COGS 181 Final report

Performance Comparison of CNN for CIFAR-10 Classification with Limited Training Time

Zening Wang
A161414249
2024/3/22

1. Abstract

This report presents an empirical study on various convolutional neural network (CNN) architectures and hyperparameters for image classification tasks using the CIFAR-10 dataset with limited training time. Something new of this study is that the training time is limited to 10 epochs, mimicking real-world scenarios where rapid model development and deployment are essential. The study delves into the impact of different network architectures, optimization algorithms, and hyperparameter settings on the accuracy and training efficiency of the models. Through a series of experiments involving baseline CNN models, custom architectures, and established architectures like AlexNet and LeNet, we assess the effectiveness of each approach and provide insights into the factors influencing model performance. We also discuss the difference in accuracy between each different class in CIFAR-10 dataset and try to gain a deeper understanding of model behavior.

2. Introduction

Image classification plays a fundamental role in computer vision, with applications ranging from object recognition to medical image analysis. Convolutional neural networks (CNNs) have become indispensable tools for image classification due to their ability to automatically learn hierarchical features from raw pixel data. However, designing an effective CNN architecture and selecting suitable hyperparameters are crucial steps in achieving high classification accuracy.

This report aims to explore the impact of different CNN architectures and hyperparameters on the performance of image classification models under the constraint of limited training time. We seek to understand how architectural choices, such as the number of layers and the size of convolutional filters, and hyperparameter settings like learning rate and batch size, influence model accuracy and training efficiency. Furthermore, we investigate how each of these factors interacts when training time is limited, providing insights into efficient model development under time constraints.

3. Method

In this study, we employ various CNN architectures and hyperparameters to conduct image classification tasks using the CIFAR-10 dataset. The CIFAR-10 dataset consists of 60,000 32x32 color images across 10 classes, making it suitable for evaluating classification models. After we set the epochs to 10, the training time for each model is only about 2 minutes when activating GPU. Thus, this is a task specifically to examine which combinations of architectures and hyperparameters would perform best under such conditions. We will compare the training time, overall accuracy, training loss and accuracy on individual classes.

CNN Architectures:

1. **Baseline CNN Models:** We start with simple CNN architectures consisting of convolutional layers followed by pooling layers and fully connected layers for classification. These baseline models serve as benchmarks for comparing the performance of more complex architectures.
2. **Custom Architectures:** We design and experiment with custom CNN architectures, varying the number of convolutional layers, filter sizes, and the depth of the networks to understand their impact on model accuracy and training time.
3. **Established Architectures:** We also evaluate well-known CNN architectures such as AlexNet and LeNet, which have been widely used in image classification tasks. These architectures often have deeper networks with intricate layer configurations and regularization techniques.

Hyperparameters:

1. **Learning Rate:** We investigate different learning rates (e.g., 0.01, 0.001) to determine their effect on model convergence and accuracy. A suitable learning rate is crucial for achieving optimal performance without overfitting or underfitting.
2. **Batch Size:** The batch size affects the speed and stability of training. We experiment with batch sizes ranging from 64 to 128 to analyze their impact on training time and model accuracy.

3. **Optimization Algorithms:** We compare the performance of optimization algorithms such as SGD with momentum and Adam optimizer. Each optimizer has unique properties that can influence model convergence and generalization.

4. **Regularization Techniques:** We apply regularization techniques like dropout to prevent overfitting and improve model generalization. By varying dropout rates, we assess their impact on model performance under limited training time.

4. Experiment

Data Preparation: We preprocess the CIFAR-10 dataset by applying standard transformations such as normalization o enhance model robustness.

Model Training: Each CNN architecture is trained using the specified hyperparameters and optimization algorithm. Training is limited to 10 epochs to simulate time-constrained model development.

Evaluation: We evaluate the trained models on a separate test set to measure classification accuracy. Additionally, we analyze the accuracy distribution across different classes in the CIFAR-10 dataset to identify potential challenges and areas for improvement.

Nine trials were evaluated in this study to understand their performance under the constraint of limited training epochs. Here's a breakdown of the architectures explored:

Trials 1-3: These represent custom designed CNNs with varying depths (number of layers) and the number of filters within each convolutional layer. This allows us to assess the impact of model complexity on performance within the limited training time.

[Notice for Tria2, I utilized the existing code from HW4, see the details at reference section]

Trial 4: AlexNet: This experiment implements the well-known AlexNet architecture, a proven CNN for image classification. Including AlexNet provides a benchmark for established architectures and allows comparison with custom designs.

Trial 5: LeNet: This trial explores LeNet, another established CNN architecture with a simpler design compared to AlexNet. Evaluating LeNet helps us understand the trade-off between model complexity and performance under limited training.

