CS230

Trash Classification Using Convolutional Neural Networks
Project Category: Computer Vision

Yujie He
ICME
Stanford University
yujiehe@stanford.edu

Qinyue Gu
ICME
Stanford University
gqy94@stanford.edu

Maguo Shi
ICME
Stanford University
smgyl@stanford.edu

Abstract

As the waste problem becomes increasingly eminent across the globe, we aim to provide an automated waste sorting tool to make it easier for residents to classify trash. Our project used TrashNet [2] as our dataset, and classified recyclables or trash into six categories. To achieve our objective, we focused on Convolutional Neural Networks (CNN), and explored several well-known architectures at early stages. We ended up with modified AlexNet by taking two layers out, and experimented different techniques based on this model architecture, including dropout, data augmentation and learning rate decay. We also experimented two classifiers, Softmax and Support Vector Machine (SVM), as the last layer of our model structure. The highest test accuracy we achieved was 79.94% with the model using partial data augmentation and SVM classifier.

1 Introduction

As China started a shift to make trash classification from voluntary to compulsory in 2019 [1], it indicates that trash classification has become a major issue that increasingly raises concern from countries and societies. In order to help save the environment and ease the process for the residents, our project aims to provide an automated waste sorting tool.

Though countries may have rules and regulations that are different on details, one commonly accepted principle is that recyclables should be separated from other trash. We take U.S. standards on sorting recyclables for this project and the outcome is a model that takes in an image of a single piece of waste and outputs a vector with probabilities of the six categories: cardboard, glass, metal, paper, plastic and trash.

2 Related Work

There is a previous project about trash sorting done by Gary Thung and Mindy Yang as a CS229 project in 2016 [3]. Their objective was the same as ours: to sort a single piece of garbage or recyclable into six categories consisting of cardboard, glass, metal, paper, plastic and trash. They implemented SVM and CNN separately and compared the results. The eleven-layer CNN they implemented was very similar to AlexNet, and they used 0.75 of the number of filters for some convolutional layers due to computational constraints. They also utilized techniques including learning rate decay, L2 regularization and Kaiming weight initialization. However, their result wasn't satisfying for the CNN approach. The SVM approach they implemented achieved a test accuracy of 63%, but the CNN method only achieved a test accuracy of 22%. Thung and Young believed that the hyperparameters they implemented were suboptimal, which caused the CNN inability to learn [3]. Our main focus for this project is to greatly increase the accuracy of using CNN on the task.

Many further works have been done based on the TrashNet [2] dataset, and transfer learning is one of the most popular approaches that many works choose to increase the accuracy of their models.

- Aral et al. applied transfer learning on TrashNet data with several well-known models, including Densenet121, DenseNet169, InceptionResnetV2, MobileNet and Xception [4]. The highest test accuracy they achieved was 95% with DenseNet121 using fine-tuning. [4]
- Özkaya et al. also made comparison between several fine-tuning models on TrashNet data [5]. The architectures they tested include Alexnet, VGG16, Googlenet and Resnet. They also tested two different classifiers, Softmax and SVM, as the last layer of their fine-tuned models and compared the effects of them. The highest accuracy was achieved with GoogleNet+SVM as 97.86%. We were inspired by them for changing the last layer of our model to SVM. [5]
- Vo et al. utilized transfer learning techniques and developed DNN-TC which was an improvement of ResNet model [6]. Their experimental results indicated that DNN-TC yielded 94% accuracy for TrashNet data. [6]
- Bircanoglu et al. experimented on many deep CNN models with and without pre-trained weights [7]. The best result they achieved was 95% test accuracy with DenseNet121 using transfer learning techniques. They also developed a model called RecycleNet which reduced the number of parameters from 7 million to 3 million, but the accuracy also decreased to 81%. [7]

While most of these works focused on transfer learning techniques, we modified the structure of existing architectures and trained them from scratch.

