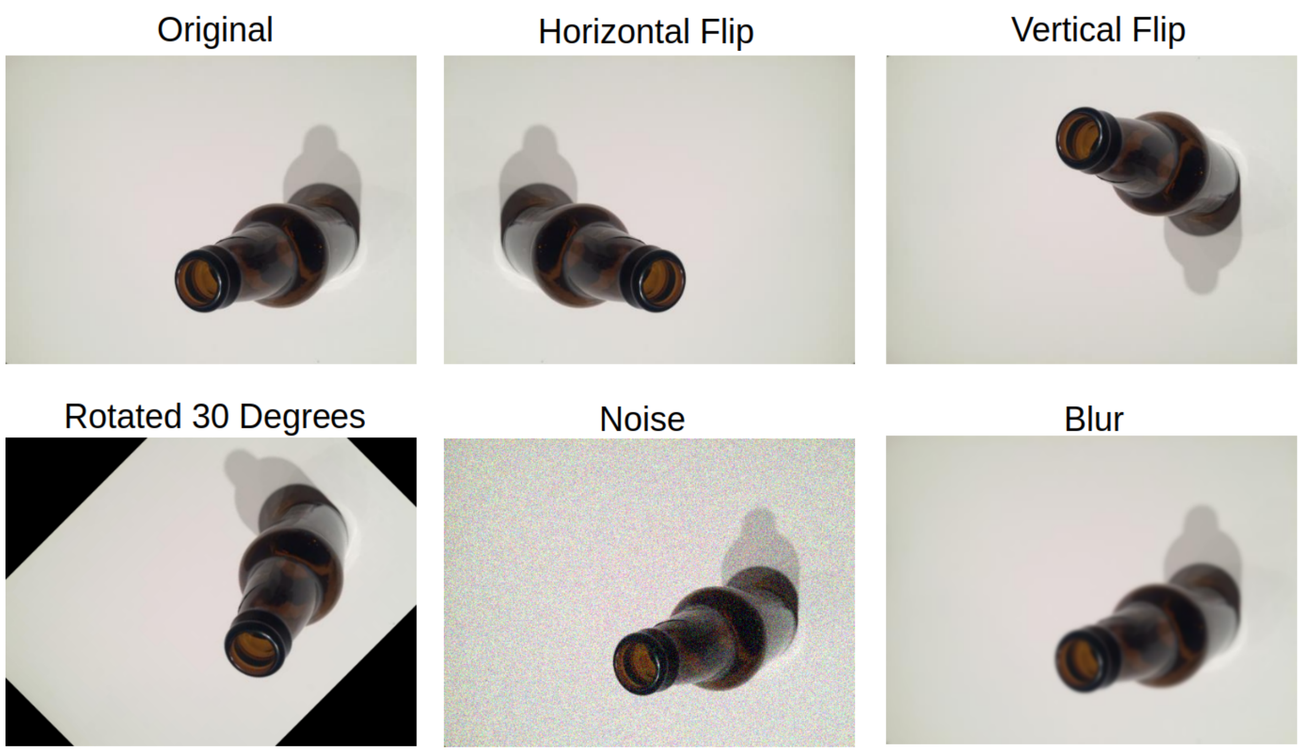

Figure 2: Augmentation Types

of providing a very well known benchmark of performance for other architectures and hyperparameter changes leveraged in both this study and in any potential future works. Transfer Learning can be summarized as using knowledge learned within a model that has already been trained on a different, usually larger, dataset to quickly train a new model on a new dataset. See Figure 3 for a general diagram of this process.

In total, we leveraged Transfer Learning on four different pre-trained models:

- ResNet18
- ResNet34
- ResNet50
- ResNeXt101

In order to allow for our model to update each of the pre-trained models, we froze the pre-trained layers so that we didn't back propagate through them during training and replaced the final fully-connected (FC) layer to update the model and provide only 6 possible classes as outputs. Additionally, we created two additional models that we called ResNet50-modnet & ResNeXt101-modnet, which differed from the ResNet50 & ResNeXt101 models in that each model's final fully connected layer was replaced by a fully connected layer with ReLu activation function, a 50% droput layer, a second fully connected layer with a LogSoftmax activation function. Results for each model are detailed below in Section 4.

