EEL5840 Final Project - Supergator

Jiamin Chen, Kai Wei, Nikodem Gazda
The Department of Electrical & Computer Engineering
The University of Florida
Gainesville, Florida

***Abstract*—This paper describes the implementation, experimentation, and results for a convolutional neural network (CNN) based on the GoogLeNet architecture trained to classify natural images of logos for 10 different companies. The GoogLeNet model surpassed other supervised models like the SVM and pre-trained models like the DenseNet and ResNet models for its higher accuracy performance, computational efficiency, and inception modules. The training data is augmented and split into an 80/20 pair of train and validation sets before the model is built, compiled, and trained with the Adam optimizer and StepLR scheduler. The results show the GoogLeNet model exhibits the lowest and most stable validation losses, however, the DenseNet and Resnet models also have similar losses with duration times 2-3x less than that of the GoogLeNet model.**

I. Introduction

This paper presents the implementation, experiments, and results of a convolutional neural network (CNN) based on the GoogLeNet architecture that was trained to classify natural images of the logos for Nike, Adidas, Ford, Honda, General Mills, Unilever, McDonald's, KFC, Gator, and 3M.

The GoogLeNet model was chosen for several reasons including its computational efficiency [4], which was essential given the limitations of Hypergator. The reduced training time that results from this computational efficiency also allowed us more freedom to experiment with the model. The model's inception modules were a valuable feature which allowed for the recognition of features with different scaling, which is common in the dataset provided for the project. Additionally, the GoogLeNet model fit our implementation because of its high accuracy performance, demonstrated by its competitive standing in the ILSVRC 2014 competition [5].

In terms of the model's architecture, GoogLeNet uses 22 layers excluding pooling layers. The model features several convolution layers in the earlier layers of the architecture and is followed by stacks of the aforementioned inception layers, which include max pooling layers [4] and an average pooling layer before classification [5]. The model also uses dropout for regularization, RELU activation functions for the hidden layers, and a softmax activation function for the fully connected output layer. One unique aspect of this architecture is that during training, the model uses classifiers in the intermediate layers and sums their losses into the total loss at a reduced weight. These reduced classifiers are then removed for model prediction to reduce prediction times [4].

In previous studies, GoogLeNet has been tested and compared against other models for diabetic retinopathy classification [1], where researchers found the ADAM optimizer outperformed other optimization techniques including RMSPROP and SGD; for tumor classification [2], where despite significantly outperforming the ResNet model, researchers concluded GoogLeNet takes a long time to train on large datasets; and for extreme weather classification [3], where researchers were able to improve accuracy and training speed by truncating the model.

The GoogLeNet model used in the final implementation is pre-trained on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2012) dataset [4]. This dataset contains a multitude of images based on WordNet's catalog with around 500–1000 clean and full resolution images for each category [6], making the GoogLeNet model ideal for our purposes.

Furthermore, the dataset used to train the model for classification of the 10 logos was obtained by each student, garnering a total of 3717 images. The training data was augmented to increase the performance and generalization of the model. Despite its significant impact on the accuracy of the CNN implemented , data augmentation has the potential to reduce training speeds by a factor of 2x and impacted the implemented model accordingly. Nevertheless, data augmentation still proved a useful tool to improve the difficulty of the training data for this classification task.

II. Implementation

A. Data preprocessing

The provided training data consists of a 27000×3717 matrix, containing 3717 samples of $300 \times 300 \times 3$ images. By employing image preprocessing techniques, we enhanced the quality of the training data, which ultimately contributed to obtaining superior results.

a) Data cleaning: Upon examining the initial dataset, it was apparent that several images were mislabeled, necessitating manual intervention to rectify these inaccuracies and ensure the integrity of the data before further analysis could be conducted.

b) Unknown class: To acquire extra images for an unknown class in a brand logo classification project, an approach was used that involves querying the Pixabay API to download diverse images that are not related to the brand logos in the original dataset. By downloading images from different categories, including abstract, nature, animals, cityscapes, patterns, architectures, food, activities, people, and objects, a diverse set of images was obtained for the unknown class (label -1). There are 200 images in each category and 2000 in total. These images were converted to the same size as the given

data and added to the training data to help the model learn to recognize when a test point class is "unknown" and was not in the original training data, allowing it to return the correct label -1 for such cases. This approach aims to improve classification accuracy on the "hard" dataset, which includes all 10 original classes as well as the unknown class.

