Waste Classification using Image Processing with CNN's

Pranay Kumar Verma
Syracuse University
prverma@syr.edu

ABSTRACT

Disposing off waste in an effective and environmental friendly manner is one of the primary concerns in the modern world. There is a continuous search to look for different ways in order to reduce the waste foot print on the world. As the amount of waste goes up everyday with more and more online shopping sites coming into limelight causing alot of waste when it comes to things like cardboard boxes and packaging material, the number of resources needed to effectively segregate waste grows. Since there are mainly two categories of waste namely Organic and Recyclable, this paper compares two different model architectures on a dataset of about 25,000 images which would be effectively able to classify waste using different image processing techniques and recent breakthroughs in the use of CNN's for image classification problems. After using different modifications of the VGG model architecture and a pretrained model *DenseNet 121* with some fully connected layers tacked on to the end, their performances were compared and the best performing model was able to yield an accuracy of 94%.

From there on the next focus was to be able to further segregate the different classes of recyclable waste. A dataset of about 2600 images classified into glass, metal, paper etc. was used to develop a simple yet efficient model using Convolutional layers, but using a variety of preprocessing techniques to augment the images to make up for the lack of a big dataset.

I. INTRODUCTION

"The world's population is growing by 1.10 percent per year, or approximately an additional 83 million people annually."[12] With this increase in population alot more stress has been put on the earth's resources to feed all these people and deal with the waste produced. "Total annual Solid waste generation in the U.S. has increased by 77% since 1980, to 268 million tons per year."[10]

"According to the EPA, of the 267.8 million tons of municipal solid waste generated by Americans in 2017, only 94.2 million tons were recycled or composted."[13]. A large amount of workforce as well as time is needed to segregate waste, but even with all the time and workforce human error can often come into play and lead to misclassification of trash.

Below is a pictorial representation of what makes up the trash in the US.

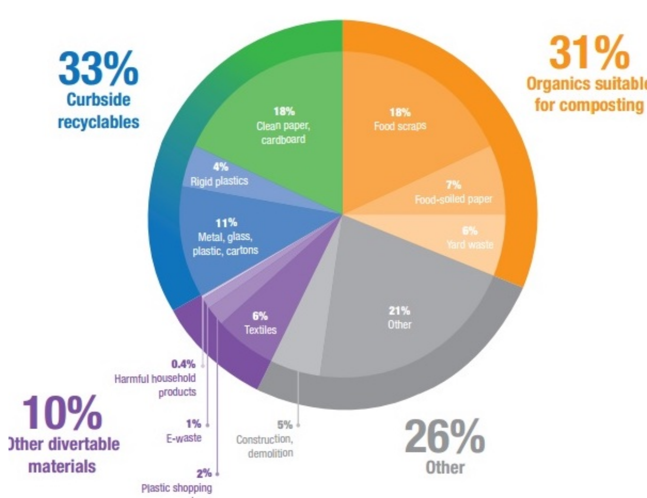

Figure 1: What's in Our Trash[8]

The efforts made to classify images with a respectable accuracy were not easy due to the following reasons.

- Using two different datasets to build a two stage model required research and it was tough to obtain respectable accuracy when classifying recyclable material further due to a very small dataset.
- Image processing tasks are processor intensive and often need alot of computational power. Running complex models like DenseNet were very time consuming and needed a GPU to be able to perform at its best.

The next section talks about more about the dataset alongwith the problem statement. This is followed by an indepth look into the models used (VGG, DenseNet), followed by the discussion of the results obtained ending with the conclusion and future implications.

II. Problem and Data Description

The goal of this project is to develop a system that can at first classify the trash as recyclable or organic. After the trash has been classified as recyclable then another model will be developed which will classify the trash into one of the 5 subcategories of recyclable trash namely cardboard, glass, metal, paper and plastic.
Two separate datasets were used for this purpose. The first dataset was comprised of about 25,000 images split into organic or recyclable. The images were all of mixed sizes with the largest being 500x500. Below is what the ditribution of these images looks like across the two classes.

