Deep Learning Mini Project – Spring -24

Authors: [Chirag Chopra, Anuj Attri, Arushi Ojha]

Mini-Project GitHub Repository: https://github.com/chiragchoprawork/deepLearningProject

Abstract

This project explores the modification of the Residual Network (ResNet) architecture to enhance CIFAR-10 image classification accuracy under a strict parameter limit of 5 million. Through strategic adjustments to the network's depth, width, and incorporation of dropout and batch normalization techniques, we present a ResNet variant that not only adheres to the parameter constraint but also demonstrates superior performance compared to its baseline counterpart. Our findings underscore the potential of architectural adjustments in optimizing deep learning models for resource-constrained environments.

Introduction

This project focuses on optimizing a custom CNN architecture, inspired by ResNet, for the CIFAR-10 dataset—a collection of 60,000 32x32 color images across 10 classes, a benchmark in computer vision. The project aimed to enhance classification accuracy without exceeding 5 million trainable parameters, balancing complexity with computational efficiency. By innovating within the residual learning framework, our model achieved remarkable efficiency, securing a test accuracy of **93.31**% with only **4.4** million parameters. This project presents a concise analysis of our architecture's adjustments and its performance on unseen test images, highlighting the potential of our approach in advancing image classification with resource constraints.

Related Work

Training deeper neural networks has historically posed challenges due to vanishing gradients and difficulties in optimization. The authors of [2] introduced a residual learning framework designed to simplify the training of networks significantly deeper than previously possible. Instead of learning the target mapping directly, this framework focuses on learning residual functions relative to the input layers. This architectural tweak makes it easier to optimize deep networks. Empirical results further demonstrate that Residual Networks (ResNets) can be optimized more easily and achieve superior accuracy as they grow in depth. For instance, on the ImageNet dataset, ResNets with up to 152 layers achieved a top-5 error rate test accuracy of 96.04% on CIFAR-10. This performance significantly exceeds that of ResNet18, which contains over 11 million trainable parameters, especially when enhanced with strategic training approaches and appropriate hyperparameters for ResNet.

Architecture[1]

The model is an instance of a Residual Network (ResNet), specifically tailored for the CIFAR-10 dataset, which consists of 60,000 32x32 color images in 10 classes. This ResNet model uses **BasicBlock** as its fundamental building unit, designed to effectively handle the challenges of deep learning by mitigating issues related to vanishing gradients.

Initial Convolution and Normalization:

- **Convolutional Layer (conv1)**: This is the first layer in the network, consisting of a convolutional layer with 64 filters of size 3x3, stride 1, and padding 1. This layer processes the input image and prepares it for deeper layers without altering its dimension.
- **Batch Normalization (bn1)**: Follows the initial convolutional layer to stabilize the learning by normalizing the activations.

Residual Blocks (Layers):

- **Layer 1:** Composed of 2 **BasicBlock** units. Each block uses 64 filters, maintaining the dimension of the feature maps since the stride is set to 1.
  Layer 2: Contains 3 BasicBlock units, increasing the number of filters to 128 and reducing the size of the feature maps by setting the stride to 2 in the first block.
  Layer 3: Comprises 2 **BasicBlock** units with 192 filters, continuing to reduce the spatial dimensions with a stride of 2 in the first block.
  Layer 4: Also consists of 2 **BasicBlock** units but with 256 filters, further downsizing the feature maps where the first block has a stride of 2.

- Each **BasicBlock** within these layers includes two convolutional layers with corresponding batch normalization and a dropout layer for regularization. The blocks are equipped with shortcut connections that either directly pass the input if the dimensions are unchanged or use a 1x1 convolution to adjust the

1

dimensions accordingly.

Output Layer:

- **Global Average Pooling:** Reduces each 256-channel feature map to a single value, effectively summarizing the spatial information.
- **Fully Connected Layer (linear):** Maps the pooled features to the 10 classes of the CIFAR-10 dataset.

Dropout Regularization

Applied after the initial batch normalization to prevent overfitting by randomly dropping units during training.

Forward Pass

Data passes through the initial convolution and normalization layers, followed by sequential processing through four stages of residual blocks. Each stage adapts the feature maps spatially and depth-wise to extract and refine features relevant for classification. Global average pooling is applied to the output of the last residual blocks before the final classification layer determines the image classes.