[Notice for Trial4-5, I utilized the existing code from public Github code resources, see the details at reference section]

Trials 6-9: These trials are variations of Trial 1 (the custom 3-layer CNN). We modify hyperparameters like the learning rate and batch size to investigate their influence on performance under limited training.

Here is the results of my trails:(saved in process.txt)

Trial	Base Architecture	Conv Layers	FC Layers	Activation	Learning Rate	Batch Size	Optimizer	Training Time	Accuracy
1	Custom	3	2	ReLU	0.01	128	SGD (0.9)	116.66	74.98%
2	Custom (HW4)	2	3	ReLU	0.01	128	SGD (0.9)	114.62	61.47%
3	Custom (deeper)	3	4	ReLU	0.01	128	SGD (0.9)	130.33	72.27%
4	AlexNet	5	3	ReLU	0.01	128	SGD (0.9)	133.63	70.75%
5	LeNet	2	3	Tanh	0.01	128	SGD (0.9)	114.23	60.94%
6	Trial 1 (tuned)	3	2	ReLU	0.01	128	Adam	117.34	56.05%
7	Trial 1 (lower LR)	3	2	ReLU	0.001	128	SGD (0.9)	123.64	53.26%
8	Trial 1 (smaller BS)	3	2	ReLU	0.001	64	SGD (0.9)	117.99	62.54%
9	Trial 1 (tuned)	3	2	ReLU	0.01	64	SGD (0.9)	122.62	75.80%

Impact of Network Architecture:

- **Custom Architectures:** The custom 3-layer CNN (Trial 1) achieved a good balance between accuracy (74.98%) and training time (116.66 seconds) within the 10-epoch constraint. Increasing the depth (Trial 3) resulted in a slight decrease in accuracy (72.27%) with a longer training time (130.33 seconds). A smaller custom architecture (Trial 2) had the fastest training time (114.62 seconds) but yielded the lowest accuracy (61.47%). These findings suggest that a moderate-sized custom CNN can achieve reasonable performance under limited training time.

Impact of Hyperparameters:

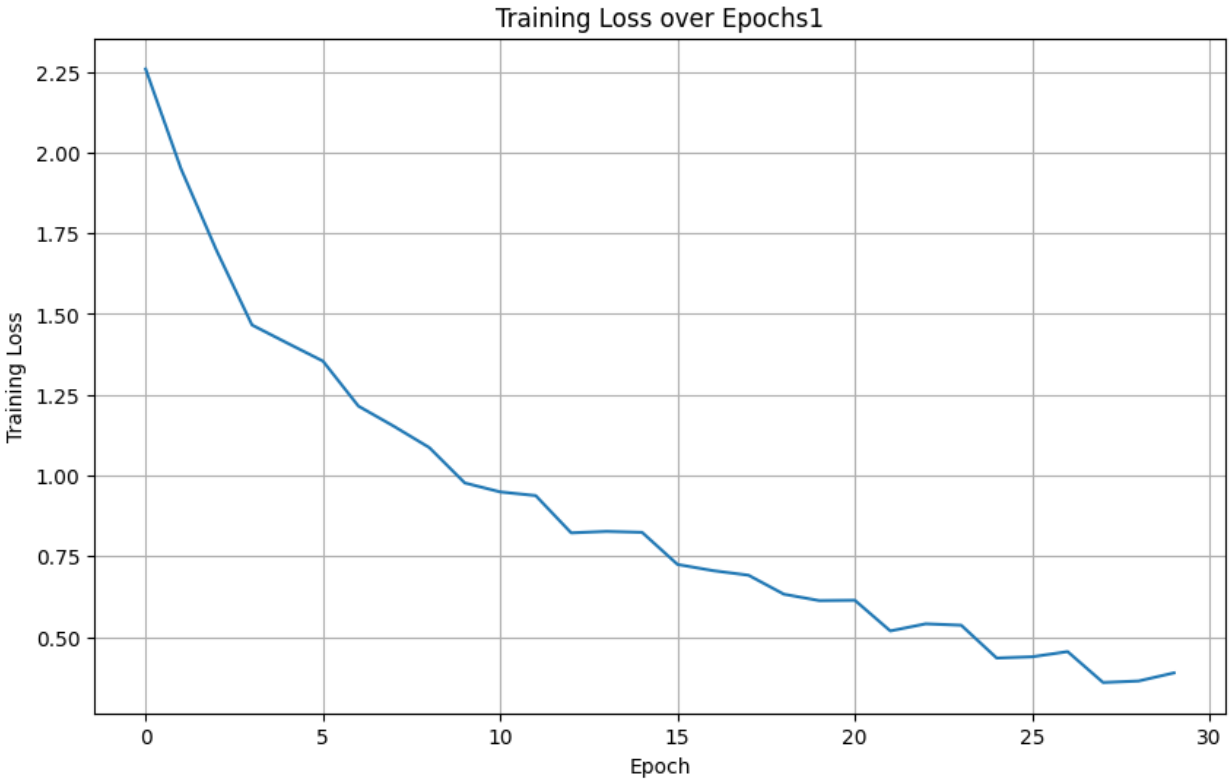
- **Learning Rate:** Reducing the learning rate from 0.01 to 0.001 (Trial 7) led to a significant drop in accuracy (74.98% to 53.26%) despite a small increase in training time. This suggests that a learning rate of 0.01 is better suited for these architectures within the limited training window. The reason is that although smaller learning rate could possibly better fir of the model, but the model hasn't fully learnt all data under limited learning time.
- **Batch Size:** Decreasing the batch size from 128 to 64 (Trial 8) improved accuracy (from 53.26% to 62.54%) while maintaining a similar training time. This is likely due to the benefits of SGD with smaller batches for updating weights with more frequent gradients. Yet the accuracy is still smaller than Trial1 indicating even with more batch size, model hasn't fully learnt all data under limited learning time.
- **Optimizer:** Replacing SGD with Adam (Trial 6) did not improve accuracy (56.05%) compared to SGD with momentum (74.98% in Trial 1). In this case, SGD seems to be a more effective optimizer for these CNNs under the given constraints.

