EEL5840: Fundamentals of Machine Learning

Final Project - Technical Report
Project Title: Brand Logos Image Classification

Regressive Reasoning

Matheus Kunzler Maldaner Pedro Moss Ruo Chen

Abstract—**Image classification is an important part of machine learning, but also can serve as an introduction to a student learning the fundamentals. This provides a summary of a project that is meant to classify ten different brand logos, and document the attempt to build a custom convolutional neural network architecture before using transfer learning to leverage the improved performance of established models. By the conclusion of the project, a 95% validation accuracy was attained using transfer learning.**

I. Introduction

Machine learning has become increasingly more feasible as sources of digital data increase. This has allowed the creation of sophisticated models that allow for the classification of images by leveraging the amount of information available. With this surge in digitization, accurate classification of logo brands is becoming increasingly important.

The task documented in this report is to train a model to detect and identify brand logos in images, using a dataset collected by multiple groups. This report will include initial attempts at designing a Convolutional Neural Network (CNN) to solve the problem, experimentation with different models and the application of transfer learning to eventually arrive at a final solution. The experimentation section will also detail challenges encountered when approaching this problem, and how these difficulties were mitigated in order to arrive at an optimal solution.

Being able to have a machine learning model automatically detect and classify images can allow for autonomous collection of data without the need for human observation, vastly saving resources and time that could be spent on other tasks.

Various attempts to simplify this problem and create more efficient models have been attempted and have proven successful [1], [2]. This project and its experiments mostly compare the different architectures and determine how well they perform on the brand-logo dataset and compare it to how a custom CNN performs.

II. Implementation

There are ten brands that the model is meant to be able to classify, listed in Table 1. The dataset consists of 5933 images, with the number of each brand in the image set varying slightly but roughly equal. The project was mainly done by utilizing shared computing resources on the University of Florida HiPerGator supercomputer network.

The models that are used in this report consist of Convolutional Neural Networks (CNNs). It was attempted to design a custom architecture, and once the validation accuracy plateaued, transfer learning using pre-trained models was used as an alternative to improve accuracy.

Brand Logo	Integer Encoding
Nike	0
Adidas	1
Ford	2
Honda	3
General Mills	4
Unilever	5
McDonald's	6
KFC	7
Gator	8
3M	9

Table 1 - Brand logos and their respective integer encodings

There were multiple avenues of approach to preprocessing the data before utilizing it to train the models. Preprocessing was conducted to eliminate less important information to mitigate memory usage issues, improve model training speeds and to reshape it in the required format needed by the transferred models.

First, each image in the dataset was manually evaluated. Any mislabeled or outlier images within the dataset which could impact model behavior was noted. During training, performance was evaluated on two variations of the dataset, one with all of the images and the other which dropped the noted images.

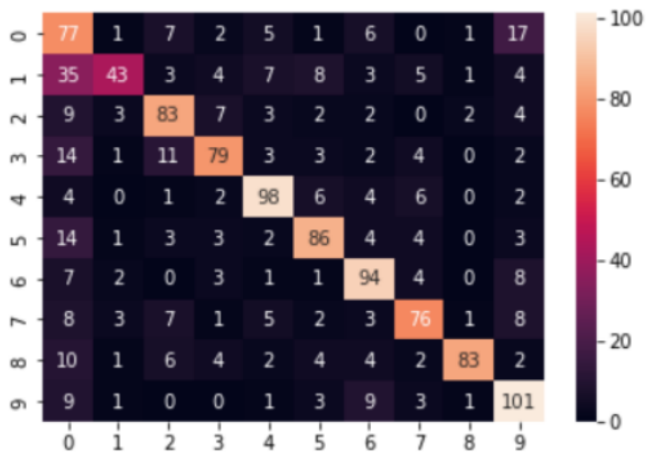
Grayscaling the images was performed with the idea that the most important information about a logo was its shape, a principle that's been explored in various image processing studies [1]. Another benefit of grayscaling is that it substantially lowered the number of parameters the model needed to learn, since the color channels tripled the number of features. Additionally, some images came in different colors which would negatively impact the model predictions, as more data would be needed for the model to learn color associations. Ultimately, the custom architecture that was self-designed for this project was trained only on grayscale images.

