Building the Strongest CNN Model based on AlexNet and GoogleNet Architectures

Khushi Raghuvanshi Elizaveta Beltyukova

Abstract

This paper compares AlexNet and GoogleNet architectures for image classification on the CIFAR-10 dataset, evaluating the impact of hyperparameters and architectural choices on performance. By tuning parameters such as learning rate, activation functions, weight decay, number of filters, dropout rate, and pooling methods, the study identifies optimal configurations for each architecture. The results indicate that GoogleNet consistently outperforms AlexNet even on CIFAR-10 with an optimal accuracy of 74.25% with ReLU activation. These findings emphasize the benefits of deeper architectures with inception modules for multi-scale feature extraction.

1. Introduction

Deep learning has transformed computer vision, enabling unprecedented performance in tasks such as image classification, object detection, and segmentation. Convolutional Neural Networks (CNNs) have become the cornerstone of modern image-processing techniques. While numerous CNN architectures have emerged, this study focuses on two models: AlexNet and GoogleNet. These architectures represent distinct approaches to image classification and provide a valuable basis for comparative analysis. The primary aim of this work is to evaluate and compare the performance of models inspired by these architectures on the CIFAR-10 dataset, a widely adopted benchmark for image recognition.

AlexNet, characterized by its eight-layer structure, demonstrated the potential of deep learning for visual tasks, achieving groundbreaking results in the ImageNet image classification challenge (Klingler, N.). GoogleNet, conversely, introduced the innovative Inception module, designed to enhance computational efficiency and feature extraction through parallel convolutional pathways. This module reduces computational costs in networks (Ballester, P., & Araujo, R.). This paper delves into a systematic evaluation of the relative strengths and weaknesses of architectures derived from AlexNet and GoogleNet when applied to the CIFAR-10 dataset.

Motivated by the ongoing need to optimize CNN architectures for specific applications, this study seeks to determine how architectural choices impact performance on a constrained dataset. By developing and training models inspired by both GoogleNet and AlexNet on CIFAR-10, we aim to identify optimal hyperparameter configurations that deliver enhanced performance. Key performance indicators, including classification accuracy, loss, and model complexity, will be analyzed to provide a thorough comparative assessment. Ultimately, this work seeks to illuminate the architectural trade-offs inherent in CNN design and their practical implications for image classification tasks.

2. Method & Architecture Description

AlexNet is a deep learning model that was created to outperform older models in image classification. AlexNet has 8 layers: 5 convolutional layers and 3 fully connected layers. Some key properties include the use of the RELU activation function, the local response normalization, and max pooling. The first convolutional layer has 96 kernels (11x11, stride 4), uses RELU as the activation function, and then performs max pooling. The second layer has 256 kernels (5x5x48). The next three layers don't involve any pooling or normalization. The third convolutional layer has 384 kernels (3x3x256), the fourth has 384 (3x3x192), and, lastly, the fifth has 256 (3x3x192). Next, the first two fully connected layers have 4096 neurons each, using a RELU activation at the end. The last fully connected layer (also known as the

output layer) has 1000 neurons and uses the softmax activation function.

GoogleNet, introduced by Szegedy et al. in 2014, represents a significant advancement in deep convolutional neural networks (CNNs), prioritizing both high accuracy and computational efficiency. Diverging from traditional CNN architectures that sequentially stack layers, GoogleNet incorporates the innovative Inception module. This module is the cornerstone of the architecture, enabling the extraction of multi-scale features within a single layer by utilizing parallel convolutional streams with varying kernel sizes (1x1, 3x3, and 5x5) to capture spatial information. Furthermore, GoogleNet employs 1x1 convolutions as bottleneck layers for dimensionality reduction with no substantial computational cost at the expense of model capacity. Key components of GoogleNet include the Inception modules, which facilitate parallel multi-scale feature extraction; 1x1 convolutions for parameter reduction; auxiliary classifiers to mitigate the vanishing gradient problem in deep networks; and global average pooling (GAP) to replace fully connected layers, reducing overfitting and computational cost. These design choices enable GoogleNet to achieve a good trade-off between efficiency and accuracy, making it a baseline model for image classification tasks using deep learning.

