# Brand Logo Classification using Deep Residual Learning

Adnan Abdullah*, *UFID: 7986-7725*, Nicholai Platonoff*, *UFID: 9927-3528*, Imran Nasrullah*, *UFID: 6065-9910*
*Department of ECE, University of Florida, Gainesville, FL-32611*

***Abstract*—This project aims to develop an image classification model that can accurately categorize different brand logos from given images. A state-of-the-art CNN structure, namely ResNet-50 is used for feature extraction, while three additional layers perform classification on top of that. The dataset used in this work is a combined effort of all the students taking this course. The train dataset consists of 3717 image samples from 10 different brand logos. Different data augmentation techniques are used in training to boost performance. The model performance is primarily evaluated on validation set which achieves an accuracy of 94%.**

***Index Terms*—*Logo classification, Convolution Neural Network, ResNet-50, data augmentation, Fine-tuning.***

## I. Introduction

IN this project, our group designed a convolutional neural network architecture, derived from the existing ResNet-50 architecture to classify image data from 10-brand logos. Image classification is a common problem that is tackled by the use of neural networks. Common to all image classification problems are key obstacles such as high-dimensionality data, and the associated cost in performance of reducing the dimensionality space for effective classification. However, in [1], on using ResNet-50, it's been noted that the use of pre-existing convolutional neural network models (CNN) has been a recent development in processing large volume of high dimensional data. With this in mind, our group adopted a similar ResNet-50 architecture for the specific image classification purpose of this project to minimize the training cost as well as achieve higher accuracy.

The objective of the project **Brand Logo classification using Deep Residual Learning** was to develop a method to effectively classify 10 different classes of common brand logos given a set of varying images taken at different frames, angles, and varying color intensities. To attain this objective, at first we curated a representative brand logo image set following provided instructions. After that, we modified an existing convolutional neural network-based algorithm that can efficiently recognize those collected handwritten symbols with a mean accuracy of 94% .

* This project and project report are equally contributed by all the authors.

** The authors would like to thank course instructor and TAs for their continuous guidance and support. The authors also acknowledge University of Florida Research Computing for providing computational resources and support that have contributed to the research results reported in this project. URL: http://www.rc.ufl.edu

This subsequent project report is expanded into the following sections - Related works, Implementation details, Experiments and Conclusion. In section II, we briefly reviewed the existing literature. In section III, an overview of our methodology is given relating to the adapted CNN *ResNet-50* model. After that, the experiments and the performance of the algorithm are represented in section IV. In section V, depicts critical takeaways as well as possible applications of our architecture.

## II. Related Works

With the growth of computational power, and the availability of digital data, artificial neural networks (ANN) have gained popularity in both academia and industry. Machine learning with ANN has been embraced by diverse groups of researchers. However, with the invention of GPU and TPU to parallarize computing, recent ML trend is to use other variants of ANNs such as- Convolution Neural Networks (CNN), Recurrent Neural Network (RNN) that helps to train a model efficiently and with higher accuracy. Several works have been done in the field of image classification, specifically brand logo classification. In [2], the authors detected SIFT features to classify brand logos. In [3], deep learning technique was used to identify logo region in an image and then perform classification. In [4], the authors focused on whether a logo is from alcohol brand or not, and then classify it.

## III. Implementation

### A. Dataset Description

The images were captured by participating students. Each student in the class was tasked to take 10 photos of each of the 10 classes. Each image has a dimension of 300x300x3. The full train set contains 3717 sample images. The images are vectorized, so the final dimension of the data set is (270000, 3717). The dataset had some misclassified labels, so we manually sorted and corrected them before training.

### B. Methods

The implementation of our initial experiments is trivial. From popular ML libraries like Scikit, we created some training pipelines and evaluated those models. Results from these models acted as an initial baseline for our work. The details are discussed in Section IV.

For final model, we designed a Convolution Neural Network (CNN) architecture. Deep CNN models have tons of parameters to learn. Training such a model with random weight initialization usually takes a lot of time and resources. A better approach is to import a pretrained CNN with learned weights (also called backbone) and then fine-tune it on a specific dataset. This is known as transfer learning. The pretrained weights serve as a low level feature extractor which helps the model to understand common keypoints in image such as-edges, basic geometry shapes etc. We adopted *ResNet-50* [5] as our backbone model. This model is trained on the *Imagenet* [6] dataset that contains 1000 classes. The residual connections between layers solve the gradient vanishing problem and accelerates learning. On top of this backbone, we added our own layers (known as head model) and performed training while keeping the Batch Normalization weights frozen. Freezing those weights help because we normalized our images before training. In this way, the training converges faster and less resource is needed. Our model architecture is shown in Fig. 1.