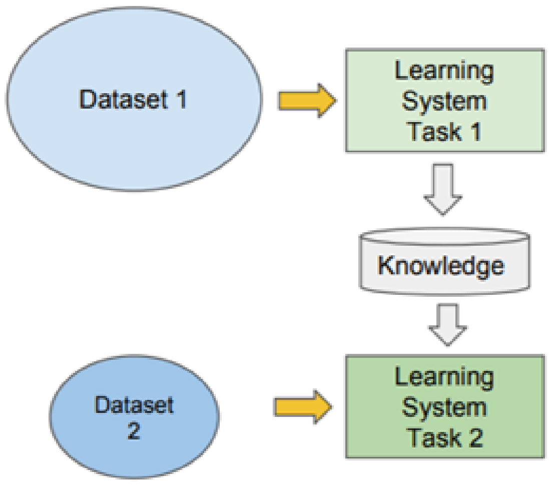

Figure 3: Transfer Learning

Table 1: Model Architectures, Epochs, Augmentations, & Accuracies

Test No.	Model	Epochs	Aug Qty	Aug Types	Accuracy
1	ResNet18	5	0	None	0.804
2	ResNet18	10	0	None	0.790
3	ResNet18	15	0	None	0.822
4	ResNet18	25	0	None	0.844
5	ResNet18	5	2	Horz Flip, Vert Flip	0.790
6	ResNet18	5	5	All	0.770
7	ResNet34	5	0	None	0.790
8	ResNet34	10	0	None	0.830
9	ResNet34	15	0	None	0.804
10	ResNet34	25	0	None	0.853
11	ResNet34	5	2	Horz Flip, Vert Flip	0.822
12	ResNet34	5	5	All	0.758
13	ResNet50	5	0	None	0.826
14	ResNet50	10	0	None	0.812
15	ResNet50	15	0	None	0.861
16	ResNet50	25	0	None	0.820
17	ResNet50	5	2	Horz Flip, Vert Flip	0.871
18	ResNet50	5	5	All	0.816
19	ResNet50	10	3	Horz Flip, Vert Flip, Rot $\pm 30°$	0.745
20	ResNet50-modnet	5	0	None	0.808
21	ResNet50-modnet	10	0	None	0.844
22	ResNet50-modnet	15	0	None	0.879
23	ResNet50-modnet	25	0	None	0.907
24	ResNet50-modnet	5	2	Horz Flip, Vert Flip	0.848
25	ResNet50-modnet	5	5	All	0.774
26	ResNet50-modnet	10	5	All	0.782
27	ResNet50-modnet	15	5	All	0.841
28	ResNet50-modnet	25	5	All	0.820
29	ResNet50-modnet	25	2	Horz Flip, Vert Flip	0.844
30	ResNet50-modnet	8	3	Horz Flip, Vert Flip, Rot $\pm 30°$	0.845
31	ResNeXt101-modnet	20	3	Horz Flip, Vert Flip, Rot $\pm 30°$	0.832
32	ResNeXt101-modnet	5	0	None	0.861

3.4. Command Line Interface (CLI) Module

Pytorch provides multiple built in functions that allow the user to perform augmentations to images, randomly sample data, and easily split data into user specified training, validation, and testing ratios all without needing to write their own modules. Additionally, it allows you to elegantly batch your data so that the users, developers, and machine learning practitioners do not have to worry about over-utilizing their memory. In order to provide a flexible means of experimenting with various data augmentations, data ratio splits, and epoch quantities we elected to wrap much of this functionality in a Command Line Interface that allowed for a relatively simple iteration process when experimenting with various combinations, while preventing accidental mis-configuration. More documentation regarding the CLI can be found in the project's README.md file.

4. Experiments and Results

Using the five different model architectures mentioned in section 3.3, we trained a total of 32 models varying the number of training epochs and augmentations in an attempt to find which architecture-hyperparameter combination performed best on our dataset. These model-hyperparameter combinations are shown in Table 1. For each of the models, we split the dataset into 60% training, 20% validation, and 20% testing. Additionally, we batched each data split into a batch size of 8 images to avoid over-utilizing the memory. As each model trained, we used the validation set to test our

Table 2: Confusion Matrix for Best Model

Class	Cardboard	Glass	Metal	Paper	Plastic	Trash
Cardboard	78	1	3	4	0	0
Glass	0	100	3	0	3	0
Metal	0	4	86	3	1	1
Paper	0	0	0	105	1	1
Plastic	0	7	3	0	74	0
Trash	0	5	3	4	1	15

model every 100 batches trained and prevent it from over fitting to the training data. Once training was completed, we used the testing dataset to test the accuracy of our model. Figures 4 & 5 summarize the results of some meta-analysis efforts. The ResNet50-modnet architecture was the most performant on test data, with an accuracy of 90.7% after training on out dataset for 25 epochs without any augmentations. The confusion matrix for this model can be seen in Table 2, with predicted values as the columns and true values as the rows, showing that paper performed the best and trash performed the worst being most confused with glass.