3. Experiments

3.1 Dataset

This project uses the CIFAR-10 dataset, a standard image classification problem benchmark. CIFAR-10 has 60,000 color images with 64×64 pixels and ten classes: airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The dataset splits into 50,000 training images and 10,000 test images.

3.2 Problem Description

The task was to train a convolutional neural network on the Cifar-10 dataset using a deep learning platform of choice. We chose PyTorch for our platform and decided to try to find the highest-performing network using AlexNet and GoogleNet as starting points. We changed multiple aspects of each model, including pooling methods, activation functions, normalization, learning rate, weight decay, and batch sizing.

3.3 Hyperparameter Training

Hyperparameters tested for AlexNet framework:

1. Different activation functions: RELU, Leaky RELU, and ELU
2. Max Pooling vs. Average Pooling
3. L2 normalization
4. Learning rate: 0.0001, 0.1
5. Weight decay: 0.00001, 0.01
6. Batch size: 32, 64, 128
7. Number of filters
8. Fully connected layer size (128, 256, 512)
9. Dropout rate (0.2, 0.5)

Hyperparameters tested for GoogleNet framework:

1. Different activation functions: RELU, ELU, and Leaky RELU.
2. Max Pooling vs. Average Pooling
3. Number of filters
4. Learning rate: 1e-4, 1e-2,
5. Dropout Rate: 0.2, 0.5
6. Batch size: 32, 64, 128

3.4 Training process for AlexNet

One of our goals was to optimize the AlexNet model's performance on the CIFAR-10 dataset by tuning hyperparameters such as learning rate, weight decay, batch size, number of filters, fully connected layer size, and dropout rate, and testing different activation functions and pooling layers. The initial code to run the model with the RELU activation function (original AlexNet) was borrowed from a Kaggle tutorial on working with this architecture in TensorFlow (Vortexkol).

Several AlexNet models were created with different activation functions and pooling layers to establish a baseline. Specifically, models using ReLU, ELU, and Leaky ReLU activation functions were defined (Fig.1). After running those, ELU was the most successful according to the accuracy metric. Because of that, a variation of the ELU model then incorporated Average Pooling instead of Max Pooling to test its effect on classification accuracy (Fig.2). According to the results, the

model that used Max Pooling performed better (60.02% accuracy) and with less overfitting.

Each model was compiled using the sparse categorical cross-entropy loss function, a learning rate of 0.001, and the 'accuracy' metric. The models were then trained for 5 epochs each using the training dataset and validated against the test dataset. Finally, the training and validation loss, as well as the training and validation accuracy, were plotted for each model to visually assess their performance.

Following the baseline experiments, extensive hyperparameter tuning was conducted using Optuna to optimize the AlexNet model's performance. Initially, a function to create a model was defined to construct an AlexNet model with ELU activation, allowing for the adjustment of learning rate and weight decay. Optuna was then employed with an objective function that suggested values for learning rate (log-uniform between 1e-4 and 1e-1), weight decay (log-uniform between 1e-5 and 1e-2), and batch size (categorical choice of 32, 64, or 128). The model was created using these suggested hyperparameters, trained for 5 epochs, and the best validation accuracy was returned to Optuna for optimization. This process was repeated for 5 trials to explore different hyperparameter combinations, and the best-performing set of parameters along with the corresponding validation accuracy were printed. Subsequently, a more comprehensive function for model creation was introduced, expanding the tunable hyperparameters to include the number of filters in the first convolutional layer (num_filters1), the number of filters in the second and third convolutional layers (num_filters2), the size of the fully connected layers (fc_size), the dropout rate (dropout_rate), the learning rate (learning_rate), and the weight decay (weight_decay). The Optuna objective function was adapted to suggest values for these additional hyperparameters, along with batch size. The remaining optimization steps remained consistent: the model was built with the suggested parameters, trained for 5 epochs, and the best validation accuracy was returned, enabling Optuna to identify the optimal hyperparameter configuration for maximizing validation accuracy. The optimal hyperparameters for each activation function are reported in Figure 4.