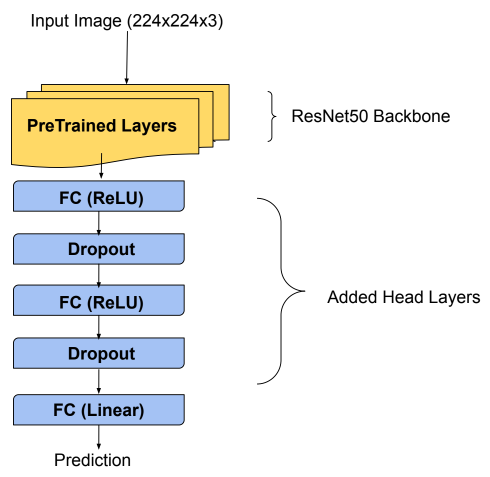

Fig. 1: Our network architecture. The detail backbone architecture can be found in [5].

### C. Training Pipeline

For training, we resized the images to 224x224x3. We also normalized the images to standard *ImageNet* mean and standard deviation since our model is pretrained on *ImageNet* dataset. We added some augmentations- resize, crop, horizontal flip, and rotation. We used Adam with learning rate = 0.001 as optimizing algorithm. For loss function, we adopted cross-entropy loss since this works best on classification problem. A batch size of 64 was used and the training was performed for 100 epochs. Since we are only fine tuning, we did not add any extra monitoring on training such as- early stopping or reduce learning rate.

### D. Hardware Platform

All our experiments were performed on *University of Florida HiPerGator* cloud account. We used an Nvidia A100 GPU with 16GB of RAM. Although we utilized gpu resources for training and testing, our code will still work without gpu support.

### E. Software Packages

For programming language, we used Python (version 3.7) since it provides a wide range of machine learning and deep learning libraries. For deep learning libraries, we used PyTorch (1.12, gpu version) and Torchvision (0.2.1). For gpu support, we used Nvidia cuda, cudatoolkit (version 11.3.1), and cudnn. Other necessary machine learning packages are scipy and scikit-learn. For image processing, we used pillow library.

## IV. Experiments

### A. Classical Machine Learning

We started with some basic machine learning approach to understand how they perform in this specific problem. First, we used a simple Logistic Regression (LR) pipeline. Next, for reducing the feature space, we used Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) with other classification algorithms. We observed that first 1000 Principal Components (PCs) can explain more than 90% variance of this dataset. so we experimented with 1000 PCs and a Logistic Regression classifier. For LDA, it can only keep a maximum of 9 dimensions since the dataset has 10 classes. We used 9 dimensional LDA with LR classifier. Since the compression from 270000 to 9 dimensions is a huge jump, many valuable features are lost. Hence, LDA performs poor in this task. Next, we applied Support Vector Machine (SVM) which is more capable than LR for classification task. We used two different kernels- RBF and Polynomial. To speed up, we again integrated dimension reduction via PCA. The highest accuracy was found when we reduced the dimension to 1000 principal components and used these components with SVM (RBF kernel) as classifier (52%). All these pipelines used *MinMaxScaling* as preprocessor. The results are presented in Table I.

### B. Transfer Learning with CNN

As discussed in Section III, instead of training from scratch, we used a pretrained *ResNet-50* architecture as backbone. We added some fully connected layers on top of it and experimented with different parameter settings. Specifically, we added three fully connected layers. The first two hidden layers had a commonly used ReLU activation and dropouts in between these layers. The final layer had 10 units with linear activation, as the model was being trained to classify 10 classes. We experimented with Softmax activation on the output layer, however it did not improve result. For learning rate, we tested with 0.01, 0.005, 0.001, and 0.0005. Higher learning rate makes the training unstable since it is a fine-tuning task. We used 0.001 learning rate for final result. Different batch size- 32, 64, 128 were used for training. Although we did not observe a significant impact of batch size on training performance, keeping the resource constraint in mind, we set the batch size to 64. We ran the training for

TABLE I: Performance of classical ML techniques on validation dataset.

| Method | LR | LR with LDA | LR with PCA | SVM with PCA |
|---|---|---|---|---|
| Accuracy | 36.56% | 18.82% | 51.22% | 53.09% |

100 epochs and as seen from Figure **??**, highest accuracy is reached approximately after 60 epochs.

Training validation split is an important parameter for learning. We started with 90/10 train/val split and observed that validation accuracy falls which indicates the model is overfitting on train data. With multiple experiments, we set the ratio to 70/30 which ensures the model learns well but is not overfitted.

For *ResNet* model, the input images must have a height and width which is multiple of 32. Since our raw images are 300x300x3, we resized these to 224x224x3 so that we do not lose much information and the size is in accordance with model requirement.