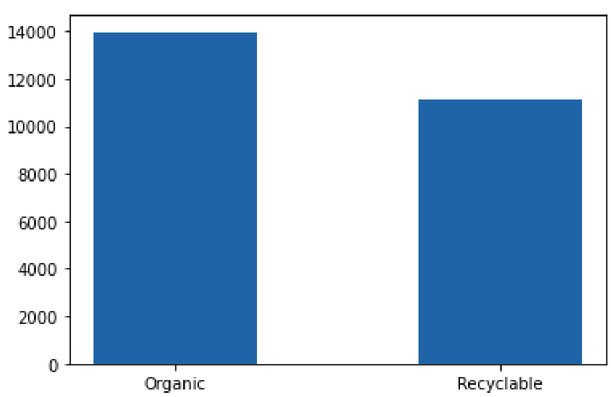

Figure 2: Data Distribution

Below are a few sample images from the dataset:

Figure 3: Organic

Figure 4: Recyclable

The second dataset used contained about 2500 images spread across 6 categories. The image sizes were 512 X 384. The distribution of the images among the classes looks like :

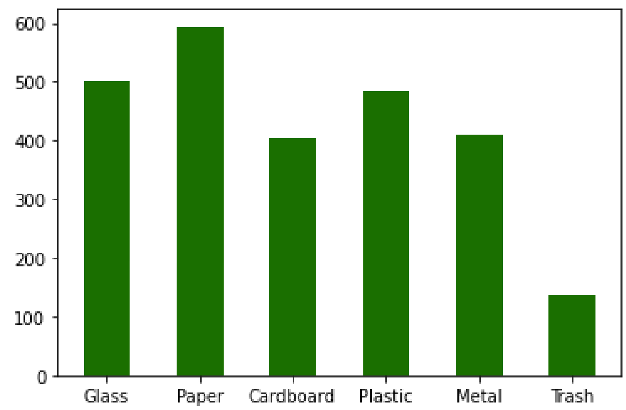

Figure 5: Data Distribution

Some images from the dataset belonging to different classes are shown below :

Figure 6: Sample Images

III. Approach and Algorithms Used

A. Data Preprocessing

The dataset was made up of images of mixed dimensions. Firstly, the images were resized using the open cv library which is a powerhouse for image processing applications. In addition to providing the algorithm with a fixed size image it also greatly reduces the computational complexity which enables the training process to be faster. “Pixel values are often unsigned integers in the range between 0 and 255. Although these pixel values can be presented directly to neural network models in their raw format, this can result in challenges during modeling, such as in the slower than expected training of the model.”[5] Once resized the image array is normalized to get the values in between 0 and 1, this is accomplished by dividing the array values by 255 which corresponds to the values each pixel can take.

After the images were resized and normalized, a concept called Image augmentation was applied in order to increase the size of the data which gives the model a larger number and variety of images to learn from. “Image augmentation is a technique of applying different transformations to original images which results in multiple transformed copies of the same image. Each copy, however, is different from the other in certain aspects depending on the augmentation techniques you apply like shifting, rotating, flipping, etc.”[6]

B. Basic Idea

There were 2 different models used to classify the garbage as recyclable or organic. One was a relatively simple network made up of Convolutional and MaxPool layers and the other was based of a pre-trained transfer learning model called DenseNet121. For classifying the recyclable items further into subclasses a VGG inspired CNN was used.

1) Simple CNN: As a starting point and baseline at first a simple CNN was developed in order to classify images. To start off with the network just consisted of a couple of convolutional layers followed by maxpooling layers. “Convolutional layers in a convolutional neural network systematically apply learned filters to input images in order to create feature maps that summarize the presence of those features in the input.”[4] The MaxPooling layer helps perform dimensionality reduction while still retaining all of the original information. This was then followed by a flatten layer whose job is to convert the multi-dimensional data into a 1-D vector for easier computation. The last two layers are fully connected layers, the last one having 2 nodes each corresponding to one of the classes in the dataset (*recyclable , organic*).

