Deep Learning Mid-Term Project

Chhatrapathi Sivaji Lakkimsetty, cl7203@nyu.edu
Shaktidhar Kanamarla, sk10945@nyu.edu
Kurapati Aravind, ksa9148@nyu.edu

https://github.com/ShaktidharK1997/DLProject

Introduction

This mini-project is dedicated to developing a modified ResNet architecture, tailored to achieve optimal performance on the CIFAR-10 dataset under strict parameter constraints. Our goal is to meticulously balance model complexity and efficiency, exploring innovative architectural adjustments and training strategies within the limitations of 5 million parameters.

CIFAR-10 is widely recognized as a benchmark in machine learning for image classification, valued for its diverse and manageable dataset size covering 10 distinct image classes. This benchmark is instrumental for testing and enhancing the robustness of image recognition models, providing a consistent framework for comparative evaluation.

Given the constraints on model parameters, this project addresses the inherent challenges such as managing model capacity and complexity, which are critical for avoiding underfitting while ensuring generalization on unseen data. Optimizing a constrained model demands a strategic blend of architecture selection and hyperparameter tuning, alongside employing advanced regularization and efficient network designs to leverage limited resources fully. This introduction sets the stage for discussing our approach, methodology, and the impact of our innovations on the performance of the tailored ResNet model on the CIFAR-10 dataset.

Architecture

The core philosophy guiding the modifications to the standard ResNet architecture was to optimize the trade-off between model complexity and computational efficiency. The goal was to enhance the model's learning capabilities within a constrained parameter budget, which required innovative approaches to design and layer configuration. The focus was on ensuring depth sufficiency to learn complex patterns without unnecessarily increasing the model's size, which could hinder training efficiency and lead to overfitting.

Detailed Layer Configuration

- **Initial Convolutional Layer:** The model starts with a standard convolutional layer using a 3x3 kernel, which is typical for capturing the initial spatial relationships in the input image. This layer has a stride of 1 and padding of 1 to preserve the spatial dimensions of the input, setting the stage for the deeper layers to perform more complex extractions.
- **Residual Blocks Configuration:** Each residual block within the layers consists of two 3x3 convolutional layers surrounded by batch normalization and followed by ReLU activations. This setup ensures non-linear learning capabilities while maintaining the efficiency of training. The choice of residual blocks helps in combating the vanishing gradient problem, allowing deeper networks to be trained effectively.
- **Expansion of Feature Maps:** The network gradually increases the number of channels in deeper layers (doubling them in each subsequent set of residual blocks), which allows the network to handle more complex features as it goes deeper. This controlled expansion is crucial for managing the parameter count while still enhancing the network's ability to abstract from raw pixels to high-level features.

Specific Modifications

- **Reduced Depth and Adjusted Width:** The architecture consists of three groups of residual layers with depths of 4, 4, and 3 blocks respectively. This configuration provides a deep enough network to capture complex features without excessive depth that could lead to overfitting or unnecessary parameter inflation.
- **Kernel Sizes in Convolutional Layers:** Each convolutional layer within the BasicBlock uses a 3x3 kernel size. This size is a standard choice in CNNs as it effectively captures spatial hierarchies with minimal computational requirements. Adjusting the kernel size allows for a balance between receptive field coverage and parameter efficiency.
- **Shortcut Connections:** The shortcut connections employ 1x1 convolutions when there is a need to adjust dimensions for addition operations with the residual

blocks' output. These are particularly important for ensuring that feature maps align correctly while adding minimal extra parameters.

Rationale Behind Architectural Choices

- **Dropout and Batch Normalization:** Dropout with a rate of 0.1 is incorporated into the BasicBlock. This regularization technique helps prevent overfitting by randomly deactivating a subset of neurons during training, forcing the network to learn more robust features. Batch normalization is employed after each convolutional layer to stabilize learning and accelerate convergence by normalizing the inputs to each activation function.
- **Squeeze-and-Excitation (SE) Blocks:** Integration of SE blocks reflects a modern approach to enhancing channel interdependencies, allowing the model to perform dynamic channel-wise feature recalibration. This feature boosts the representational power of the network by focusing it on the most informative features, without a substantial increase in parameter count.
- **Adaptive Average Pooling:** The final layer before the classifier uses adaptive average pooling to reduce each channel to a single value. This technique simplifies the output feature map management, making it independent of the input size, which is particularly beneficial for maintaining model flexibility and effectiveness across varying input dimensions.
- **Number of Channels:** Starting with 64 channels and expanding in powers of two across the subsequent blocks allows for a gradual increase in feature complexity and abstraction while managing the total parameter budget efficiently.

Implementation Details

Normalization and Data Augmentation:

- **Normalization:** Each image in CIFAR-10 is normalized using mean and standard deviation values typical for this dataset:

$$\text{Mean} = [0.4914, 0.4822, 0.4465]$$
$$\text{Standard Deviation} = [0.2023, 0.1994, 0.2010]$$

  This standardization helps in stabilizing the learning process and ensures that the input features are on a comparable scale.