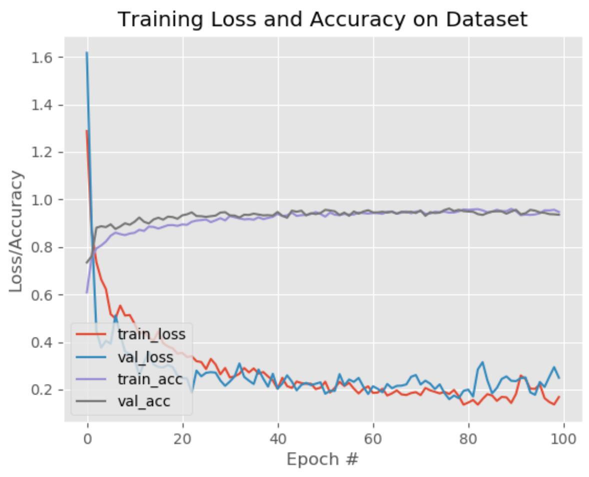

Fig. 2: Training performance. The model reaches close to optimal solution at around 60 epoch.

## C. Results

First, we demonstrate the results of classical machine learning models in Table I. As we see from the table, these methods can not accurately classify the logos. It is obvious since our dataset contains large number of samples and the images vary widely in terms of visual perspective. Hence a deep learning model is necessary correctly classify this dataset. With the deep CNN model, we achieved 94% accuracy on the validation dataset. The class-wise performance metric on validation dataset is presented in Table II.

We investigated the misclassified examples. Some examples are shown in Fig 3. It can be claimed that the model fails on those images where the logo is too small with respect to background or when there are many other irrelevant information other than the desired logo.

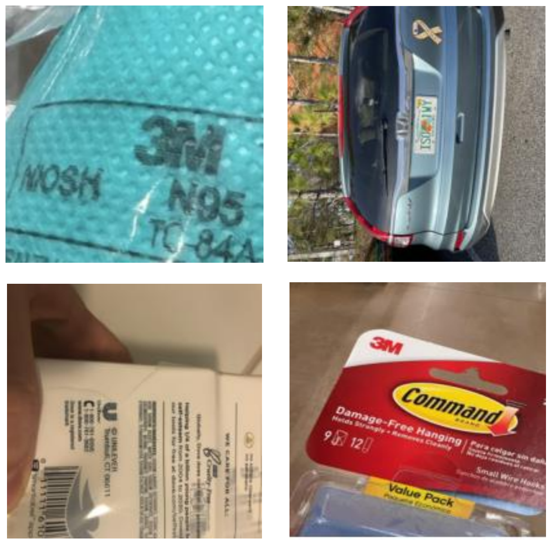

Fig. 3: Some misclassified examples.

TABLE II: Class-wise performance of final model on validation dataset. The mean accuracy is 94%.

| Class | Precision | Recall | F1-score | No. of samples |
|---|---|---|---|---|
| 0 | 0.96 | 0.94 | 0.95 | 112 |
| 1 | 0.95 | 0.84 | 0.89 | 111 |
| 2 | 0.88 | 0.96 | 0.92 | 102 |
| 3 | 0.95 | 0.97 | 0.96 | 129 |
| 4 | 0.96 | 0.96 | 0.96 | 116 |
| 5 | 0.95 | 0.89 | 0.92 | 111 |
| 6 | 0.89 | 0.99 | 0.94 | 102 |
| 7 | 1.00 | 0.90 | 0.95 | 117 |
| 8 | 0.86 | 0.94 | 0.90 | 108 |
| 9 | 0.95 | 0.98 | 0.96 | 107 |

## V. Conclusion

In this project, we developed a deep machine learning pipeline that can successfully recognize 10 brand logos from different challenging images. Our customized ResNet-50 model shows satisfactory performance on validation dataset. With more preprocessing on training data, specially cropping out the irrelevant parts, it is possible to further improve the accuracy. This project can be deployed into various industry level applications, such as brand monitoring and trademark protection.

## References

[1] N. Sharma, V. Jain, and A. Mishra, "An analysis of convolutional neural networks for image classification," *Procedia computer science*, vol. 132, pp. 377–384, 2018.

[2] J.-H. Chiam, "Brand logo classification," *Stanford University*, 2015.

[3] S. Bianco, M. Buzzelli, D. Mazzini, and R. Schettini, "Deep learning for logo recognition," *Neurocomputing*, vol. 245, pp. 23–30, 2017.

[4] P. Pimkote and T. Kangkachit, "Classification of alcohol brand logos using convolutional neural networks," in *2018 International Conference on Digital Arts, Media and Technology (ICDAMT)*. IEEE, 2018, pp. 135–138.

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] A. M. Neha Sharma, Vibhor Jain, “Imagenet: A large-scale hierarchical image database,” in *Procedia Computer Science*. Elsevier, 2018, pp. 377–384.