2) DenseNet121: In transfer learning, we apply the knowledge of a domain that was learnt in another application to our problem set, hoping to replicate the performance on our model. “Transfer learning is performed by taking a model that was trained on a huge set of images, and making use of the pre-trained weights in another dataset to leverage the performance of the trained system. This is done as image classification is computationally very expensive and requires a large enough dataset to yield optimal performance. ”[7]

DenseNet is a very powerful model with 121 layers. With torchvision.models these pre-trained networks can be downloaded and used in applications. DenseNets are comprised of smaller *DenseBlocks*, “the dimensions of the feature maps remains constant within a block, but the number of filters changes between them. The in between layers are called as Transition Layers. These layers are responsible for the downsampling applying a batch normalization, a 1x1 convolution and a 2x2 pooling layers.”[14]

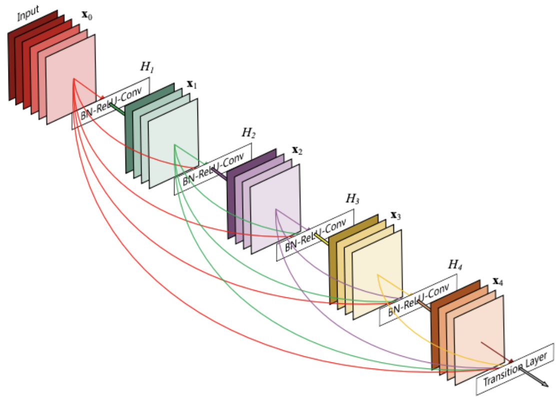

Figure 7: DenseNet Architecture

"Dense Convolutional Network (DenseNet), connects each layer to every other layer in a feed-forward fashion. The 1-crop error rates on the imagenet dataset with the pretrained model are 25.35 for top-1 error and 7.83 for top-5 error."[11]

3) VGG inspired CNN: "VGG is an innovative object-recognition model that supports up to 19 layers.".[15] The model was built taking the VGG architecture as an inspiration but shrinking down the model size due to computational constrains. "The convolutional layers in VGG use a very small receptive field (3x3, the smallest possible size that still captures left/right and up/down). There are also 1x1 convolution filters which act as a linear transformation of the input, which is followed by a ReLU unit."[15] The model consisted of a pair of convolutional and max pool layers repeated a total of 4 times followed by a flatten layer to transform the data to 1-D vector. At the end, the model has a fully connected layer followed by an output layer of size 6 corresponding to the different classes.

C. Experiments - Hyper-parameter tuning

This section will discuss more about the model architectures and how various hyperparameters were chosen to get expected results.

1) Simple CNN: To determine how many layers and the size of the kernel to use for the CNN architecture the below approach was followed :

- To start off, a very simple model was developed with just one convolutional layer, followed by a max pool layer.
- From there on more convolutional and max pool layers were added in succession gauging the performance benefits as well as the extra computational effort needed for a total of 10 epochs.
- Once the performance gains were not worth the extra computational effort the number of convolutional and max pool layers were fixed
- Then at the end a fully connected layer was added, two combinations of the layer was tried having different number of neurons.

2) DenseNet121: Since DenseNet121 is a pretrained model which leverages the concept of Transfer learning, there are not many changes we can make to the pre trained model. However, adding the final layers onto the model is upto the application and this is where we can experiment with different number of layers or different activation functions to see which yields the best result. To start of, a linear transformation layer was added, which would apply a linear transform reducing the number of features simultaneously. The number of features were reduced to half by the first linear layer. In between the linear layer and output layer a ReLU activation layer was added followed by a dropout layer. "Dropout is a regularization technique that 'drops out' or 'deactivates' few neurons in the neural network randomly in order to avoid the problem of overfitting."[3].

After experimenting with different output features for the linear layer and the dropout layer, the model which performed the best was chosen. Also a relatively low learning rate was chosen to avoid the problem of overfitting as the model would never see a similar image in the real world while classifying trash because of varying surroundings and other variables.

