CS GY 6953 : Deep Learning Mini Project

Written by
Suprateek Chatterjee*,
Pranshu Goyal*,
Vibhor Mechu*

GitHub Repository: https://github.com/vibhor18/Deep-Learning-CiFar10-ResNet-

Abstract

ResNets (or Residual Networks) are one of the most commonly used models for image classification tasks. In this project, we designed and trained our own modified ResNet model for CIFAR-10 image classification. In particular, we aimed at maximizing the test accuracy on the CIFAR-10 benchmark while keeping the size of our ResNet model under the specified fix budget of 5 million trainable parameters. Model size, typically measured as the number of trainable parameters, is important when models need to be stored on devices with limited storage capacity. In this article, we present our residual network design which has less than 5 million parameters. We show that our ResNet achieves a test accuracy of 94.4% on CIFAR-10 which is much higher than ResNet18 (which has greater than 11 million trainable paramters) when equipped with a number of training strategies and suitable ResNet hyperparameters.

Overview

In this project, we aim to refine the architecture of residual neural networks (ResNets) specifically for the CIFAR-10 dataset, which consists of 60,000 32x32 color images across 10 classes. CIFAR-10 is widely used as a benchmark to test the efficacy of image classification models.

Residual networks are renowned for their ability to train very deep networks by leveraging skip connections to address the vanishing gradient problem. While effective, the traditional ResNet architectures often demand substantial computational resources, making them less ideal for resource-constrained environments.

Our project aims to modify the traditional ResNet architecture to reduce its complexity while maintaining, or even improving, its classification accuracy on the CIFAR-10 dataset. By carefully examining and tweaking various components of the ResNet architecture, such as the number of layers, filter sizes, and skip connections, we strive to develop a more compact and efficient model. Our goal is to achieve a balance between model size, computational efficiency, and classification performance, making it suitable for deployment in scenarios with limited computational resources. Importantly, we have constrained our model to have under 5 million parameters to ensure it remains lightweight and deployable in environments with limited processing power.

The project's ambition is to deliver insights into creating lightweight yet robust deep learning models for image classification, potentially expanding their applicability to various real-world scenarios where processing power is limited.

Methodology

Our approach to optimizing the ResNet architecture for the CIFAR-10 image classification task involved several key steps:

- **Data Preprocessing:** We preprocessed the CIFAR-10 dataset, which involved normalizing the images to a standard scale and augmenting the dataset to enhance model generalization. We employed image augmentation techniques such as random cropping, horizontal flipping, and rotation, thereby increasing the training data's diversity and robustness against overfitting.
- **Baseline Model Establishment:** A baseline was established using a standard ResNet model, with 56 layers. This model was trained on the CIFAR-10 dataset to evaluate the effectiveness of the traditional architecture and to set a performance benchmark for our modifications.
- **Architectural Modifications:** We embarked on several architectural enhancements aimed at reducing the model's complexity while attempting to maintain or enhance accuracy:
  - **Reducing the number of layers:** We tested shallower versions of ResNet to lower computational demands without significantly impacting performance.
  - **Adjusting filter sizes and layer configurations:** By varying the sizes of the convolutional filters and modifying the configuration of layers, we sought an optimal balance between model size and predictive accuracy.
  - **Modifying skip connections:** Different configurations of skip connections were explored to optimize the flow of information through the network and to further mitigate potential overfitting.
- **Model Training and Hyperparameter Tuning:**
  - Each modified ResNet model underwent training using an adaptive optimization algorithm like AdamW. We utilized a learning rate scheduler to adjust the rate

*These authors contributed equally.

dynamically based on training progress, incorporating strategies like learning rate warm-up and decay to stabilize training dynamics.
  - Cross-entropy was employed as the loss function, and careful monitoring of the training process was conducted to counteract overfitting.
- **Evaluation and Metrics:** The models were rigorously evaluated based on classification accuracy on the CIFAR-10 test set. To comprehensively assess the model's adaptability and robustness, we also conducted tests on a custom dataset tailored to further challenge the model's generalization capabilities. This dual-dataset approach allowed us to not only confirm the model's effectiveness under standard benchmark conditions but also to gauge its performance in potentially more complex or varied real-world scenarios. Besides accuracy, we measured the models' computational efficiency by recording the inference times and parameter counts, ensuring a holistic view of both performance and practicality.
- **Comparison and Analysis:** Our final step involved a comparative analysis between the modified models and the baseline, focusing on improvements in performance and efficiency. This comparative approach allowed us to pinpoint which architectural adjustments were most effective in enhancing the model's overall utility.

