CS GY 6953 : Deep Learning Project

Written by :
Mukesh Durga
Tanaya Pawar
Tejaswini Ojha

GitHub Repository: `https://github.com/tanaya09/CS6953-Deep-Learning-Project`

Abstract

ResNets are one of the most commonly used models for image classification. In this project we have the architecture, and outcomes of a deep learning model we trained using PyTorch on the CIFAR-10 dataset. We customized a version of ResNet-18, which we refer to as "narrow ResNet," by reducing the number of channels, tweaking batch normalization settings, and incorporating dropout to fine-tune the balance between model complexity and computational efficiency. The training process included specific data augmentations and normalizations to boost the model's robustness and accuracy. Our results showed high accuracy, which suggests effective learning, though there was a slight tendency toward overfitting on the training data. Key takeaways from this project underscore the significance of having a solid architecture, employing continuous integration and testing, optimizing performance, and ensuring secure and scalable deployments in a microservices environment.

Project Overview

Our aim is to refine the architecture of residual Neural Networks (ResNets) for the CIFAR-10 dataset, which consists of 60,000 images across 10 classes.

Recognizing the inherent strengths of ResNets in facilitating deep network training through skip connections to combat the vanishing gradient problem, our initiative seeks to tailor these networks within a parameter constraint of 5 million to ensure suitability for resource-limited environments.

The project revolves around developing a deep learning model using a customized Narrow ResNet-18 architecture tailored for efficient image classification tasks on the CIFAR-10 dataset.

We also ensured consistent data preprocessing across training and inference stages by employing specific normalization values. This is complemented by a robust data augmentation strategy that includes techniques such as random cropping, flipping, and advanced color adjustments to ensure that the model is not only accurate but also robust against overfitting.

This project aspires to not only advance the computational efficiency and classification prowess of ResNets on CIFAR-10 but also to explore broader applications in scenarios where computing resources are scarce, providing a blueprint for deploying high-performance models in constrained settings.

Methodology

We developed a Narrow ResNet-18, a customized version of the traditional ResNet architecture designed for enhanced efficiency. It includes the following steps:

- **Data Preprocessing and Augmentation:** In the preprocessing phase, we applied a series of transformations to enhance the model's efficiency and performance. These included random cropping, horizontal flipping, Cutout augmentation, and normalization to prepare the CIFAR-10 dataset images for effective training and testing. The aim was to improve generalization and mitigate overfitting during model training.
- **Baseline Model Establishment:** A standard ResNet-18 model serves as the baseline, providing a benchmark for improvements and adjustments in the new architecture. Initial testing of the baseline model on the CIFAR-10 dataset establishes key performance metrics for comparison.
- **Architectural Modifications:** The architectural modifications to the traditional ResNet model were aimed at reducing complexity while enhancing efficiency. Our Narrow ResNet-18 reduces the channel width using a width multiplier of 0.5, significantly reducing the number of parameters. This adjustment, coupled with a reduction in initial convolutional channels to 32 and the introduction of dropout at a rate of 0.5, ensures the model remains lightweight yet capable of capturing complex features. Additionally, adaptive average pooling is applied before the final fully connected layer to improve the network's response to varying input sizes and maintain spatial hierarchy.
- **Model Training and Hyperparameter Tuning:** The model is trained using stochastic gradient descent (SGD) with momentum = 0.9 and weight decay = $5e^{-4}$ for effective learning.

- Hyperparameters such as learning rate, batch size, and number of epochs are tuned based on initial tests to find the optimal settings.
- Adopted Cosine Annealing with a warmup phase to optimize the learning rate progression over 250 epochs.
- Employed CrossEntropyLoss with label smoothing = 0.05 to mitigate overfitting.

- **Evaluation and Metrics:** Performance metrics such as accuracy, precision, recall, and F1-score are used to evaluate the model. Loss metrics, including cross-entropy loss, are monitored throughout the training process to gauge model improvement and convergence.
- **Comparison and Analysis:** The enhanced model's performance is compared against the baseline ResNet-18 model to assess the effectiveness of the architectural modifications. Analysis involves detailed examination of the model's performance on unseen data, discussing strengths and potential areas for improvement. Further comparisons are drawn with other state-of-the-art models to establish the competitive positioning of our model within the field.

Architectural Choices

This section outlines the key architectural modifications made in our Narrow ResNet-18 model for CIFAR-10, along with their advantages and trade-offs.