3) VGG inspired CNN: The VGG model consits of a total of 16 layers, "The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014." [9] Keeping the basic 4 separate layer architecture a model was developed with a combination of convolutional and max pooling layers repeated over 4 times to replicate the basic idea of *VGG 16*. The ReLU activation function is applied to each of the convolutional layers.

After exprimenting with different combinations of the number of filters in each convolutional layer the final chosen combination that performed the best was 32:64:32:32.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 248, 248, 32)      896
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 124, 124, 32)      0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 122, 122, 64)      18496
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 61, 61, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 59, 59, 32)        18464
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 29, 29, 32)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 27, 27, 32)        9248
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 13, 13, 32)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 5408)              0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               692352
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 774
=================================================================
Total params: 740,230
Trainable params: 740,230
Non-trainable params: 0
```

Figure 8: Model Architecture

IV. Evaluation Results

This section will provide more insight to the results obtained by the above discussed models. The results will be compared both numerically and graphically, also looking into the computational complexity aspect of each of the models. The numerical statistics mentioned are the average across a total of 5 runs per model.

A. *Simple CNN*

The simple CNN model performed reasonably well for the size of the model but was not able to go above the 85% mark. The loss value which was minimized was *categorical_crossentropy* the best obtained can be seen in the table.

Metric	Value
Accuracy	84%
Loss	0.29

Table I: Simple CNN Results

Since the model was relatively simple and did not contain a large number of layers or neurons it did not require alot of computational power and was able to get through 10 epochs in about 10 minutes running on Google Colab which provides GPU's to enable faster processing.

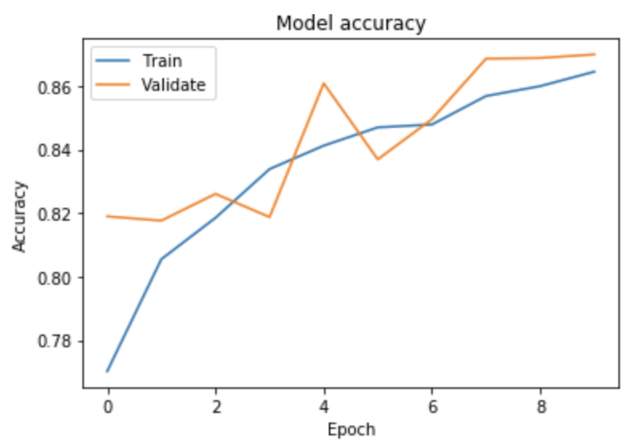

Figure 9: Model Accuracy vs Epochs

B. *DenseNet121*

The DenseNet model though took a long time to train was able to yield impressive results getting an accuracy of almost 95% with 98% predicted correctly for the organic class.

Metric	Value
Accuracy	94.9%
Loss	0.194

Table II: DenseNet Results

Below is what the model accuracy looks like through the training process.