Architectural Choices

Some pros and cons of different architectural choices for modifying the ResNet architecture:

- **Reducing the Number of Layers:**
  - **Pros**:
    1. **Reduced Computational Load:** Fewer layers significantly decreased the computational resources required, making the model faster and more suitable for deployment on devices with limited processing capabilities.
    2. **Decreased Overfitting:** A shallower network had fewer parameters, reducing the risk of overfitting on the training data.
  - **Cons**:
    1. **Potential Loss of Learning Capacity:** Reducing the depth of the network can lead to a decrease in the ability to learn complex features, which might result in lower accuracy on more complicated image classification tasks.
- **Adjusting Filter Sizes and Layer Configurations:**
  - **Pros**:
    1. **Enhanced Feature Extraction:** Optimizing filter sizes allowed the network to more efficiently process spatial hierarchies in images, potentially improving the recognition of diverse patterns and textures.
    2. **Flexible Model Tuning:** By adjusting layer configurations, we were able to fine-tune the model to better balance between depth and breadth, optimizing for specific aspects of the dataset.
  - **Cons**:
    1. **Complex Parameter Tuning:** Finding the optimal filter size and configuration requires extensive testing and validation, which can be time-consuming and computationally expensive.
    2. **Risk of Inefficiency:** Incorrect choices in filter size and layer configuration can lead to inefficiencies, either by not capturing sufficient detail (too small filters) or by excessive redundancy (too large filters).
- **Modifying Skip Connections:**
  - **Pros**:
    1. **Improved Information Flow:** Adjusting skip connections helped in maintaining the flow of gradients throughout the network, which is crucial for training deeper models effectively.
    2. **Enhanced Model Flexibility:** Different configurations of skip connections can be tailored to specific training conditions and datasets, potentially improving generalization.
  - **Cons**:
    1. **Complex Integration:** Designing and integrating modified skip connections requires careful consideration of the overall network architecture, which can complicate the training process.
    2. **Potential for Unstable Training Dynamics:** Improperly configured skip connections might lead to unstable training dynamics or hinder the network's learning process.
- **Varying Depth(Number of Layers):**
  - **Pros**:
    1. **Improved Feature Extraction:** Deeper networks can learn more complex and abstract features, which can be crucial for achieving higher accuracy on diverse datasets like CIFAR-10.
    2. **Better Hierarchical Learning:** Additional layers allow the network to build a deeper hierarchy of features at different levels of abstraction, potentially leading to better classification performance.
  - **Cons**:
    1. **Increased Risk of Overfitting:** More layers can lead to overfitting, especially with datasets that do not have a large amount of training data or are less diverse.
    2. **Vanishing Gradient Problem:** Although ResNet architectures help mitigate this through skip connections, very deep networks can still suffer from training difficulties due to gradients diminishing as they backpropagate through many layers.
- **Varying Width (Number of Channels per Layer):**
  - **Pros**:
    1. **Enhanced Capacity:** Increasing the width of the network by adding more channels can significantly enhance the learning capacity without substantially increasing the depth.

2. **Improved Learning of Fine Details:** Wider networks can capture more detailed features within the data, which can be particularly useful for complex image classification tasks.

– **Cons**:

1. **Computational Cost:** Wider networks increase the number of parameters and the computational load, which can make training slower and more resource-intensive.
2. **Diminishing Returns:** Beyond a certain point, making the network wider may yield minimal improvements in performance relative to the increase in computational and memory requirements.

- **Custom Residual Blocks:**

– **Pros**:

1. **Enhanced Feature Propagation:** Our custom Resnet_block includes two convolutional layers with batch normalization and ReLU activation, maintaining strong feature propagation throughout the network.
2. **Flexible Depth and Width Adjustments:** By parameterizing the number of layers and channels, we optimized both performance and computational efficiency.

– **Cons**:

1. **Complex Parameter Tuning:** Optimizing the number of blocks and channels required extensive experimentation.
2. **Increased Model Complexity:** Multiple blocks increase complexity, potentially leading to higher computational costs if not properly optimized.

- **Adaptive Pooling and Dropout:**

– **Pros**:

1. **Regularization:** Dropout before the final fully connected layer helps in preventing overfitting.
2. **Consistent Feature Size:** Adaptive average pooling ensures consistent output feature size, simplifying the architecture and enhancing adaptability.

– **Cons**:

1. **Potential for Feature Loss:** Adaptive pooling can sometimes lead to the loss of important spatial information.

