CS230

# OscarNet: Using Transfer Learning to Classify Disposable Waste

**John Knowles**
Department of Computer Science
Stanford University
jkn0wles@stanford.edu

**Sam Kennedy**
Department of Computer Science
Stanford University
sjkenned@stanford.edu

**Tom Kennedy**
Department of Computer Science
Stanford University
tkenned9@stanford.edu

## Abstract

Our group was interested to explore how to utilize Neural Networks in situations where a small amount of data is available. We saw the potential for this application after learning about Transfer Learning, where we could leverage a large network pre-trained on large amounts of data, and optimize this network based on whatever data was available. Rather than training a network to solve a specific problem, we wanted to take a generalized solution and apply it to a specific domain. We saw potential for this to apply in the domains of text, speech and images, and ultimately decided to pursue an image classification task. To complete this task, we utilized a large CNN network pre-trained for the ImageNet task. We removed the FC classification layers, and add our own network with 1 hidden dense layer to classify images of disposable waste into 7 categories. After training for 150 epochs, we achieved a validation accuracy of 88.42 percent.

## 1 Introduction

The fundamental problem motivating this project was the mis-sorting of disposable waste into Compost, Recycling and Trash bins. People currently have tools in place to help them sort their waste, like labels on items and waste bins, but the waste is still mis-sorted. As a way to "fix" the problem, we thought of a mobile application, where by pointing a phone at an object, the app would apply a label to the object to indicate where it should be disposed. This app would use a neural network to make its classification. Thus, we wanted to pursue making a neural network capable of classifying small objects as waste. Additionally, we wanted to explore Transfer Learning, and this problem provided a small-data scenario with which we could explore.

In solving this problem, we built a neural network. CNN's have typically been successful at image classification tasks, so we chose to use a CNN for our transfer learning task. The input to our network is an image. We then use a neural network to output a predicted class of waste (paper, plastic, metal, glass, cardboard, trash, non-waste).

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

## 2 Related work

The VGG-19 architecture was developed by Karen Simonyan and Andrew Zisserman [1]. It represents the state-of-the-art for well tested, general-purpose image classification. They showed that an extra-deep network with many convolutional layers (interspersed with max-pooling layers) could classify images with unprecedented accuracy. We were inspired by VGGNet's success on [x] to utlilize not only its architecture, but the weights of the primary layers through transfer learning. (see methods section). VGG-19 has been demonstrated as an effective source for transfer learning.

Mindy Yang and Gary Thung [3] built a CNN for recycling classification that provided useful insights to us. We built our model in part based off an analysis of where theirs had fallen short. Yang and Thung note that "[their] CNN was not trained to its full capability due to trouble finding optimal hyperparameters". Hoping to avoid this difficulty, we decided to leverage the pretrained weights of the VGG-16 network rather than training from scratch. They discussed planning on using transfer learning with VGG-19 in the "future" section of their paper. This approach was comparatively successful. We also decided to add a non-waste category, allowing the network to classify objects that weren't waste.

Building off Yang and Thung's work, another team [4] utilized faster r-cnn pretrained on the PASCAL VOLC dataset. [5] Ultimately, we considered their architecture but used VGG-19 because it has proven sufficient for similar tasks in the past, and we wanted to devote our efforts to tuning and analysis rather than novel architecture. Also, the primary benefit of their faster r-cnn architecture seemed to be training speed, which would have been redundant for us because our access to GPU already met our computational needs.

## 3 Dataset and Features

Out dataset consists of 2498 training examples, and 147 validation examples. We did not generate a test set during the training of the network, but have begun to collect a test set. See the Future section for more on this. We did not apply any preprocessing to the images, but instead used data augmentation to increase our training set size. We applied shear, rotation, zoom and shifts. The images in the dataset have a resolution of 512 x 384, and are reshaped to 224 x 224 x 3 before they are inputted to the network.

In considering datasets, we searched online for datasets for inspiration. We planned on generating a hand-labeled dataset ourselves, consisting of around 5000 images, but found a dataset of up-close photos of different categories of waste taken with iPhone cameras, which was exactly the distribution we were looking for. The dataset consisted of 2527 images split into 6 categories. The dataset was shot and labeled by the team of Gary Thung and Mindy Yang. Examples from the dataset are shown below:



We also used created a non-waste objects dataset from PASCAL VOC 2012 [5] and the Flowers dataset by Visual Geometry Group at the University of Oxford [7].