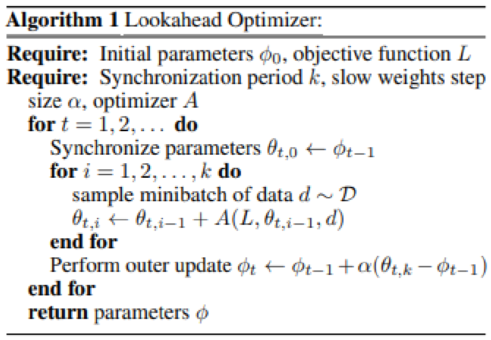
- **Data Augmentation:** To enhance the model's ability to generalize and to prevent overfitting, we employ several data augmentation techniques:
  - **Random Horizontal Flipping:** This transformation mirrors images along the y-axis, effectively doubling the dataset's diversity concerning object orientations.
  - **Random Cropping:** Images are randomly cropped to 32x32 pixels after padding them by 4 pixels on each side. This technique introduces translational invariance, teaching the model to recognize objects regardless of their position in the frame.

These preprocessing and augmentation strategies are particularly suited for CIFAR-10 due to the dataset's relatively small image size and the high variability in object placement and orientation across the dataset.

Training Hyperparameters and Scheduling:

- **Learning Rate and Optimizer:** We initiate training with a learning rate of 0.1, utilizing the SGD optimizer for its robustness and effectiveness in handling noise and its general superiority in reaching deeper, sharper minima on complex datasets like CIFAR-10. The optimizer is enhanced with a momentum of 0.9 and a weight decay of 0.0005 to aid in faster convergence and regularization.

- **Lookahead Optimizer:** The SGD optimizer is wrapped with a Lookahead mechanism, which periodically updates the network weights based on a slow-moving average. This strategy helps in navigating flatter regions of the loss landscape and provides a form of automated learning rate annealing, leading to potentially more stable and improved convergence.



```
Algorithm 1 Lookahead Optimizer:
Require: Initial parameters $\phi_0$, objective function $L$
Require: Synchronization period $k$, slow weights step
  size $\alpha$, optimizer $A$
  for $t = 1, 2, \dots$ do
    Synchronize parameters $\theta_{t,0} \leftarrow \phi_{t-1}$
    for $i = 1, 2, \dots, k$ do
      sample minibatch of data $d \sim \mathcal{D}$
      $\theta_{t,i} \leftarrow \theta_{t,i-1} + A(L, \theta_{t,i-1}, d)$
    end for
    Perform outer update $\phi_t \leftarrow \phi_{t-1} + \alpha(\theta_{t,k} - \phi_{t-1})$
  end for
  return parameters $\phi$
```

Figure 1: Look ahead optimizer

- **CutMix Data Augmentation:** As part of our strategy to enhance model generalization and robustness, we incorporate CutMix augmentation into the training process. CutMix randomly combines patches from two different images and mixes their labels proportionally to the size of the patches. This technique not only enriches the training data variability but also encourages the model to focus on less dominant features of the objects, enhancing its predictive capabilities across varied scenarios.
- **Epochs and Early Stopping:** The model is set to train for up to 200 epochs. However, to prevent overfitting and to save computational resources, we implement an Early Stopping mechanism. This technique monitors the validation loss, and if no improvement is seen for a patience period of 7 epochs, training is halted. This approach ensures that the model training ceases at an optimal point, maintaining high generalization ability.

- **Learning Rate Scheduler:** The Cosine Annealing Learning Rate Scheduler is employed, where the learning rate gradually decreases following a cosine curve from an initial rate of 0.1 to nearly zero over the course of the training epochs. This method helps in fine-tuning the model parameters by taking smaller steps as we approach potential minima, enhancing the ability to find a global minimum.

Experimental Setup

Evaluation Metrics:

- **Loss Function:** The Negative Log Likelihood Loss (NLLLoss) was used as the loss function, which is standard for classification tasks with multiple classes.
- **Accuracy:** Model performance was primarily measured in terms of classification accuracy on the validation set.
- **Early Stopping:** To prevent overfitting and unnecessary training cycles, an early stopping mechanism was implemented. This mechanism monitors validation loss and halts training if no improvement is seen for 80 consecutive epochs.

Cross-Validation:

- The CIFAR-10 dataset was split into a training set and a validation set, with the standard split of 50,000 training images and 10,000 validation images.
- While full k-fold cross-validation was not employed due to resource constraints, the robustness of the model was ensured through a thorough validation process, where the validation set provided a reliable estimate of the model performance on unseen data.

Results

During the model's training process, the losses for both training and testing datasets were recorded to monitor the model's performance and convergence. This is crucial for diagnosing training behavior such as overfitting or underfitting and making informed adjustments to the model architecture or training workflow.