3 Dataset

We utilized the dataset that Thung and Young set up for TrashNet [2] for our project. This dataset spans six categories and consists of 2527 images in total. Detailed number of images for each class can be found in the table below. The pictures were all taken by Apple iPhones [2], and original images had a resolution of 72 pixels per inch. We used approximately 75% from each category as training data, and 25% as development data. Therefore, our training set had 1894 images in total, and our development set had 633 images in total. Detailed number of images for training set and development set can also be found in the table below.

Class Type	Number of Images in Training Set	Number of Images in Development Set	Total
Cardboard	302	101	403
Glass	376	125	501
Metal	307	103	410
Paper	445	149	594
Plastic	361	121	482
Trash	103	34	137
Total	1894	633	2527

We resized all training and test images to a size of 227x227 to best fit with our models. We incorporated data augmentation techniques including vertical flip, horizontal flip, width shift, height shift, zooming and rotation [8][9]. A samples from each class of our dataset can be found in Figure 1 on the next page.

4 Methodology

We explored and modified ResNet50 [10], VGG-11 [11][12] and AlexNet [13] architectures for Milestone #2 [14]. We used Softmax as the activation for the last fully connected layer, and compared performances of the three architectures. We found that modified AlexNet model gave the highest test accuracy and required relatively short time to train. Thus, we decided to continue exploring our modified AlexNet architecture, and developed two different models using distinct loss functions and activation functions for the last fully connected layer. Our code can be accessed here: https://github.com/yujiehe05/YH_QG_MS_Final.

The basic model architecture for both of our models is the same and is presented in Figure 2 on the next page. We first employed basic AlexNet following lecture on Coursera, and added normalization following instructions from this website [13]. We deactivated two convolutional layers containing 384 filters since we

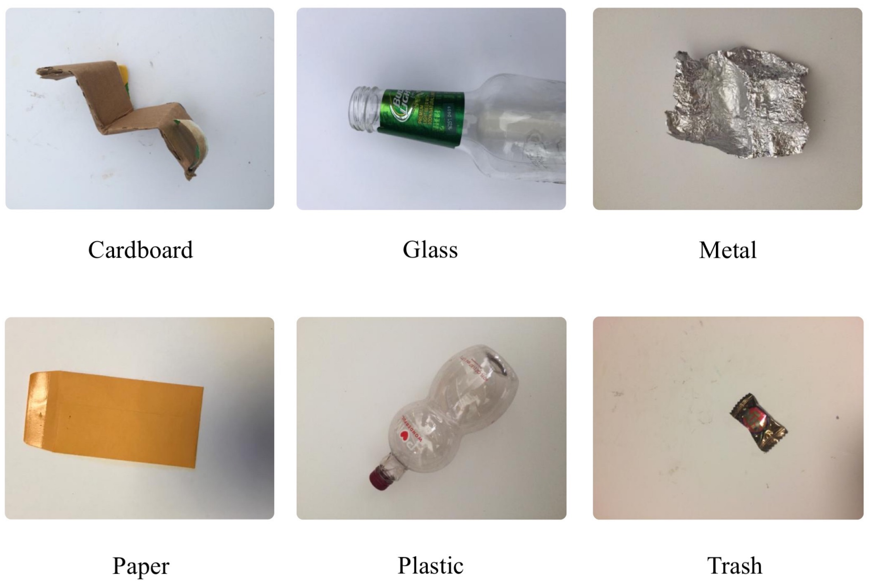

Figure 1: A Sample from Each Class

observed similar test accuracy through our training experiments with or without these two layers. Thus, we have eleven layers in total.