- **Narrow Residual Network (Reduced Width)**
  - **Pros**:
    - **Efficient Memory Usage:** By slimming down to a width multiplier of 0.5, we tried to effectively cut down the number of channels, easing the memory demands significantly.
    - **Faster Computation:** With fewer filters per layer, the model requires fewer operations, making it more efficient on lower-end GPUs.
  - **Cons:**
    - **Lower Representation Power:** A reduced number of channels limits the ability to learn complex patterns. It may struggle to capture complex patterns due to its reduced channel count.
    - **Potential Underfitting:** With fewer parameters, the model might struggle on more complex datasets beyond CIFAR-10.
- **Use of Cutout Data Augmentation**
  - **Pros:**
    - **Improved Generalization:** By masking out random sections of the input images, we compel the model to learn from a broader range of features, enhancing its generalization capabilities.
    - **Better Robustness:** This technique helps the model stay resilient against partially occluded or missing image regions in real-world scenarios.
  - **Cons:**
    - **Hyperparameter Sensitivity:** Choosing the number and size of cutout holes requires tuning, as excessive masking may lead to information loss.
    - **Increased Training Time:** Learning from masked images introduces more variance, which slows down convergence.
- **Cosine Annealing Learning Rate with Warmup**
  - **Pros:**
    - **Smooth Transition:** Starting with a warmup phase, we gently ramp up the learning rate during the initial 25 epochs, safeguarding against early training instabilities.
    - **Efficient Learning:** Cosine decay ensures that the learning rate decreases smoothly, leading to better convergence.
  - **Cons:**
    - **Manual Scheduling Required:** To extract the best performance, it's crucial to meticulously plan the warmup duration and the T_{max} settings for the learning rate decay.
    - **Dataset Dependency:** Depending on the complexity of the dataset, simpler models might achieve similar results even with a straightforward, fixed learning rate.

These choices finally reflect our strategic efforts to modify the model not only to fit within hardware constraints but also to adapt dynamically to training challenges, aiming for an optimal blend of speed, efficiency, and accuracy.

Lessons Learned

- **Efficient Architecture Design:** Throughout our project, we discovered that tweaking the number of channels per layer, finessing kernel sizes in skip connections, and integrating bottleneck layers made a world of difference. Not only did these adjustments keep us within the constraints of our 5-million-parameter limit, but they also bumped up our model's accuracy significantly.
- **Optimization Strategies:** After experimenting with various optimization techniques, we found that Stochastic Gradient Descent (SGD) with momentum, coupled with a cosine annealing learning rate scheduler, clearly outshone both Adam and RMSProp. Introducing a learning rate warm-up provided stability in the early stages of training, while strategies like weight decay and dropout were crucial in preventing overfitting.
- **Residual Connections Impact:** The project confirmed that skip connections are more than just a pathway as they are essential for mitigating vanishing gradients, speeding up convergence, and enabling the preservation of hierarchical features. Any experiments where we tweaked or removed these connections often led to noticeable drops in performance.
- **Data Augmentation and Regularization:** Using techniques like random cropping, horizontal flipping, and cutout significantly enhanced model generalization. Batch normalization stabilized training and ensured robust performance.
- **Systematic Experimentation and Insights:** Ablation studies revealed the impact of tuning kernel sizes, batch

normalization momentum, and pooling layers. Teacher-student distillation and quantization showed potential efficiency gains but required further tuning.

Results

Our project's main aim was to modify the ResNet architecture for CIFAR-10, achieveing a balance between both model complexity and performance. Below are the key results of our findings:

- **Model Performance and Efficiency:** We managed to fine-tune our model to achieve a test accuracy of **95.53%** while maintaining only **2,797,610** trainable parameters, staying well below the 5-million limit.
- **Training Stability and Convergence:** Training completed in **250 epochs**, but our strategic use of warm-up learning rate techniques really paid off by smoothing early convergence. The introduction of a cosine annealing scheduler further refined our optimization process, which resulted in enhanced generalization.
- **Comparative Analysis and Optimizations:** Parameter-efficient modifications led to improved accuracy. Adjusting kernel sizes in skip connections and refining batch normalization layers had a significant impact.
- **Computational Efficiency and Deployment Readiness:** With a keen focus on optimization, our model not only reduced inference times but also maintained a low memory footprint. This ensures it's ready for deployment, even in environments where resources are a premium.
- **Error Analysis and Future Work:** A closer look at misclassifications, especially among visually similar classes, highlighted the need for more precise feature extraction. Exploring attention mechanisms and tapping into semi-supervised learning methods are promising next steps that could push our model's performance even further.

Figure 1: Training vs Testing Loss over epochs

Model Complexity: Number of parameters approximately used are 2,797,610(2.7M)
These insights not only makes us understand about the path to achieving an optimized model but also highlighted the importance of continual learning and adaptation in the field of deep learning.