The VGG19 network extracts a 7 x 7 x 512 feature map from the convolutions applied over various filters across the convolutional blocks. These features are extracted by convolving filters first across

the image input, and then across the output of other convolutions on the image as the network gets deeper. These features are used as "encodings" by the transfer network, which maps from the feature map to a class label output.

## 4 Methods

Under the Transfer learning paradigm, we use a combination of two neural networks. Starting with the VGG19 network with weights trained on ImageNet, we remove the fully connected layers at the top of the network, and connect our transfer network. The input to the VGG19 network is 224 x 224 x 3 RGB images, and the VGG19 is comprised of convolutional-blocks [1], which are several convolutional layers followed by a max-pooling layer. These blocks differ in their size and number of filters. This network is trained to map from the image input to a softmax output of 1000 classes. We remove the classification FC layers, and the amended network maps from image inputs to a 7 x 7 x 512 feature map from convolutions.

We combine this with our transfer network, which is comprised of a 256 Neuron Dense Fully-Connected layer with a ReLu activation, a Dropout layer, a Batch Normalization layer, and a Dense output layer with a softmax activation layer. To implement our network, we rely on the regularizing effects of the Dropout and BatchNorm layers, which "shut off" neurons with a fixed probability and also normalizes the weights relative to their mean and standard deviation.

During training, we calculate the loss using the categorical crossentropy function, shown below:

$$CCE = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{J} y_j \cdot log(\hat{y}_j) + (1 - y_j) \cdot log(1 - \hat{y}_j)$$

Additionally, during backpropagation, we use the Adam optimizer, which is a descendent from both AdaGrad and RMSProp. By using moment analysis, this optimizer is able to tune how fast it should learn, which allows for more control over the training period. The formula for the Adam optimizer is shown below:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \qquad \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t.$$

## 5 Experiments/Results/Discussion

We performed a thorough search over hyperparameters, and found that a larger mini-batch led to better results in shorter "time" (in terms of epochs). However, the smaller batches ran quicker in actual time. We decided to balance speed of training with a larger batch size and chose a mini-batch size of 32. For our learning rate, we saw that a higher learning rate led to faster, but more variable training. We fonud that learning rates between .001 and .0001 returned good results. We decided to use different learning rates based on what stage of training we were at, as a way to control the rate of training. For our epsilon, we saw that 0 was unusable, and found .5 to be the best parameter.

We began training using just a network with two dense layers, a hidden layer and an output layer. We found that this network architecture tended to quickly overfit, and to address this, we added a Dropout and a Batch normalization layer to the network. Similar to all other hyperparameters, we tuned the dropout rate before settling on .6.

In order to train the network, we used three training periods: short, medium and long periods, in order to apply different hyperparameter conditions. During the first short training session, we used no L2 regularization and a learning rate of .001 to make the initial steps on the network. We trained

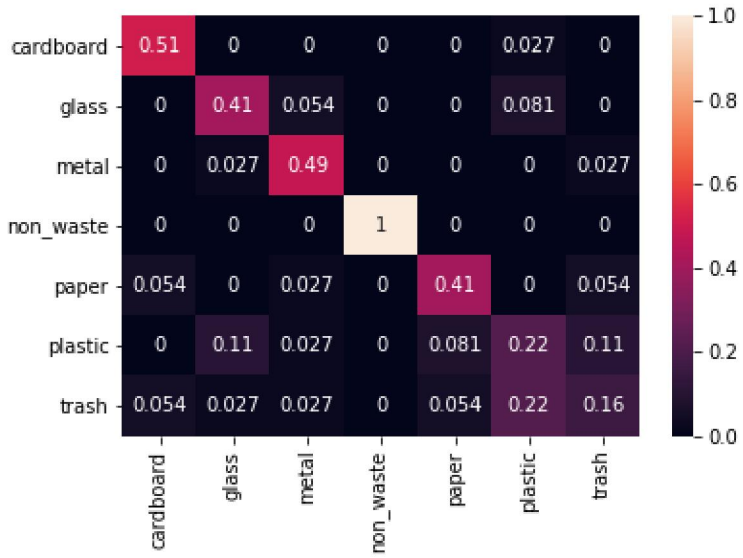
this for 10 epochs, achieving modest results.During the medium training period, we used a higher learning rate of .005 and added L2 regularization with a parameter of .01. We train for 50 epochs. In this training period, we begin to retrain the final layers of the VGG19 network. During this period, we retrain the final conv-block. The final training period consists of 100 epochs with a low learning rate of .0001 and an increased regularization parameter. We found that this helped to increase our validation accuracy, which reduced our variance.