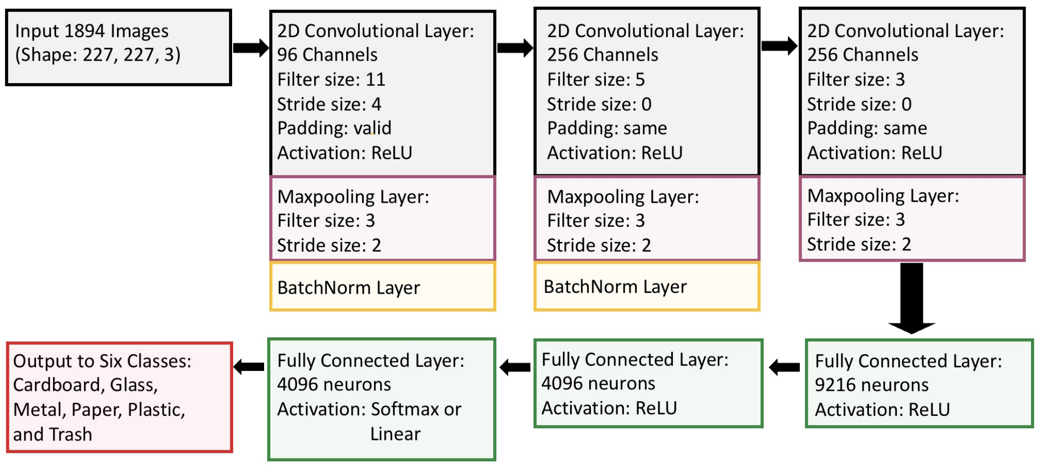

Figure 2: Basic Model Architecture

The first model we tried used Softmax as activation function for the last fully connected layer, and a categorical cross entropy loss function, whose formula is as follows [15].

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{M}(y_{ij} \times \log(\hat{y}_{ij}))$$

This was also used in Milestone #2.

The second model we tried used linear activation function for the last fully connected layer, and a categorical hinge loss function, whose formula is as follows.

$$L(y, \hat{y}) = max(y^T\hat{y} - max(1 - y) + 1, 0)$$

The combination of linear activation function and hinge loss function provides a "maximum-margin" classification [16], and therefore is very similar to SVM structure. SVM is a powerful tool while dealing with binary classification problems, but our project dealt with a multi-class classification problem. We thus used categorical hinge loss, which is an extension of hinge loss used in SVM [17].

For both models, we used Adam optimizer and a mini-batch size of 32. Besides the plain model, we tried adding dropout rate to our first and second fully connected layers. We experimented several dropout rates but did not see significant impacts on our models' performances, so we chose the default dropout rate of 0.5. We also employed data augmentation techniques introduced in Dataset section. Due to computation power limit, we could only train with data augmentation for 30 epochs. Therefore, we used a partial data augmentation technique, which was to train the model with data augmentation for 30 epochs, and continue training the model without data augmentation for another 30 epochs. The learning rate and learning rate decay schedule for each model depended on how the model converged [18]. We experimented different learning rate schedules and found that a learning rate of 0.000005 worked fine with most of our models. We only used a learning rate of 0.00001 and a decay rate of 0.00000025 with dropout, since adding dropout significantly slowed down model convergence for the first twenty epochs. The number of epochs we trained also depended on how the model converged and is shown in detail in Results and Analysis section. We assessed model convergence based on training accuracy and training loss.

5 Results and Analysis

Training and test accuracy of our two models can be found in the table below. In general, our models had test accuracy ranging from about 70% to 80%, and models adding partial data augmentation achieved slightly better results. Model using categorical hinge loss with partial data augmentation achieved the highest result of 79.94%.

Model 1 (Softmax)	Training Accuracy	Test Accuracy	Number of Epochs
Plain Model	99.63%	74.57%	40
Adding Dropout	94.56%	72.04%	80
Adding Data Augmentation	78.04%	72.20%	30
Partial Data Augmentation	99.68%	79.46%	60

Model 2 (Linear)	Training Accuracy	Test Accuracy	Number of Epochs
Plain Model	99.89%	72.67%	40
Adding Dropout	89.39%	69.04%	80
Adding Data Augmentation	74.97%	74.09%	30
Partial Data Augmentation	99.79%	79.94%	60

Convergence of training loss, training accuracy and test accuracy can be found in Figure 3 on the next page. We use "Model 1" to denote our model with categorical cross entropy loss, and "Model 2" to denote our model with categorical hinge loss.