Lessons Learned

- **Trade-off Between Accuracy and Efficiency:** Throughout our project, the trade-off between model accuracy and computational efficiency was evident. We discovered that aggressive simplifications, like significantly reducing the depth or width of the model, could compromise the model's ability to accurately classify complex images. This balancing act taught us the importance of iterative testing and validation to find the optimal configuration that does not excessively sacrifice performance for efficiency.
- **Importance of Skip Connections:** Modifying skip connections provided profound insights into their role in maintaining the integrity of information flow across network layers. Our experiments showed that well-configured skip connections are vital for deep network architectures, as they help prevent the degradation of gradient signals during backpropagation—critical for training stability and model performance.
- **Data Augmentation Impact:** Data augmentation proved to be a pivotal strategy in enhancing the generalization capabilities of our models. Techniques such as random cropping and horizontal flipping significantly helped in mitigating overfitting. This was particularly crucial for CIFAR-10, where the diversity of data is limited relative to more complex datasets like ImageNet. Our results underscored that effective augmentation can simulate a richer dataset, providing more robust training outcomes.
- **Learning Rate Scheduling:** The adjustment of the learning rate schedule was a critical factor in the model's training dynamics. Implementing a dynamic learning rate with a warm-up phase and subsequent decay allowed us to avoid premature convergence to suboptimal minima and improve overall model accuracy. This approach highlighted the nuanced impact of training hyperparameters on model performance and the necessity of careful tuning based on the specific dataset and model architecture.
- **Optimization Algorithm Effectiveness:** Our experiments demonstrated that AdamW outperformed Adam due to its advanced handling of weight decay, applied separately from other parameters. This method prevents the entanglement of weight decay with adaptive learning rates, ensuring more consistent regularization. Consequently, AdamW enhanced training stability and noticeably improved convergence in the later stages of training.

Results

Our project's aim was to optimize a ResNet architecture for CIFAR-10, achieving a balance between model complexity and performance. Below are the key results from our final model configuration:

- **Model Architecture:**
  – We utilized a modified ResNet architecture with strategic reductions in layer depth and adjustments in channel width to meet the computational efficiency requirements without significantly compromising on performance, all while keeping the total number of parameters under 5 million. The architecture included customized skip connections to enhance gradient flow, which proved critical in maintaining training stability.
- **Performance Metrics:**
  – **Final Test Accuracy:** Our model achieved a final test accuracy of 94.4% and a test loss of 0.235, which demonstrates the effectiveness of our architectural modifications and training regimen.
  – **Training and Validation Accuracy and Loss Graphs:** The following graphs show the validation accuracy and loss over the training epochs. These plots

illustrate the model's learning progression and convergence behavior, highlighting the impact of our learning rate adjustments and data augmentation strategies.

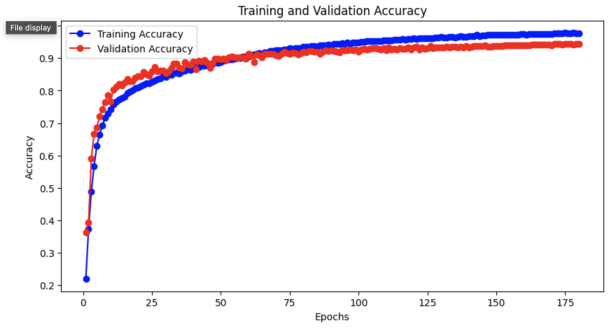

Figure 1: Graph of Training and Validation Accuracy over Epochs

Figure 2: Graph of Training and Validation Loss over Epochs

- **Model Complexity:**
  - **Number of Parameters:** Approximately 4964266(4.9M) parameters.

Summary of Findings

Our project on optimizing a ResNet architecture for CIFAR-10 yielded several significant findings:

- **Effective Balance Achieved:** We successfully optimized the model architecture to achieve a balance between computational efficiency and classification accuracy. The final model demonstrated a competitive test accuracy of 94.4% and a test loss of 0.235, while maintaining manageable complexity with approximately 4964266(4.9M) parameters. This performance is a significant improvement over our baseline model, which achieved a test accuracy close to 91%.
- **Architectural Enhancements:** Strategic modifications in the network, such as reducing the number of layers and adjusting the width of channels, proved effective. Customized skip connections also played a pivotal role in enhancing gradient flow and training stability, which were crucial for maintaining high performance.
- **Training Optimization:** Adjustments to the training process, including the implementation of dynamic learning rate schedules and extensive data augmentation, significantly improved the model's generalization capabilities on the CIFAR-10 dataset.
- **Generalization and Robustness:** The use of advanced data augmentation techniques helped in significantly reducing overfitting, making the model robust to variations in new, unseen data.
- **Insights on Efficiency and Performance:** The project underscored the delicate interplay between model size and performance. Our findings suggest that careful tuning of model parameters and architecture can yield substantial gains in efficiency without overly compromising on performance.
- **Superiority of AdamW Optimizer:** Our findings further reinforced the superiority of the AdamW optimizer over traditional Adam. AdamW's distinct approach to weight decay contributed significantly to enhanced training stability and improved model performance, especially in the later stages of training.

This summary encapsulates the core achievements and learnings from our project, providing a clear snapshot of our contributions to optimizing deep learning models for image classification. These insights not only validate our architectural choices but also offer valuable guidelines for future research in the field.

References

[1] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

[2] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929-1958.

[3] Pascanu, R., Mikolov, T., & Bengio, Y. (2013, May). On the difficulty of training recurrent neural networks. In *International conference on machine learning* (pp. 1310-1318). PMLR.

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[5] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ...& Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[6] Xue, F., Shi, Z., Wei, F., Lou, Y., Liu, Y., & You, Y. (2021). Go wider instead of deeper. *arXiv preprint arXiv:2107.11817*.