Results for training are presented in the table below:

| | Training Loss | Validation Loss | Training Accuracy | Validation Accuracy | AUC |
|---|---|---|---|---|---|
| 15 Epochs | 5.2889 | 4.9947 | 50.08% | 49.68% | 14 |
| 65 Epochs | 1.1599 | 1.7571 | 98.62% | 81.8% | 18.5 |
| 163 Epochs | 0.8578 | 1.4382 | 99.78% | 88.42% | 19 |

Based on these results, we believe that we have overfit to the dataset. However, we believe that by implementing early stopping using a different metric, like validation loss, we could reduce overfitting.

In order to analyze our results, we used a confusion matrix, shown below.




It became clear that because out non-waste dataset came from a different distribution, that it was clearly being classified. In considering the strengths of other classes, trash and plastic were particularly weak, with the highest mislabeling rates. An analysis of the images revealed similar shapes between plastics and glasses, and similar textures and colors between plastics and trash. In order to account for this, we decided to implement a color-normalization augmentation to the training data, but did not have time to fully retrain.

## 6 Conclusion/Future Work

In this project, we successfully used Transfer Learning from the VGG19 network trained on ImageNet to classify images of waste. We found that a shallow neural network performed best as the transfer network. Using a network with only one hidden layer with 256 neurons, we were able to achieve above 80 percent accuracy on a seven class image classification task. We believe that this shallow network architecture performed well because it was tasked with decoding the feature map outputted from the VGG19 network, which does not require a deep network, as the images have already undergone feature extraction.

In the future, we would like to explore an “ensemble” style network, which would feed the features of multiple large CNNs into a single “transfer” network, which could be trained simultaneously by feeding the images to each CNN and then concatenating the results, which would serve as the encoding period, of images to feature maps. Then, using these concatenated feature maps, the “transfer” network would serve as the decoder, producing output labels from the encodings. This could be further optimized by ideal combinations or reductions of the feature maps before decoding.

Additionally, we are interested in other projects of similar scales, where data either is scarce, or must be self-produced. Using the tool of mobile phones, data can be gathered, either through the camera or the microphone. Transfer learning allows for networks to be built in situations of low data, as these networks are trained on big amounts of data, and are able to generalize. Thus, we are interested in future exploration around refining the process from ideas in this domain to building a functional network.

## 7 Contributions

Tom Kennedy: Project Research, AWS, and Hyperparameter search

Sam Kennedy: Project Research and Data Acquisition/Processing

John Knowles: Project Design, Training and results compilation, Poster, Report

## References


[1] Simonyan, and Zisserman. "Very Deep Convolutional Networks for Large-scale image recognition." arXiv:1409.1556 (2014).

[2] Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun. "Deep Residual Learning for Image Recognition" CoRR, 2015.

[3] Thung, and Yang. "Classification of Trash for Recyclability Status.” CS229 Project Report 2016 (2016).

[4]Oluwasanya Awe, Robel Mengistu, Vikram Sreedhar. Final Report: Smart Trash Net: Waste Localization and Classification. CS229 Project Report 2017 (2016)

[5] Everingham, Gool, Williams, Winn, and Zisserman. “The Pascal Visual Object Classes Challenge” International Journal of Computer Vision, 303–338, 2010.

[6] Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun. "Deep Residual Learning for Image Recognition" CoRR, 2015.

[7] Maria-Elena Nilsback and Andrew Zisserman. "102 Category Flower Dataset". http://www.robots.ox.ac.uk/ vgg/data/flowers/102/index.html

[8] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[9] Guido van Rossum: Python Library Reference. May 1995. CWI Report CS-R9524.

[10] J.D. Hunter. Matplotlib. Computing in Science and Engineering (Volume: 9, Issue: 3, May-June 2007). 2007.

[11] Travis E, Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006).

[12] Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

[13] Francis Chollet. Keras. 2015.