c) Setting hyperparameters: The "num_classes" hyperparameter specifies the number of classes in the dataset, "in_channel" specifies the number of channels in the input images, and "learning_rates" is the learning rate of the optimizer used during training. The "batch_size" hyperparameter is the number of samples in each batch during training and "num_epochs" specifies the number of epochs (full passes through the training dataset).

d) Data Augmentation: To convert the data into a PyTorch-ready data type, we use the function "transform.ToTensor" to convert the image to a PyTorch tensor, and normalize the tensor by subtracting the mean and dividing by the standard deviation of the image. To improve the difficulty of the data, we use the function "transform.RandomHorizontalFlip" to randomly flip the image horizontally, randomly rotate the image by 15 degrees, resize the image to 224×224, and crop it randomly from the original image.

B. Building the training model

In this project, we construct a CNN model utilizing the GoogLeNet architecture to train the data. Aiming to enhance the training outcomes, we establish the loss function, optimizer, and scheduler for the model's training process.

a) Data splitting: We use PyTorch's subset and DataLoader classes to split a given dataset into training and validation sets and create PyTorch dataloaders for each set.

b) GoogLeNet: GoogLeNet is a deep neural network model, founded on the Inception module, introduced by Google. Comprising 22 layers [4], the GoogLeNet network is distinct from models like VGG, LeNet, and AlexNet due to its avoidance of the three-layer fully connected network, which is known for its large number of parameters, high computational complexity, and propensity for overfitting. Instead, GoogLeNet employs Average Pool and Dropout methods immediately following the Inception modules, contributing to dimensionality reduction and mitigating overfitting to some degree.

C. Compile the training model

After constructing the model, the subsequent step involves compiling it, establishing the optimizer and loss function, incorporating metrics for evaluating the training process, and then training the model.

a) Build a training framework: We set up a PyTorch model for training using cross-entropy loss as the loss function, Adam optimizer as the optimization algorithm, and a learning rate specified by a variable. Additionally, a StepLR scheduler is set up to adjust the learning rate during training. Our code also implements early stopping to stop training if the validation loss does not improve for a specified number of epochs.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Fig. 1: The architecture of GoogleLeNet

b) Training loop: For each epoch iteration during training, the data and labels are transferred to the GPU, a forward pass is performed, the loss is computed, the backward pass is performed, and the optimizer updates the learning rate.

III. Experimentation

A. Data Augmentation

Data augmentation improves model generalization and robustness by increasing dataset diversity and size. It helps avoid overfitting, allows models to learn important features, and performs better on unbalanced datasets by creating new variations of the original images, such as cropping, flipping, and rotating. This makes models more resistant to changes in orientation, scale, and perspective, resulting in improved overall performance.

We used different transformations for the train set and validation set. The train set was randomly flipped, rotated, cropped, and scaled, while the test set had the same cropping and scaling applied to each sample.

In the experiment for data augmentation, we test the performance with data augmentation and without it on GoogLeNet model. The comparison is shown in Table I.

Model	Train Loss	Train Accuracy	Val. Loss	Val. Accuracy	Elapsed Time (s)
Data Aug.	0.0055	99.96%	0.0768	97.55%	141.49
No Data Aug.	0.0041	99.98%	0.0966	97.20%	106.18

TABLE I: Comparison of Effect of Data Augmentation

We can see that the case without data augmentation takes shorter time, but with data augmentation, we can get a better accuracy in validation set.

B. Comparison Between Models

In this project, we experimented with different models including SVM, GoogLeNet, Resnet51, and DenseNet121. For training and validation, we split the dataset into a train and validation set with an 80/20 separation.

1) Support Vector Machine (SVM): The first classification method we chose was the Support Vector Machine (SVM) because it is simple and easy to implement. We used a soft-margin SVM with the RBF kernel and utilized grid search cross-validation to determine the best parameters. In order to avoid overfitting, we resized the images to 50×50, and converted them to grayscale. Though this model performs well on the train set with an accuracy of 96.06%, the results show it only achieved an accuracy of 49.83% on the validation set with the best parameters `gamma = 'scale'`, and `C = 5`. The model also took over 900 seconds to run, which is too time-consuming.