- In the first plot, we compared training loss and training accuracy convergence of plain models. Both of their training loss converged relatively quickly and smoothly, and Method 2 had in general higher losses. The two models had similar training accuracy convergence.
- In the second plot, we compared training loss and training accuracy convergence of models with dropout rate of 0.5 added to our first and second fully connected layers. Convergence of training loss and accuracy was significantly slower than that of plain models since the dropout feature made it more difficult for the models to learn. Similar to plain models, Method 2 had higher losses. Training accuracy of Method 1 was in general higher than that of Method 2, even though they had similar increase rates. Due to computation power limit, we only trained for 80 epochs and neither of the models reached a full convergence.
- In the third plot, we compared training loss and training accuracy convergence of models with partial data augmentation. In the first 30 epochs, convergence rates of training loss and training accuracy were about the same for two models. Convergence in this period was apparently slower than that of the plain models. Behaviors of training loss and training accuracy resembled those of plain models in the second 30 epochs as we switched to plain models.

- The last plot shows convergence of test accuracy for all models. From this plot, we can see that plain models and models with partial data augmentation had relatively stable convergence, while models with dropout appeared to be less stable. Models with partial data augmentation both achieved higher final test accuracy. We therefore conclude that data augmentation could be helpful for our project. If we have more computation power to train for a longer period with data augmentation, we expect to see better result using partial data augmentation technique for both models. From this plot, we can also see that although adding dropout indeed reduced variance in our models, it did not give obvious improvement to test accuracy. Features of our images might be a possible cause for this result.