The last step in trying to build the best model based on AlexNet's general architecture was to train models based on the highest-performing parameters. As a result of that, out of the 3 models we trained at the end, we got the best validation accuracy of 69.93%. That result was achieved using the ELU activation function, and the respective parameters listed in Figure 4. The validation and training accuracy and loss were graphed and shown in Figure 4.1.

activation function	accuracy	loss	validation accuracy	validation loss
RELU	0.5308	1.306	0.5901	1.1562
ELU	0.6002	1.137	0.6357	1.0468
Leaky RELU	0.5324	1.3084	0.5811	1.1628

Fig.1: Initial testing of activation functions with otherwise unchanged AlexNet architecture.

pooling type	accuracy	loss	val accuracy	val loss
Max	0.6002	1.137	0.6357	1.0468
Average	0.451	1.5318	0.4954	1.3893

Fig.2: Testing results with ELU activation function with different types of pooling for AlexNet architecture (everything else unchanged).

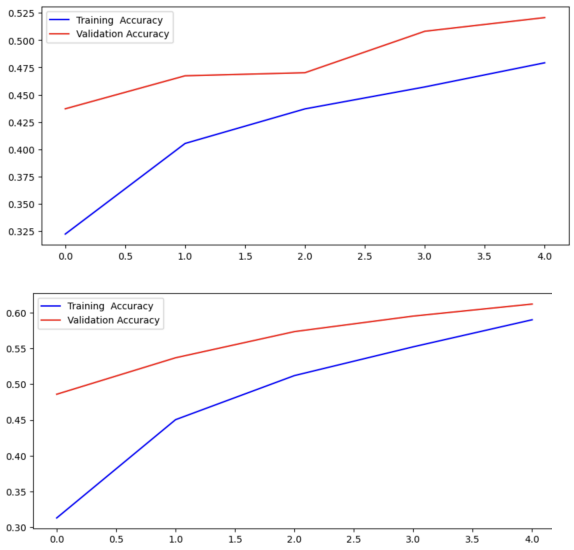

Fig.3: A visual of training vs. validation accuracy for AlexNet with ELU activation function, comparing Average Pooling (top) and Max Pooling (bottom).

activation function	number of filters	fc size	dropout rate	learning rate	val accuracy
ELU	64, 256	256	0.205	0.003	0.6993
RELU	32, 64	512	0.397	0.01	0.6542
Leaky RELU	128, 128	256	0.279	0.005	0.6779

Fig.4: Best hyperparameters found for each model with AlexNet architecture. The best batch size and weight decay were found to be 64 and 0.0003 for all, respectively.

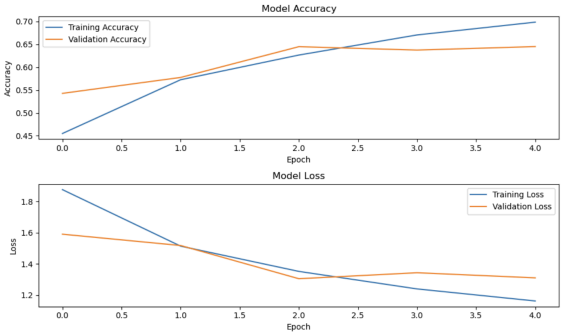

Fig.4.1: The validation & training accuracy and loss graphed for the best model with the ELU activation function (based on AlexNet architecture).

3.5 Training Process for GoogleNet:

For our second experiment, we hyper-tuned parameters for our CNN model built using GoogleNet. For baseline, various GoogleNet models were trained with different activation functions, including ReLU, ELU, and Leaky ReLU. The models were trained using an SGD optimizer and sparse categorical cross-entropy loss. ReLU performed the best after initial testing, so additional experiments were conducted with ReLU and different pooling methods. We tested Max Pooling and Average Pooling to observe their impact on classification accuracy.