It is worth noting that the 'Trash' class was the most problematic class, performing with the lowest accuracy of all six classes in 23 out of 32 of the trained models. 'Trash' was confused with the 'Metal' class the most often (11 out of 23 instances), which likely has to do with shiny chip bags & candy wrappers taking up a significant portion of the 'Trash' dataset. Of the six classes, 'Trash' was expected to perform most poorly as the size of the 'Trash' dataset was only about 25% of the size of the other five class datasets.

5. Discussion

While not a fully exhaustive exploration, some general trends became clear in the limited number of architectures and permutations we attempted. As demonstrated in Figure 4, the number of epochs is loosely correlated with improved performance. This is by no means a hard and fast rule, as some model-hyperparameter combinations began to suffer from performance degradation after too many epochs elapsed. This behavior is often referred to as "overfitting" and is a well known, well documented phenomenon in the world of machine learning. There are many canonical approaches to help prevent this problem that will be discussed later in section 5.1. As one may expect this phenomenon is difficult, if not impossible to predict accurately, and as such means that some of the presented accuracy scores could still be improved simply with additional training time while others would either level off or even regress to far less accurate inference capability.

What is far less intuitive, though, is the general effect of data augmentation on model inference capability. Despite a wide consensus and a significant body of experimental evidence that data augmentation often provides more robust and accurate models this study demonstrated that, for this particular dataset and for the selected augmentation methods selected, there was virtually no correlation between accuracy and augmentations. Note that in the instances where accuracy did improve with data augmentation (as shown in Figure 5) the two augmentations used were Horizontal Flipping and Vertical Flipping. In all such instances the testing accuracy ended up decreasing below the "No augmentation" level when all five available augmentations are used. This does beg the question of whether or not the nature of the data augmentation has a larger effect or if the number of augmentations plays a part. This study does not have extensive enough evidence to detangle these variables. Additionally, two of the five controlled groups demonstrated no improvement from data augmentation whatsoever. In fact, they only showed an overall decrease in accuracy.

In order to better optimize this system, a far more exhaustive search of the the epoch and augmentation space would be necessary to determine how these parameters could be tuned to best optimize inference accuracy.

5.1. Future Work

This study provides both a nicely expandable technical framework as well as a stepping-off point for additional research on this particular application area of neural network based recycling classifications. Assuredly there are countless improvements and virtually limitless tuning combinations. From this infinite set of possibilities a small horizon of future work seems evident.

In order to truly address the effect of epochs on testing accuracy some changes would need to be made to the stopping criteria. Rather than using a set number of epochs, a more modern approach leveraging early stopping based on validation set accuracy should be used. These approaches when properly implemented demonstrably finds the correct number of epochs to train a model by training until overfitting is detected. Some methods then retain the most performant model and continue training for an additional few