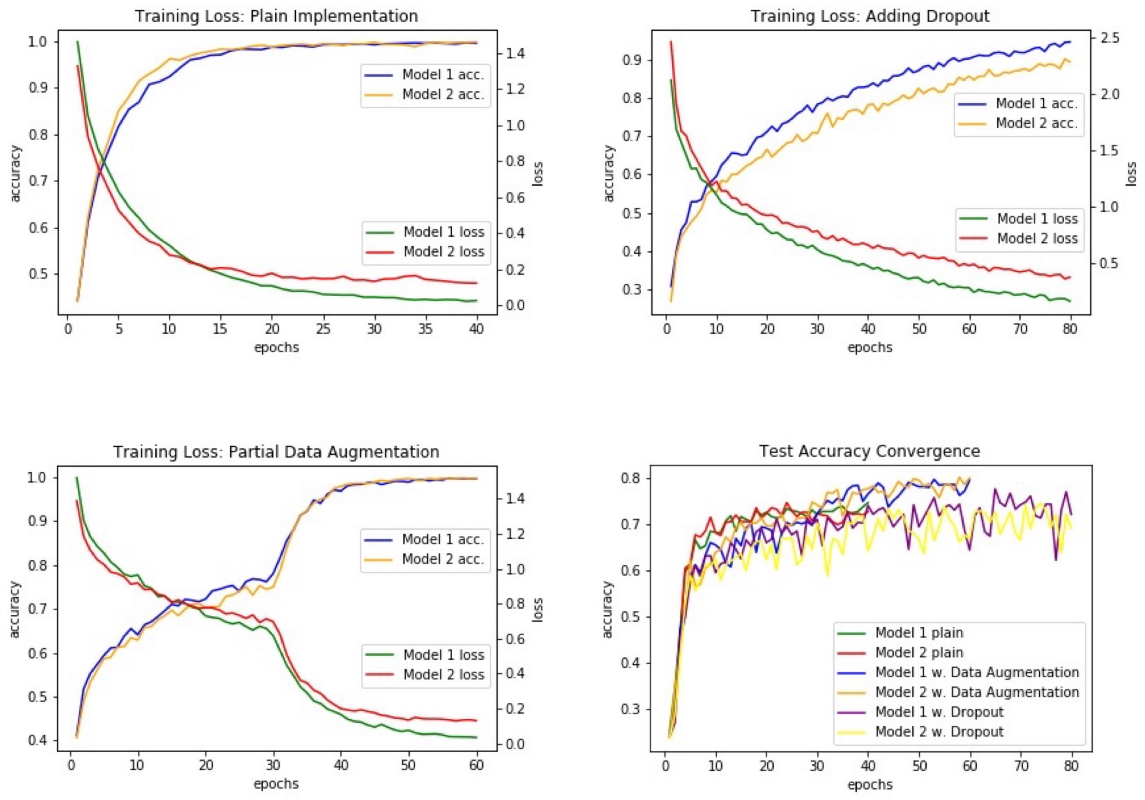

Figure 3: Convergence Comparison

6 Conclusion

We greatly increased the accuracy of using CNN to classify recyclables and trash compared to Thung and Yang's work [3]. We managed to reach stable test accuracies of 70% to 80% for our model alone or with different techniques added, including dropout, data augmentation and learning rate decay. Among all the options we explored, the highest accuracy we achieved was 79.94% with Model 2 (SVM as last layer) using partial data augmentation. However, no matter what techniques we add on, we found it extremely hard to break through the 80% threshold with our scratch model. All the accuracies higher than 90% in the related works we read about were achieved by utilizing transfer learning techniques.

In the future, we hope to explore other models further, such as VGG and ResNet, and potentially utilize transfer learning to achieve a higher accuracy. What's more, we want to upgrade this tool to classify waste according to more detailed rules for different countries such as Japan and China, when we have more data and computation power.

7 Contributions

We basically worked on everything together, discussed our thoughts throughout the quarter, and contributed quite evenly. To be a little more specific in terms of the final part, Yujie trained plain models and models with data augmentation, Qinyue trained models with dropout and learning rate decay, and Maoguo worked on SVM.

References

[1] "Shanghai Municipality's Regulation on Household Waste". *Baike*, 01 Jul. 2019, https://baike.baidu.com/item/%E4%B8%8A%E6%B5%B7%E5%B8%82%E7%94%9F%E6%B4%BB%E5%9E%83%E5%9C%BE%E7%AE%A1%E7%90%86%E6%9D%A1%E4%BE%8B/23289169?fr=aladdin. Accessed 21 Apr. 2020.

[2] G. Thung and M. Yang. "Dataset of images of trash; Torch-based CNN for garbage image classification". *GitHub Repository*, 2016, https://github.com/garythung/trashnet. Accessed 23 May 2020.

[3] G. Thung and M. Yang. "Classification of Trash for Recyclability Status". *CS229 Course Report*, 2016, http://cs229.stanford.edu/proj2016/report/ThungYang-ClassificationOfTrashForRecyclabilityStatus-report.pdf. Accessed 08 Jun. 2020.

[4] R. A. Aral, Ş. R. Keskin, M. Kaya and M. Hacıömeroğlu, "Classification of TrashNet Dataset Based on Deep Learning Models", 2018, *IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 2058-2062. Accessed 08 Jun. 2020.

[5] U. Özkaya1 and L. Seyfi. "Fine-Tuning Models Comparisons on Garbage Classification for Recyclability". *arXiv*, 07 Aug. 2019, https://arxiv.org/abs/1908.04393. Accessed 08 Jun. 2020.

[6] A. H. Vo, L. Hoang Son, M. T. Vo and T. Le, "A Novel Framework for Trash Classification Using Deep Transfer Learning," in *IEEE Access*, vol. 7, pp. 178631-178639, 2019. Accessed 08 Jun. 2020.