Following these initial experiments, extensive hyperparameter tuning was performed using Optuna to optimize the performance of GoogleNet. We defined a function in which learning rate, weight decay, number of filters, and batch size could be optimized. Optuna suggested learning rate (log-uniform between 1e-4 to 1e-1), weight decay (log-uniform between 1e-5 to 1e-2), dropout rate (0.2 or 0.5), and batch size (32, 64, 128). The model was implemented with these suggested hyperparameters, trained for 5 epochs, and the peak validation accuracy was passed back to Optuna for optimization. We performed multiple trials to find the best combination of hyperparameters, including learning rate, weight decay, dropout rate, and batch size. The tuning process revealed that a learning rate of approximately 0.00278, 256 filters, a dropout rate of around 0.4535, and a batch size of 64 provided the best accuracy for ReLU-based models. Interestingly, models using ELU and Leaky ReLU had lower optimal filter sizes and batch sizes, which suggests that different activation functions may require different network architectures for best performance.
This was done many times for numerous trials to optimize across different combinations of hyperparameters and find the optimal configuration.

Activation Function	Learning rate	Number of filters	Dropout rate	Batch Size
ReLU	~0.00278	256	~0.4535	64
Leaky ReLU	~0.00138	64	~0.2513	32
ELU	~0.00259	64	~0.2525	128

Fig.5: Optimal Hyperparameters for the 3 activation functions using Optuna for GoogleNet.

When testing the models with the three activation functions, after choosing the best hyperparameters, we got the following accuracies.

Activation Function	Accuracy
ReLU	74.25%
Leaky ReLU	72.31%
ELU	73.23%

Fig. 6: Accuracies for GoogleNet with different activation functions with optimal hyperparameters

To further analyze our model's performance, we visualized its predictions on the CIFAR-10 test set. The model correctly classified most images, but certain classes were more challenging than others. Specifically, the models occasionally confused automobiles with trucks and birds with airplanes. This could be due to similarities in image structure, where certain background features might mislead the classifier (Fig. 6-8).