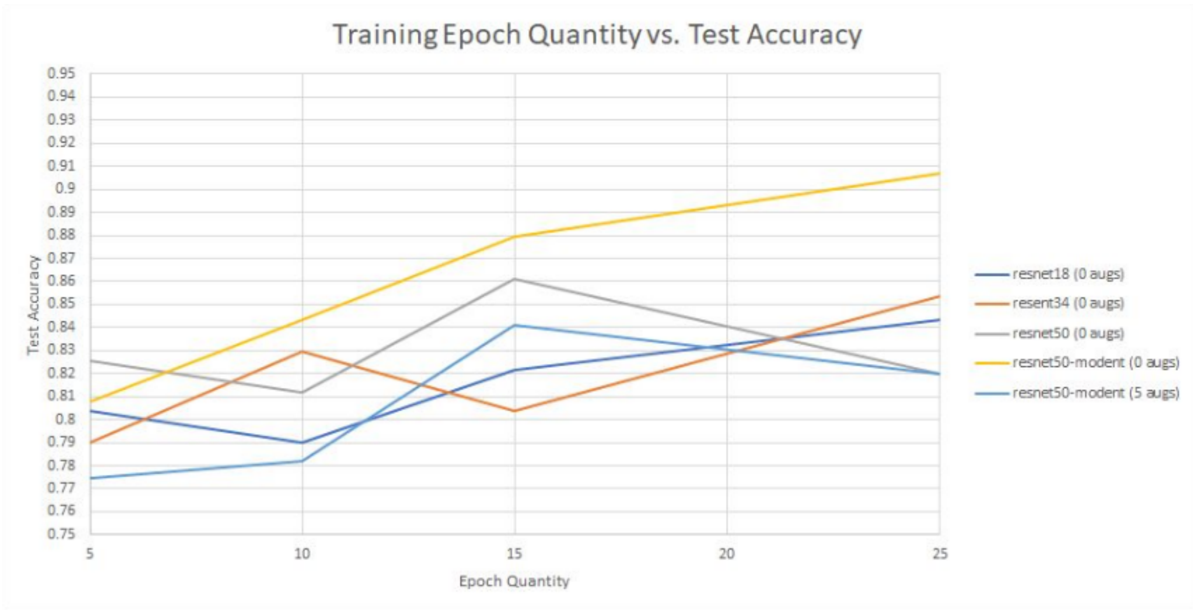

Figure 4: Training Epochs vs. Testing Accuracy

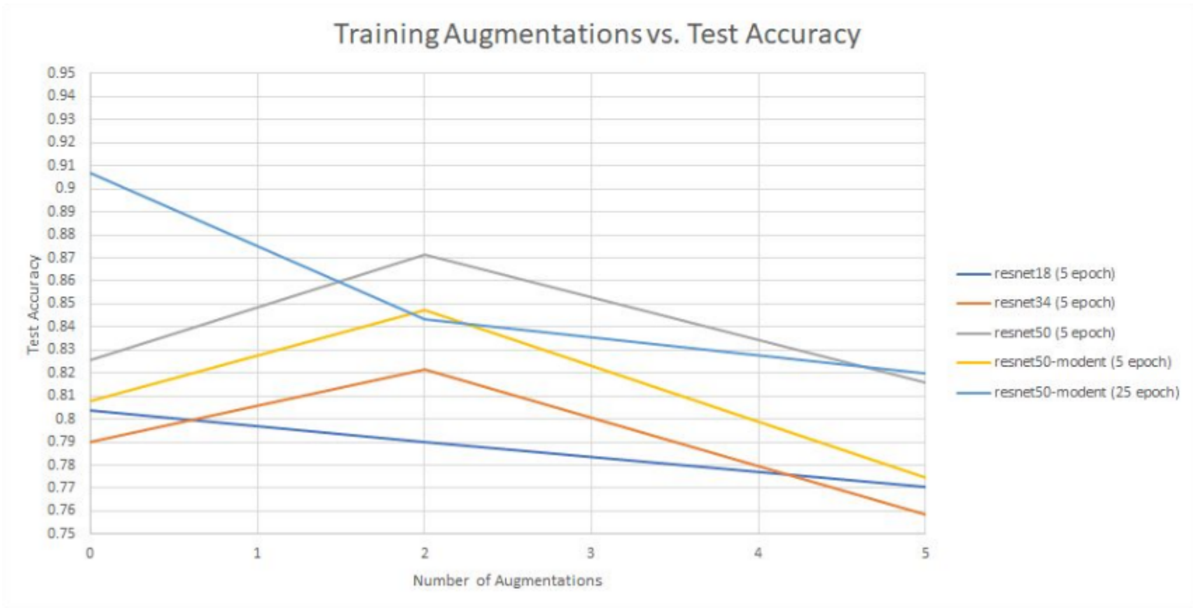

Figure 5: Training Augmentation Quantity vs. Testing Accuracy

epochs to ensure that the plateau or decrease in accuracy isn't just a localized phenomenon. This would allow for optimal epoch selection without manual selection which could greatly improve overall accuracy while also limiting the effective number of hyperparameters at play.

The counter-intuitive behavior presented by the data augmentation meta-analysis in this study also provides another exciting avenue of future work. While the aforementioned epoch optimization may also help shed some light and limit this search space there is plenty left unanswered here. Additional trials with differing combinations of augmentations could prove very enlightening as to why some augmentations may be useful and others may not. Additional image transformations may also be added in order to not only further address this question, but hopefully also increase the overall inference accuracy of the networks in use.

As with most machine learning applications a larger wealth of data would likely help improve all of these networks performance. In the world of image classification 2500 images is not generally considered a large dataset. Given a large enough data set, it may become possible to train a fully custom ResNet or some other architecture that, while more computationally expensive, may prove more accurate. The current dataset size even with all augmentations available would still not generally be considered large enough to train a Deep Convolutional Neural Net from scratch.

It would also be interesting to run additional experiments leveraging transfer learning on other prebuilt models to gather a good understanding of how effective the underlying learned features are at classifying various recycleable materials. This could be done without an expanded dataset and like many of these suggested scopes of future work would not require extensive developer time or efforts, simply compute resources.

References

[1] cchangcs. Garbage classification, 2019. 6 classes: cardboard, glass, metal , paper, plastic and trash., `https://www.kaggle.com/asdasdasasdas/garbage-classification`.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Appendix

The data necessary to replicate these results and conduct further experiments in this space can be found simply by searching the URL mentioned in [1].

The code generated for this study and these experiments can be found here on GitHub.

Data was recorded and collected using this google sheet.