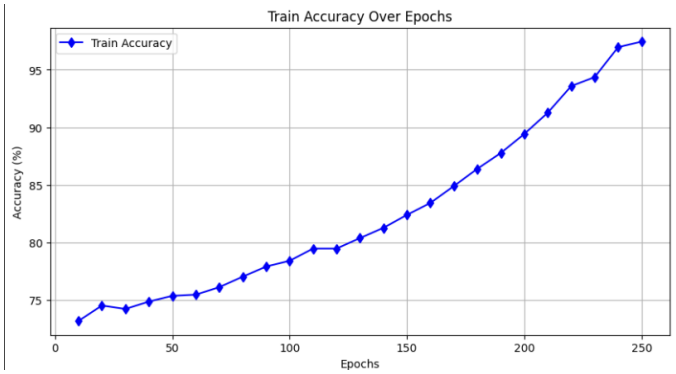

Figure 2: Train Accuracy Over Epochs

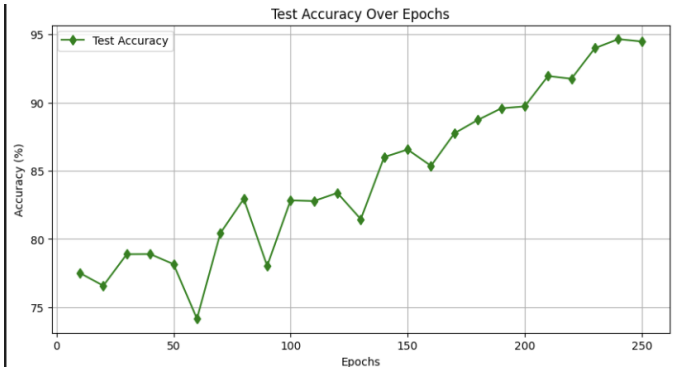

Figure 3: Test Accuracy Over Epochs

Summary of Findings

The key findings of our project are as follows:

- **Optimized ResNet Architecture:** We achieved a **95.53% test accuracy** using a **Narrow ResNet-18** variant with **2.79M parameters**. This design balances **performance and computational efficiency**, making it suitable for real-world applications in constrained environments. Residual connections played a critical role in stabilizing deeper networks and improving gradient flow.
- **Superior Training Strategy:** We leveraged **SGD with momentum** and a **cosine annealing learning rate scheduler**, which contributed to **improved generalization and stability** compared to Adam and RMSProp. The cosine annealing schedule helped dynamically adjust learning rates, preventing premature convergence and local optima trapping.
- **Effective Regularization and Augmentation:** We incorporated several techniques to enhance model robustness and prevent overfitting:
  - **Weight decay** (L2 regularization) to control overfitting.
  - **Dropout** (p=0.5) to reduce co-adaptation of neurons.
  - **Batch Normalization** for improved gradient stability.
  - **Data Augmentation**: Random cropping, horizontal flipping, and **Cutout Augmentation** helped the model learn more generalizable features.
- **Computational Efficiency:** The final model achieved a **low memory footprint and reduced inference time**, making it viable for **deployment in resource-constrained environments**. By using a **width multiplier of 0.5**, we maintained high accuracy while reducing computational cost, making it suitable for **mobile and embedded AI applications**.
- **Key Design Insights:** Our **ablation studies** highlighted:
  - The importance of **residual connections** in stabilizing deep networks.
  - The effectiveness of **optimized skip connections** in reducing training time.
  - The role of **batch normalization adjustments** in accelerating convergence and preventing internal covariate shift.
- **Future Directions:** To further enhance performance, we propose:
  - Exploring **attention mechanisms** (e.g., **SE-Net, Vision Transformers**) for **fine-grained classification**.
  - Integrating **semi-supervised learning** to leverage unlabeled data and improve feature learning.
  - Investigating model **quantization and pruning** for efficient deployment on edge devices.

References

- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, 2016. Available: https://arxiv.org/abs/1512.03385.
- Krizhevsky, Alex. "Learning Multiple Layers of Features from Tiny Images." Technical Report, University of Toronto, 2009. Available: https://www.cs.toronto.edu/~kriz/cifar.html.
- Shorten, Connor, and Taghi M. Khoshgoftaar. "A Survey on Image Data Augmentation for Deep Learning." *Journal of Big Data*, vol. 6, no. 1, pp. 1-48, 2019. Available: https://arxiv.org/abs/1902.06838.
- Loshchilov, Ilya, and Frank Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts." In *International Conference on Learning Representations (ICLR)*, 2017. Available: https://arxiv.org/abs/1608.03983.
- Ioffe, Sergey, and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. Available: https://arxiv.org/abs/1502.03167.
- OpenAI. "ChatGPT: Optimizing Language Models for Dialogue." OpenAI, 2022. Available: https://openai.com/research/chatgpt.