Python libraries used in the implementation of both the custom CNN and the models used for transfer learning were:

- Numpy 1.22.3
- Matplotlib (to display results)
- Tensorflow w/ Keras libraries 2.7.0
- Scikit-Learn 0.24.2

To separate the data into a training, validation, and test set the scikit python library was utilized [9] while Numpy was employed for numerical computations and data handling [10]. From the full image set, 64% of the images were used for training, 16% for validation, and 20% for testing.

The custom CNN was first implemented by creating a shallow neural network. Changes were made to the neural network and its performance was evaluated to determine how such changes affected model performance, which will be detailed in the next section. Upon each iteration of the model, confusion matrices were generated as a way of identifying which classes the model had trouble identifying.



	0	1	2	3	4	5	6	7	8	9
0	77	1	7	2	5	1	6	0	1	17
1	35	43	3	4	7	8	3	5	1	4
2	9	3	83	7	3	2	2	0	2	4
3	14	1	11	79	3	3	2	4	0	2
4	4	0	1	2	98	6	4	6	0	2
5	14	1	3	3	2	86	4	4	0	3
6	7	2	0	3	1	1	94	4	0	8
7	8	3	7	1	5	2	3	76	1	8
8	10	1	6	4	2	4	4	2	83	2
9	9	1	0	0	1	3	9	3	1	101

Figure 1 - Confusion Matrix depicting heavy misclassification associated with Class 0.

Once it became noticeable that the custom model was not increasing in performance and it became extremely difficult to improve the model despite adjusting parameters and layers, the focus was changed to exploring the implementation of transfer learning architectures such as ResNet [3], VGG19 [2], and EfficientNet [4]. Transfer learning involved additional challenges. The models explored were slow and prone to crashing the kernel due to memory allocation issues. These issues were eventually handled, which will be detailed in the experiments section.

III. Experiments

The first approach to building a model to classify the brand logos was to design a custom CNN. The CNN was gradually built upon and layer parameters adjusted while its performance was evaluated by observing its validation accuracy and plotted confusion matrices. Each variation of the CNN involved flattening the convolutional output as inputs into fully-connected hidden layers before outputting through a 10-node fully-connected output layer using the softmax activation function. Each iteration of the model was trained with a batch size of 64 for 80 epochs, an early stopping function with a patience of 10 epochs that used the validation loss as a metric, and an Adam optimizer with a learning rate of 0.0001. All convolutional layers used a ReLU activation function.

For the sake of expediency, only models that improved significantly over previous iterations will be discussed. The shallow CNN performed poorly with a validation and training accuracy of about 30%. The cause of this was determined to be that the model was not complex enough to capture features of the images to make accurate predictions, on top of not performing regularization or batch normalization [5].

Reviewing the model summary revealed that the total number of parameters was over 19 million, the majority of which was in the fully-connected layer. To reduce the number of parameters that needed to be learned, the number of filters in the first convolutional hidden layer was reduced and a max pooling layer was introduced. This reduced the number of parameters to 18,000, which significantly sped up training and increased validation accuracy to 40%.

Adding a batch normalization layer after the first convolutional neural network and two convolutional layers with fewer filters and smaller kernel sizes further increased validation accuracy, up to 50%.

Adding batch normalization and max pooling layers after each convolutional layer, as well as increasing the number of filters in each convolutional layer improved validation accuracy by 20%, up to 70%. This however, dramatically increased the model complexity from 18,000 parameters to 1,000,000 parameters. This added complexity also caused the model to overfit, as training accuracy became 100% while validation accuracy stagnated at 70%.

After a certain point it was observed that the CNN had a possible issue with overfitting [5], highlighted by the validation accuracy being lower than the training accuracy. To attempt to mitigate this, an attempt was made to add regularizers and just layers to reduce the complexity of the model. Dropout layers were added after every convolutional layer which mitigated overfitting and led to an accuracy increase in the model to 75% [6]. This also halved the number of parameters to 560,000. At this point, changes made to the model made no noticeable difference and even hampered its ability to classify images. It was decided to use transfer learning.