2) Pre-trained Models: Using pre-trained models for image classification offers advantages such as faster convergence, improved performance, and resource efficiency, primarily due to the transfer of learned features from large-scale datasets. There are several popular models for image classification tasks. Some of them are:

1) GoogLeNet: A model which features the Inception module, a novel building block that allows the network to be deeper and more efficient while reducing computational complexity [4].
2) ResNet: A popular model with multiple variants (ResNet18, ResNet34, ResNet50, ResNet101, ResNet152). Also trained on the ImageNet dataset, ResNet's aim is to mitigate the vanishing gradient problem in deep networks with shortcut connections. These shortcut connections add the input of a node to its output, creating residual functions that can approximate the outputs and allow for identity mapping, which let the network learn updates between the layers instead of learning the entire function. This architecture is implemented without introducing extra parameters or computational complexity and is applicable both to fully-connected layers as well as convolutional layers [7].
3) DenseNet: This model uses densely connected layers, where each layer receives feature maps from all preceding layers and outputs to all subsequent layers. Like ResNet, this model improves the flow of information through the network, diminishes the vanishing gradient problem, and cuts down on the amount of parameters. Since DenseNet connects previous layers to subsequent layers, it follows that this model helps with feature propagation and amplifies feature reuse as well [8].

For logo classification, it's crucial to choose a model that best fits the computational resources, dataset size, and desired accuracy. Since logos can have complex patterns, deeper models like GoogLeNet, ResNet50, and DenseNet121 might be suitable for better performance.

Model	**Train Loss**	**Train Accuracy**	**Val. Loss**	**Val. Accuracy**	**Elapsed Time (s)**
SVM	/	96.06%	/	49.83%	902.47
ResNet51	0.0388	98.78%	0.1243	96.85%	134.52
DenseNet121	0.0145	99.78%	0.1033	97.38%	155.44
GoogLeNet	0.0055	99.96%	0.0768	97.55%	141.49

TABLE II: Comparison of Performance of Different Models

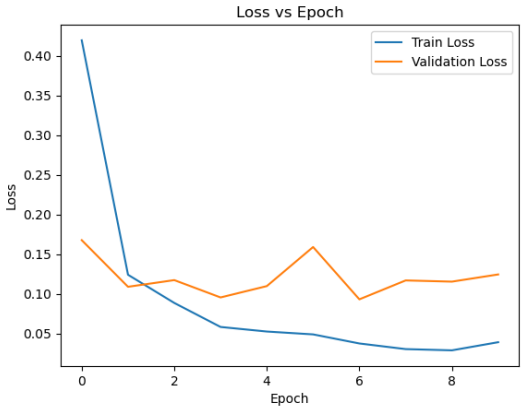

Fig. 2: Loss vs epoch of ResNet51.

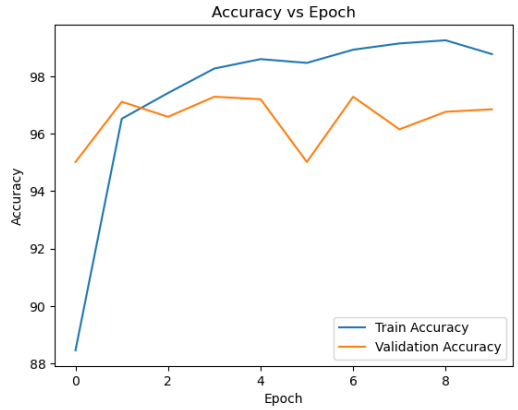

Fig. 3: Accuracy vs epoch of ResNet51.

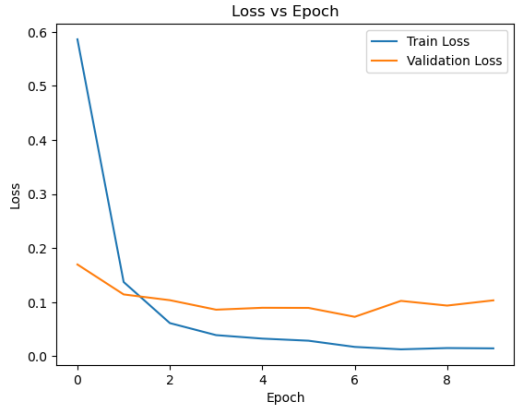

Fig. 4: Loss vs epoch of DenseNet121.

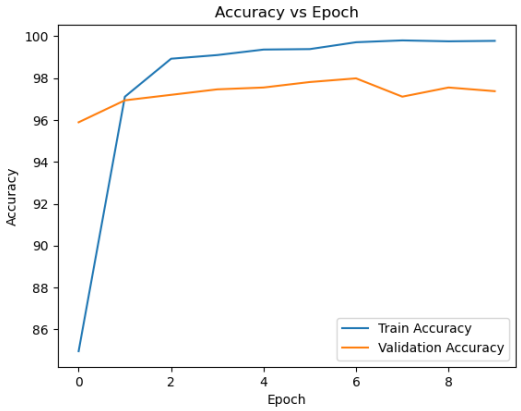

Fig. 5: Accuracy vs epoch of DenseNet121.