[7] C. Bircanoğlu, M. Atay, F. Beşer, Ö. Genç and M. A. Kızrak, "RecycleNet: Intelligent Waste Sorting Using Deep Neural Networks," *2018 Innovations in Intelligent Systems and Applications (INISTA)*, Thessaloniki, 2018, pp. 1-7, doi: 10.1109/INISTA.2018.8466276. Accessed 08 Jun. 2020.

[8] vasantvohra. "Waste Segregation Project to classify waste into different classes". *GitHub Repository*, 13 Aug. 2019, https://github.com/vasantvohra/TrashNet/blob/master/Notebooks/Trashnet%20CNN%2080%25.ipynb. Accessed 08 Jun. 2020.

[9] Shorten, C., Khoshgoftaar, T.M. "A survey on Image Data Augmentation for Deep Learning". J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0. Accessed 08 Jun. 2020.

[10] "Residual Networks". *Coursera*, https://www.coursera.org/learn/convolutional-neural-networks/notebook/jK9EQ/residual-networks. Accessed 23 May 2020.

[11] Muneeb ul Hassan. "VGG16-Convolutional Network for Classification and Detection". *Neurohive*, 20 Nov. 2018, https://neurohive.io/en/popular-networks/vgg16/. Accessed 23 May 2020.

[12] Rohit Thakur. "Step by step VGG16 implementation in Keras for beginners". *Towards Data Science*, 06 Aug. 2019, https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c. Accessed 23 May 2020.

[13] Hao Gao. "A Walk-through of AlexNet". *Medium*, 07 Aug. 2017, https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637. Accessed 23 May 2020.

[14] Siddharth Das. "CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more…". *Medium*, 16 Nov. 2017, https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5. Accessed 23 May 2020.

[15] "Categorical crossentropy". *Peltarion*, https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy. Accessed 08 Jun. 2020.

[16] "Hinge Loss". *Wikipedia*, 09 Apr. 2020, https://en.wikipedia.org/wiki/Hinge_loss. Accessed 08 Jun. 2020.

[17] Chris. "How to use categorical / multiclass hinge with Keras?". *MACHINECURVE*, 17 Oct. 2019, https://www.machinecurve.com/index.php/2019/10/17/how-to-use-categorical-multiclass-hinge-with-keras/. Accessed 08 Jun. 2020.

[18] A. Rosebrock. "Keras learning rate schedules and decay". *pyimagesearch*, 22 Jul. 2019, https://www.pyimagesearch.com/2019/07/22/keras-learning-rate-schedules-and-decay/. Accessed 08 Jun. 2020.

[19] "Formatting instructions for CS230-Winter 2018". *Overleaf, CS230 Course Material*, 2018, https://www.overleaf.com/project/5c452ffd45b91847469ef724. Accessed 21 Apr. 2020.

[20] "Bibliography management with bibtex". *Overleaf*, https://www.overleaf.com/learn/latex/bibliography_management_with_bibtex. Accessed 21 Apr. 2020.

[21] "Bibtex bibliography styles". *Overleaf*, https://www.overleaf.com/learn/latex/Bibtex_bibliography_styles. Accessed 21 Apr. 2020.

[22] "Including images on Overleaf". *Overleaf*, https://www.overleaf.com/learn/how-to/Including_images_on_Overleaf. Accessed 23 May 2020.

[23] "Inserting Images". *Overleaf*, https://www.overleaf.com/learn/latex/Inserting_Images. Accessed 08 Jun. 2020.

[24] "Tables". *Overleaf*, https://www.overleaf.com/learn/latex/tables#Captions.2C_labels_and_references. Accessed 08 Jun. 2020.

[25] "MLA Works Cited: Electronic Sources (Web Publications)". *Purdue Online writing Lab*, https://owl.purdue.edu/owl/research_and_citation/mla_style/mla_formatting_and_style_guide/mla_works_cited_electronic_sources.html. Accessed 08 May 2020.