Train Vs Test : Loss

The depicted graph, *Figure 2*, traces the evolution of model losses over 200 epochs, delineating a sharp initial learning phase followed by a more nuanced refinement period. The training loss manifests a steady descent, reflective of the model's growing adeptness on familiar data. Concurrently, the test loss oscillates before stabilizing, indicating the model's evolving proficiency with novel data. Post epoch 50, a diminishing volatility in test loss suggests the onset of convergence. Throughout this process, the training loss predictably undershoots the test loss, embodying the typical dynamic of machine learning where models are primed on training datasets. This visualization encapsulates the dual objectives in machine learning: optimizing for both recall of training data and predictive acumen on unencountered data.

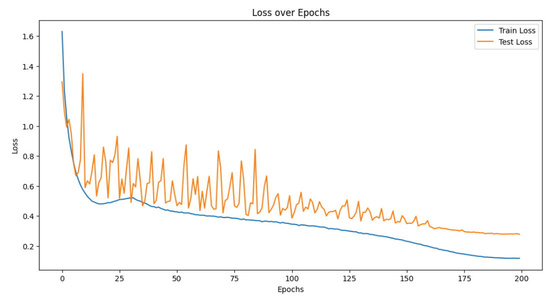


Figure 2: Loss

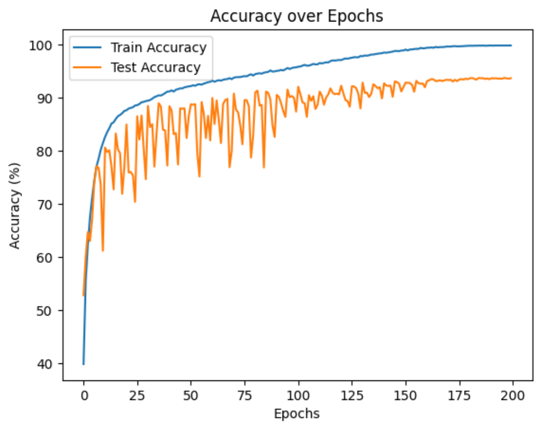


Figure 3: Accuracy

Accuracy

The graph, *Figure 3*, presents a progression of model accuracy over 200 epochs, illustrating a significant upsurge in the initial phase, with training accuracy plateauing near perfection as epochs advance. Test accuracy, following an upward but fluctuating trend, indicates the model's generalization capability is improving, albeit with some inconsistency. Notably, the test accuracy experiences a convergence trend post-epoch 75, aligning closer to the training accuracy, which is indicative of a well-fitting model. The graph underscores the iterative refinement of model performance, balancing adeptness on seen data against its predictive power on unseen data, and ultimately reflecting the model's maturation and stability over time.
Final Metrics on evaluation are:

Average test loss: 0.2790
Accuracy: 93.67%

Visualize Predictions

In the Figure 2, the image provides a visual representation of a model's predictions on the CIFAR-10 dataset, with a series of images correctly classified across various classes. These results showcase the model's ability to accurately discern and categorize diverse images, from animals to vehicles. The consistent accuracy across different object types reflects the model's robust feature extraction and generalization capabilities. This successful outcome underscores the potential of the model for reliable image classification tasks and implies a high level of performance. The precise predictions in this sample set are indicative of an effectively trained model that performs well on this standard dataset.

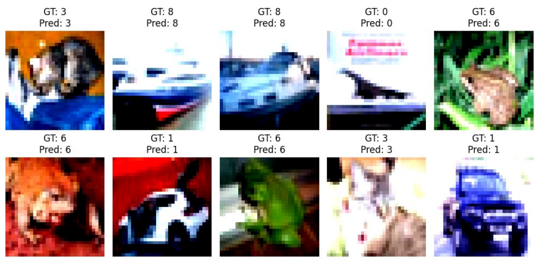


Figure 4: Predicted Vs Actual Labels

Trainable parameters: **4697742**, which is less than the given criteria of **5M** trainable parameters limit.

Improvements

To enhance the performance of the model represented above, implementing the following techniques might lead to better results:

- **Data Augmentation:** Increase the diversity of the training set with advanced data augmentation techniques like rotation, scaling, Cutout, or Mixup. These methods can improve the model's ability to generalize from the training data to unseen data
- **Learning Rate Scheduling:** Implement dynamic learning rate adjustment methods such as cyclic learning rates or learning rate warm-up to fine-tune the training process over epochs.
- **Pre-train a Teacher Model:** Start by training a large and deep neural network (the teacher) on the dataset until it achieves high performance. This model can afford to be computationally expensive since it's only used during training.
- **Distill Knowledge to Student Model:** Once the teacher model is trained, use its predictions to train a smaller, simpler student model. The student learns to mimic the teacher's predictions. This process can involve training the student to match the teacher's class probabilities (soft targets), which contain more information per sample than hard labels.

References

[1] M. R. Zhang, J. Lucas, G. Hinton, and J. Ba, "Lookahead Optimizer: k steps forward, 1 step back," Available: https://arxiv.org/abs/1907.08610

[2] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu, "Squeeze-and-Excitation Networks", Available: https://https://arxiv.org/abs/1709.01507

[3] K. Liu, "pytorch-cifar," GitHub repository, 2023. [Online]. Available: https://github.com/kuangliu/pytorch-cifar