Additional Observations:

- **Established Architectures:** AlexNet (Trial 4) and LeNet (Trial 5) achieved moderate accuracy (70.75% and 60.94% respectively) within the limited training time. While these

architectures are not specifically optimized for the 10-epoch constraint, they provide a baseline for comparison with custom designs.

Also, we have graphed train loss lots for each of the trails. Fortunately, they all have similar and reasonable trends. So we would like to skip this part and show only the first example here. (see the rest in figs)



Here is the result of different trails on different class in CIFAR-10:

Trial	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
1	75%	75%	75%	64%	66%	63%	66%	84%	87%	76%
2	44%	67%	48%	47%	55%	51%	86%	76%	81%	69%
3	58%	78%	78%	70%	51%	45%	61%	80%	87%	82%
4	68%	67%	66%	41%	66%	48%	75%	60%	87%	89%
5	58%	82%	60%	50%	51%	27%	66%	44%	62%	56%
6	55%	71%	42%	23%	59%	21%	75%	48%	62%	56%
7	41%	75%	30%	32%	18%	36%	58%	52%	75%	53%
8	69%	78%	29%	42%	52%	50%	66%	51%	72%	60%
9	83%	82%	65%	61%	74%	61%	78%	76%	82%	83%
mean	**61%**	**75%**	**55%**	**48%**	**55%**	**45%**	**70%**	**63%**	**77%**	**69%**

1. **Variation in Class Accuracy**: There is considerable variation in the accuracy of different classes across trials. For example, in Trial 9, the model achieved high accuracy for

classes like Airplane, Automobile, Ship, and Truck, while it struggled with classes like Bird and Cat.

2. **Effect of Architecture Changes**: Changing the architecture, such as in Trial 2 (Custom, smaller), resulted in noticeable changes in class accuracies. For instance, the accuracy for classes like Airplane, Automobile, and Frog improved compared to Trial 1, but there was a decrease in accuracy for classes like Cat and Deer.
3. **Impact of Hyperparameters**: Adjusting hyperparameters, such as learning rate and batch size, led to varying outcomes. Lowering the learning rate in Trial 7 resulted in decreased overall accuracy and lower accuracies for most classes compared to Trial 1.
4. **Lower Performing Classes :** All trials showed lower accuracy on Cat, Deer, Dog, and Horse compared to other classes. This suggests these classes might benefit from techniques like data augmentation or class-specific hyperparameter tuning to improve accuracy. And this might also suggest that CNN still have difficulty in identifying animals compared vehicles in limited training time.

5. Conclusion

Overall, these experiments highlight the importance of considering both network architecture and hyperparameter tuning for achieving high accuracy on specific classes within limited training time. Specifically, Trial 9 achieved the best performance on most classes. It used a custom architecture with 3 convolutional layers, 2 fully connected layers, ReLU activation, learning rate of 0.01, batch size of 64 and SGD optimizer with momentum 0.9. This tells us that in limited training, such choices might be better than complicated parameters and structures. Lastly, we also find that CNN is not good at identifying animals specially of small sizes.

6. Reference

Code link: https://github.com/Zening-W/Performance-Comparison-of-CNN-for-CIFAR-10-Classification-with-Limited-Training-Time

https://github.com/soapisnotfat/pytorch-cifar10/blob/master/models/LeNet.py

https://github.com/soapisnotfat/pytorch-cifar10/blob/master/models/AlexNet.py

https://en.wikipedia.org/wiki/Convolutional_neural_network

https://sites.google.com/view/ucsd-cogs-181-winter-2024/assignments/final-project

COGS181 winter 2024 Homework 4