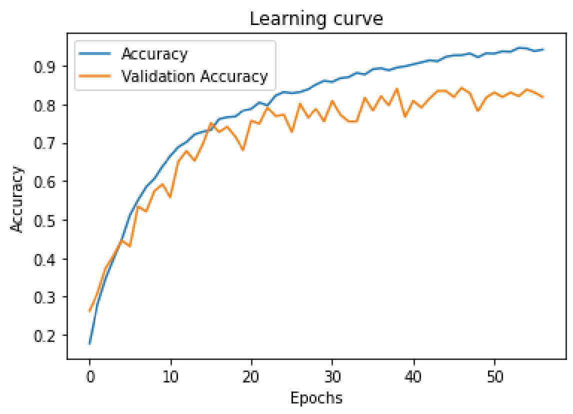
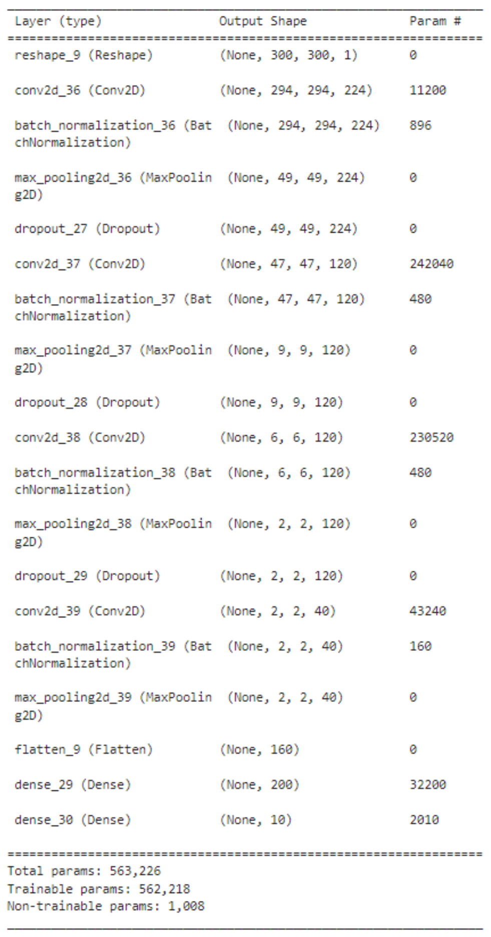


Figure 2 - Graphs showing the training and validation accuracy per epoch.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 reshape_9 (Reshape)         (None, 300, 300, 1)       0

 conv2d_36 (Conv2D)          (None, 294, 294, 224)     11200

 batch_normalization_36 (Bat  (None, 294, 294, 224)    896
 chNormalization)

 max_pooling2d_36 (MaxPoolin  (None, 49, 49, 224)      0
 g2D)

 dropout_27 (Dropout)        (None, 49, 49, 224)       0

 conv2d_37 (Conv2D)          (None, 47, 47, 120)       242040

 batch_normalization_37 (Bat  (None, 47, 47, 120)      480
 chNormalization)

 max_pooling2d_37 (MaxPoolin  (None, 9, 9, 120)        0
 g2D)

 dropout_28 (Dropout)        (None, 9, 9, 120)         0

 conv2d_38 (Conv2D)          (None, 6, 6, 120)         230520

 batch_normalization_38 (Bat  (None, 6, 6, 120)        480
 chNormalization)

 max_pooling2d_38 (MaxPoolin  (None, 2, 2, 120)        0
 g2D)

 dropout_29 (Dropout)        (None, 2, 2, 120)         0

 conv2d_39 (Conv2D)          (None, 2, 2, 40)          43240

 batch_normalization_39 (Bat  (None, 2, 2, 40)         160
 chNormalization)

 max_pooling2d_39 (MaxPoolin  (None, 2, 2, 40)         0
 g2D)