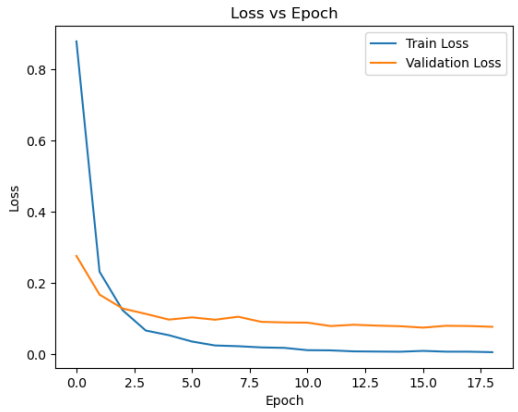

Fig. 6: Loss vs epoch of GoogLeNet.

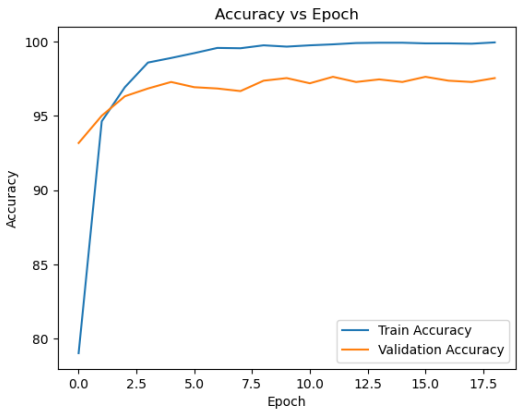

Fig. 7: Accuracy vs epoch of GoogLeNet.

IV. Conclusions

From Table II, we can see that all models performed similarly for training accuracy, but the GoogLeNet model performed best for validation accuracy, followed by the DenseNet121, ResNet51, and SVM models. In terms of duration, ResNet51 outperformed the next quickest model (GoogLeNet) and the DenseNet121 model. The SVM model, while simple to implement, performed the worst in both duration and accuracy.

The accuracy and loss plots also add insight into each model's performance. Figs. 2 and 3 show that as the train set's loss decreases and accuracy increases, the validation set's loss initially decreases before exhibiting an overall increase, indicating the possible presence of overfitting. The validation accuracy and loss in the DenseNet121 model also initially improve in Figs. 4 and 5 before oscillating at a steady value. Though the trend is less noticeable than with the ResNet51 model, these plots also exhibit a slight incline in loss and decline in accuracy in the validation set as the train set improves. The GoogLeNet model we chose for the classification problem displays the highest stability and accuracy for the validation set (Figs. 6, 7).

References

[1] A. H. M. Mohammed and M. Çevik, "GoogleNet CNN Classifications for Diabetics Retinopathy," 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1-4, doi: 10.1109/HORA55278.2022.9799971.

[2] G. V. Sai Charan and R. Yuvaraj, "Efficient Tumor Classification using GoogleNet Approach to Increase Accuracy in Comparison with ResNet," 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2022, pp. 490-494, doi: 10.1109/ICIEM54221.2022.9853203.

[3] Z. Zhu, J. Li, L. Zhuo and J. Zhang, "Extreme Weather Recognition Using a Novel Fine-Tuning Strategy and Optimized GoogLeNet," 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA), Sydney, NSW, Australia, 2017, pp. 1-7, doi: 10.1109/DICTA.2017.8227431.

[4] C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.

[5] M. A. E. Muhammed, A. A. Ahmed and T. A. Khalid, "Benchmark analysis of popular ImageNet classification deep CNN architectures," 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bengaluru, India, 2017, pp. 902-907, doi: 10.1109/SmartTechCon.2017.8358502.

[6] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[7] F. Albardi, H. M. D.Kabir, M. M. I. Bhuiyan, P. M.Kebria, A.Khosravi and S. Nahavandi, "A Comprehensive Study on Torchvision Pre-trained Models for Fine-grained Inter-species Classification,2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia,2021pp.2767-2774, doi: 10.1109/SMC52423.2021.9659161.

[8] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for ImageRecognition,2016 IEEE Conference on Computer Vision and PatternRecognition (CVPR), Las Vegas, NV, USA, 2016, pp.770-778,doi:10.1109/CVPR.2016.90.