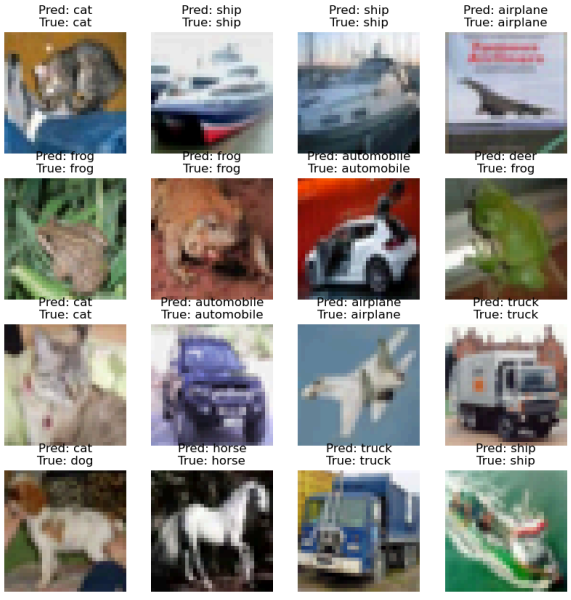

Fig. 6: GoogleNet ReLU prediction

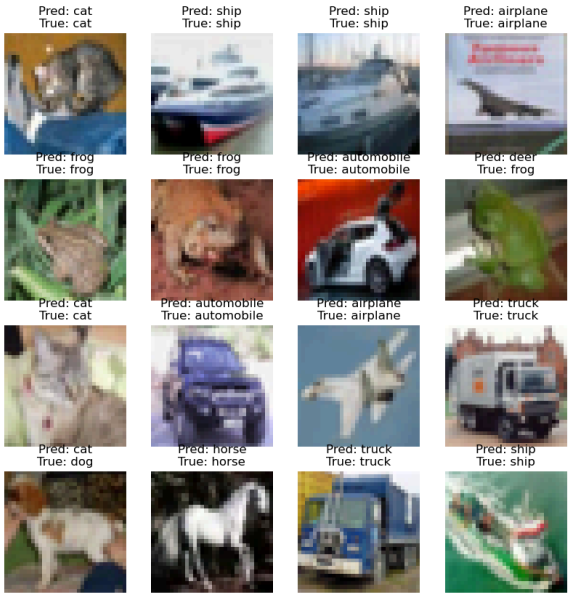

Fig. 7: GoogleNet Leaky ReLU predictions

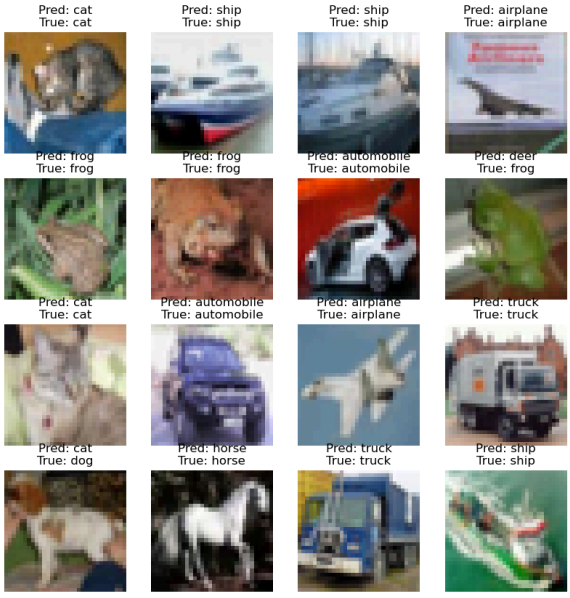

Fig. 8: GoogleNet ELU predictions

4. Discussion & Conclusion

4.1 Discussion

One significant limitation of our study was the relatively low number of epochs used for training; we trained our models for only 5 epochs due to computational constraints. Increasing the number of epochs could lead to improved model performance and more stable results, allowing the models to achieve higher classification accuracy, demonstrate better

generalization capabilities, and reveal more differences between AlexNet and GoogleNet architectures. Furthermore, the available processing power significantly limited our ability to conduct more extensive experiments, restricting training time, narrowing the range of hyperparameters we could explore, and limiting model complexity. By addressing these limitations, future research could provide more comprehensive insights into the performance and behavior of AlexNet and GoogleNet architectures on the CIFAR-10 dataset.

4.2 Conclusion

One thing noticed in our experiments was how GoogleNet always outperformed AlexNet on CIFAR-10. While our best AlexNet models achieved about 69.93% accuracy after hyperparameter tuning, GoogleNet achieved 74.25% accuracy when parameters were optimized. This is testimony to the advantage of having deeper structures that include inception modules that facilitate extraction at many scales of features. GoogleNet's use of greater than one filter size simultaneously in parallel almost certainly contributed to increasing its classification performance, especially when utilizing a large, heterogeneous dataset like CIFAR-10.

The other important variation was in the networks' behavior regarding hyperparameter tuning. AlexNet was very sensitive to activation function choice, while GoogleNet's performance was less sensitive to different activation functions. While batch size had a large impact on AlexNet, there was slightly less influence on GoogleNet's performance, which suggested that the deeper network had more capacity to generalize.

Through our experiments, we found that GoogleNet with ReLU activation, Max Pooling, and optimized hyperparameters significantly outperformed other variations. The model reached an accuracy of 74.25% after tuning, demonstrating the effectiveness of deep architectures on image classification tasks. Compared to AlexNet, GoogleNet was more stable across different hyperparameters and achieved higher accuracy, reinforcing the benefits of inception modules and increased depth. Further improvements could be achieved through additional data augmentation and fine-tuning training duration.

References

1. Akwaboah, A. D. (2019, November). (PDF) convolutional neural network for CIFAR-10 Dataset Image Classification. https://www.researchgate.net/publication/337240963_Convolutional_Neural_Network_for_CIFAR-10_Dataset_Image_Classification
2. Klingler, N. (2024, December 30). Alexnet: A revolutionary deep learning architecture. viso.ai. https://viso.ai/deep-learning/alexnet/
3. Vortexkol. (2020, September 21). Alexnet CNN Architecture on TensorFlow.Kaggle. https://www.kaggle.com/code/vortexkol/alexnet-cnn-architecture-on-tensorflow-beginner
4. Ballester, P., & Araujo, R. (2016). On the Performance of GoogLeNet and AlexNet Applied to Sketches. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1). https://doi.org/10.1609/aaai.v30i1.10171
5. He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 10). Deep residual learning for image recognition. [1512.03385v1] Deep Residual Learning for Image Recognition. https://export.arxiv.org/abs/1512.03385v1