 flatten_9 (Flatten)         (None, 160)               0

 dense_29 (Dense)            (None, 200)               32200

 dense_30 (Dense)            (None, 10)                2010

=================================================================
Total params: 563,226
Trainable params: 562,218
Non-trainable params: 1,008
_________________________________________________________________
```

Figure 3 - Custom neural network model final iteration summary

For the shift to transfer learning, experimentation was performed with several different pre-trained models in order to find the one that had the best performance. An experiment was set up where various models were trained sequentially using the same train, validation, and test sets, with their accuracy being recorded along the way.

Typically the approach to transfer learning is to freeze the layers of the pre-trained model, and then train layers on top of this to use the features obtained from the underlying model to help solve the problem. However, in these experiments, a different approach was taken. The underlying model was used as a starting point for weights and these weights were then updated during training. This was found to result in higher accuracy on the test set.

The models used were those readily available on Tensorflow through Keras Hub, which allowed for easy cycling through different models to assess performance. For all models, a single output layer was applied on top of the pre-trained layers that had 10 nodes to match the desired output, with a softmax activation. Sparse categorical cross entropy was used as the loss function with the ADAM algorithm used as the optimizer. The models were trained for up to 200 epochs, with an early stopping callback that monitored the validation loss with a patience of 3 epochs. The table below provides all of the models that were trained along with their final accuracy on the test set.

Model	**Accuracy**
Resnet_v2_50	86%
Resnet_v2_152	68%
Resnet_v2_101	83%
Inception _v3	89%
Inception_resnet_v2	93%
BiT-S R50x1	85%
Efficientnetv2_s	96%
Efficientnetv2_m	95%
Efficientnetv2_b3	95%
Efficientnetv2_b2	93%
Efficientnetv2_b1	95%
Efficientnetv2_b0	94%

Table 2 - Model names and their test set accuracies

It is observed here that the EfficientNet models performed the best on the test set after training. It should be noted that it appeared that the larger models actually did not perform as well on the test set. This could be due to the way the transfer learning model was set up, and the larger models may have required more layers on top of the model in order to interpret obtained features. This, however, would have resulted in increased computation time. The results above led to the choice of “EfficientNet_V2_s” as the final model. Below is the confusion matrix for this model on the test set.

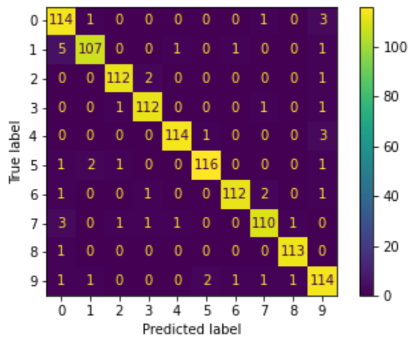


Figure 4 - EfficientNet_V2_s confusion matrix.

IV. Conclusions

Throughout the machine learning project, various challenges were faced and several key discoveries were made. The initial goal was straightforward: classify brand logos from images. While this task is easy for humans, it becomes complex when approached with machine learning. The first efforts using custom Convolutional Neural Networks (CNNs) highlighted the importance of model design. It was discovered that even small adjustments can greatly affect the overall model performance.

An important lesson from the experiments is the value of transfer learning. By using pre-trained models like ResNet, VGG19, and EfficientNet, benefits were gained utilizing the features these networks learned from other large datasets. [2], [3], [4]. This approach showed how flexible and powerful CNNs can be. Through the usage of the Keras framework with the TensorFlow backend, models trained for one task could be adapted and excel in another task. [7],[8].

To extend this project, there is the possibility of using multi-classification models, such as the YOLO algorithm, to allow for the identification of multiple objects, or brand logos, in the same image. There is also the possibility to have the model classify images it has never seen before as ‘unknown’.

In conclusion, the project demonstrates the typical process of machine learning: start with a basic model, address challenges as they come, and apply advanced techniques like transfer learning when needed. This approach led us to achieve a high validation accuracy of 95%.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, vol. 25, 2012.
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.
[4] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," arXiv preprint arXiv:1905.11946, 2019.
[5] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.
[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," The Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929-1958, 2014.
[7] F. Chollet, "Keras," 2015. [Online]. Available: https://keras.io.
[8] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," arXiv preprint arXiv:1603.04467, 2016.
[9] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of machine learning research, vol. 12, no. Oct, pp. 2825-2830, 2011.
[10] C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, no. 7825, pp. 357-362, 2020