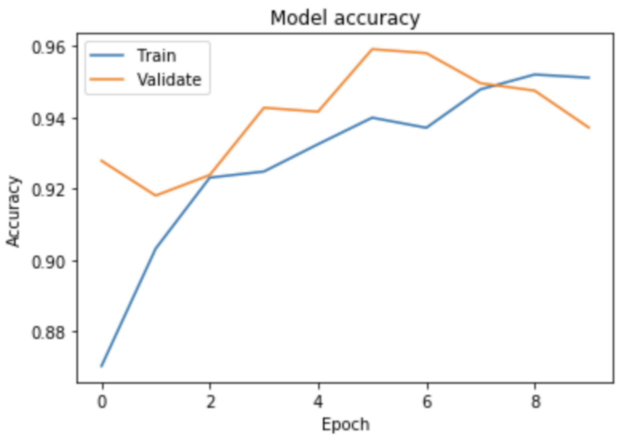

Figure 10: Model Accuracy vs Epochs

Apart from the accuracy and loss another critical aspect is the computational complexity of the algorithm i.e. the amount of time and resources it needs to run. Below we can see the time taken by the forementioned models across 5 different runs.

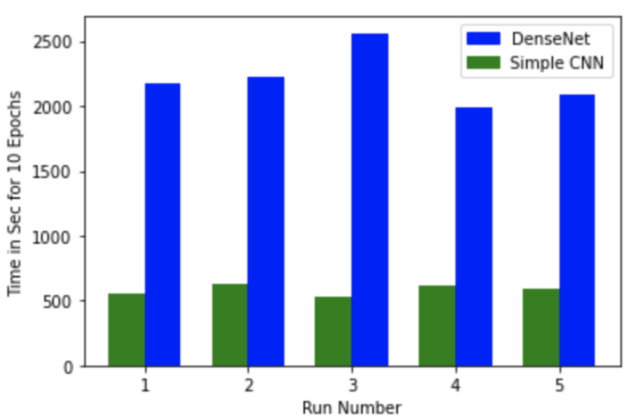

Figure 11: Time Taken for 10 Epochs Across Multiple Runs

As we can see the DenseNet model takes approximately 5 times as long as the simple CNN to execute. Althought this is quite a significant gap, the extra computational effort needed shines through in the results as DenseNet greatly out performs the simple Naive CNN model.

C. VGG inspired CNN

The lack of a large number of labelled samples made it really hard to get good results. Even though the model obtained a high accuracy on the training data, on the validation set it was able to get only about 75%

Metric	Value
Accuracy	73%
Loss	0.77

Table III: VGG Inspired CNN Results

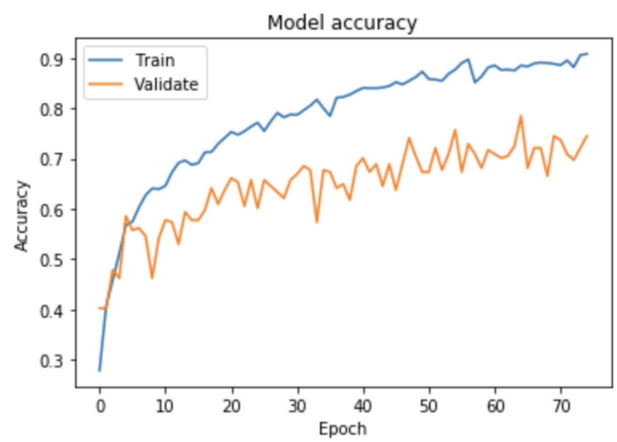

Figure 12: Model Accuracy vs Epochs

V. Future Scope

There is a scope to extend the current code to improve the accuracy as well as the efficiency of the current models. Another good to have feature would be to create a small User Interface which would popup and the user can upload images and see results in real time after the model has been trained. In the future this model can also be adopted to recognize images in real time video and help with autonomous trash collection.

VI. Conclusion

Through the course of the project a number of different models were used and their results compared and contrasted. The notable ones were mentioned in the paper above. The DenseNet model which is a new breakthrough development in the field of image processing greatly outperformed a traditional convolutional model. Using DenseNet, waste was able to be segregated as organic or recyclable with upto 95% accuracy. Due to the lack of a large number of training data for the subclasses of recyclable waste the VGG inspired CNN model was able to achieve an accuracy of about 73% which can act as a baseline for further studies and developments.

REFERENCES

[1] Color to Grayscale: Does the Method Matter in Image Recognition? : https://tinyurl.com/y63hylae, Referred 10-25-2020.

[2] Understanding Convolutions and Pooling in Neural Networks : https://towardsdatascience.com/understanding-convolutions-and-pooling-in-neural-networks-a-simple-explanation-885a2d78f211, Referred 11-04-2020.

[3] Batch normalization and dropout in neural networks with pytorch. https://towardsdatascience.com/batch-normalization-and-dropout-in-neural-networks-explained-with-pytorch-47d7a8459bcd , downloaded on 12-2-2020.

[4] A gentle introduction to pooling layers for convolutional neural networks. https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/, downloaded on 12-3-2020.

[5] How to manually scale image pixel data for deep learning. https://machinelearningmastery.com/how-to-manually-scale-image-pixel-data-for-deep-learning/, downloaded on 12-1-2020.

[6] Image augmentation on the fly using keras imagedatagenerator! https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/, downloaded on 12-2-2020.

[7] A introduction to transfer learning for deep learning. https://machinelearningmastery.com/transfer-learning-for-deep-learning/, downloaded on 12-5-2020.

[8] NYC – commercial organics recycling mandate. https://ilsr.org/rule/food-scrap-ban/nyc-organics-recovery/, downloaded on 11-28-2020.

[9] VGG16 – convolutional network for classification and detection. https://neurohive.io/en/popular-networks/vgg16/ , downloaded on 12-2-2020.

[10] Municipal solid waste factsheet. http://css.umich.edu/factsheets/municipal-solid-waste-factsheet, downloaded on 11-29-2020.

[11] Pytorch official densenet documentation. https://pytorch.org/hub/pytorch_vision_densenet/, downloaded on 12-4-2020.

[12] Quick facts about the world's population. https://www.compassion.com/poverty/population.htm, downloaded on 11-28-2020.

[13] Recycling in the u.s. is broken. how do we fix it? https://blogs.ei.columbia.edu/2020/03/13/fix-recycling-america/, downloaded on 11-29-2020.

[14] Understanding and visualizing densenets. https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a, downloaded on 12-4-2020.

[15] Vgg neural networks: The next step after alexnet. https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c, downloaded on 12-5-2020.

[16] Xia Li, Xi Shen, Yongxia Zhou, Xiuhui Wang, and Tie-Qiang Li. Classification of breast cancer histopathological images using interleaved densenet with senet (idsnet). *PloS one*, 15(5), Referred 11-10-2020.

[17] Mindy Yang and Gary Thung. Classification of trash for recyclability status. Referred 10-20-2020.

VII. APPENDIX

After the initial code submission, work has been done to try and improve the model responsible for further classifying the recyclable trash as glass, paper etc. Since the earlier model did not perform well enough, a full VGG 16 network was implemented having a total of 16 layers consisting of various convolutional layers with batch normalization done at various stages. Below is a pictorial representation of the VGG 16 architecture.

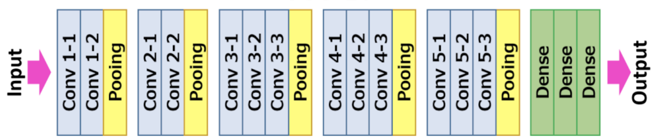

Figure 13: Model Architecture [9]

Although the model was much more complex as compared to the model discussed previously it did not perform too well. This can be attributed to the lack of a big dataset, with only about 500 images per class it is difficult to get a high accuracy.

Metric	Value
Accuracy	77%
Loss	0.69

Table IV: VGG16 Results

Below is what the accuracy looked like as the number of epochs increased.

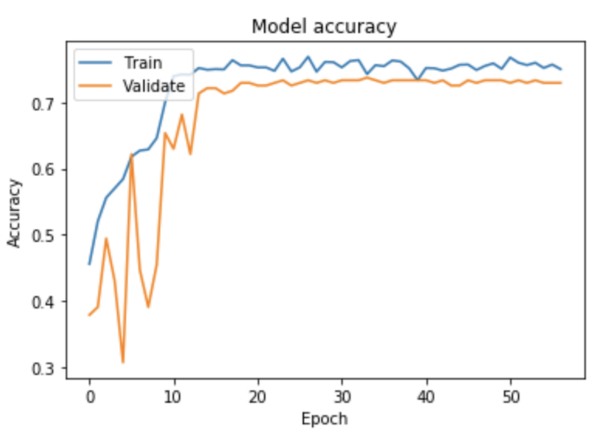

Figure 14: Model Accuracy vs Epochs