In total, the model has 64 layers as shown in Figure

```
# Displaying summary of the model:
summary(model, input_size=(3,32,32))
```

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	1,728
BatchNorm2d-2	[-1, 64, 32, 32]	128
Dropout-3	[-1, 64, 32, 32]	0
Conv2d-4	[-1, 64, 32, 32]	36,864
BatchNorm2d-5	[-1, 64, 32, 32]	128
Dropout-6	[-1, 64, 32, 32]	0
Conv2d-7	[-1, 64, 32, 32]	36,864
BatchNorm2d-8	[-1, 64, 32, 32]	128
BasicBlock-9	[-1, 64, 32, 32]	0
Conv2d-10	[-1, 64, 32, 32]	36,864
BatchNorm2d-11	[-1, 64, 32, 32]	128
Dropout-12	[-1, 64, 32, 32]	0
Conv2d-13	[-1, 64, 32, 32]	36,864
BatchNorm2d-14	[-1, 64, 32, 32]	128
BasicBlock-15	[-1, 64, 32, 32]	0
Conv2d-16	[-1, 128, 16, 16]	73,728
BatchNorm2d-17	[-1, 128, 16, 16]	256
Dropout-18	[-1, 128, 16, 16]	0
Conv2d-19	[-1, 128, 16, 16]	147,456
BatchNorm2d-20	[-1, 128, 16, 16]	256
Conv2d-21	[-1, 128, 16, 16]	8,192
BatchNorm2d-22	[-1, 128, 16, 16]	256
BasicBlock-23	[-1, 128, 16, 16]	0
Conv2d-24	[-1, 128, 16, 16]	147,456
BatchNorm2d-25	[-1, 128, 16, 16]	256
Dropout-26	[-1, 128, 16, 16]	0
Conv2d-27	[-1, 128, 16, 16]	147,456
BatchNorm2d-28	[-1, 128, 16, 16]	256
BasicBlock-29	[-1, 128, 16, 16]	0
Conv2d-30	[-1, 128, 16, 16]	147,456
BatchNorm2d-31	[-1, 128, 16, 16]	256
Dropout-32	[-1, 128, 16, 16]	0
Conv2d-33	[-1, 128, 16, 16]	147,456
BatchNorm2d-34	[-1, 128, 16, 16]	256
BasicBlock-35	[-1, 128, 16, 16]	0
Conv2d-36	[-1, 192, 8, 8]	221,184
BatchNorm2d-37	[-1, 192, 8, 8]	384
Dropout-38	[-1, 192, 8, 8]	0
Conv2d-39	[-1, 192, 8, 8]	331,776
BatchNorm2d-40	[-1, 192, 8, 8]	384
Conv2d-41	[-1, 192, 8, 8]	24,576
BatchNorm2d-42	[-1, 192, 8, 8]	384
BasicBlock-43	[-1, 192, 8, 8]	0
Conv2d-44	[-1, 192, 8, 8]	331,776
BatchNorm2d-45	[-1, 192, 8, 8]	384
Dropout-46	[-1, 192, 8, 8]	0
Conv2d-47	[-1, 192, 8, 8]	331,776
BatchNorm2d-48	[-1, 192, 8, 8]	384
BasicBlock-49	[-1, 192, 8, 8]	0
Conv2d-50	[-1, 256, 4, 4]	442,368
BatchNorm2d-51	[-1, 256, 4, 4]	512
Dropout-52	[-1, 256, 4, 4]	0
Conv2d-53	[-1, 256, 4, 4]	589,824
BatchNorm2d-54	[-1, 256, 4, 4]	512
Conv2d-55	[-1, 256, 4, 4]	49,152
BatchNorm2d-56	[-1, 256, 4, 4]	512
BasicBlock-57	[-1, 256, 4, 4]	0
Conv2d-58	[-1, 256, 4, 4]	589,824
BatchNorm2d-59	[-1, 256, 4, 4]	512
Dropout-60	[-1, 256, 4, 4]	0
Conv2d-61	[-1, 256, 4, 4]	589,824
BatchNorm2d-62	[-1, 256, 4, 4]	512
BasicBlock-63	[-1, 256, 4, 4]	0
Linear-64	[-1, 10]	2,570

```
Total params: 4,479,946
Trainable params: 4,479,946
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 14.25
Params size (MB): 17.09
Estimated Total Size (MB): 31.35
----------------------------------------------------------------
```

Figure 1: Summary of the Model.

Methodology

Data augmentation is a technique used to enhance the diversity and size of a training dataset by applying various random transformations to the images. In our implementation, several data augmentation strategies are employed to strengthen the model's predictive capabilities and prevent overfitting. These transformations simulate conditions that might occur in real-world scenarios, thereby improving the model's ability to generalize from the provided dataset to new, unseen data. Below is a summary of the methods utilized and their objectives:

- **Random Crop and Padding:** This involves cropping a random part of the input image and padding it if necessary. Here, it randomly crops a region of the image and then pads it back to its original size. This can simulate the effect of zooming in on parts of the image, helping the model to focus on details.
- **Random Horizontal Flip:** It gives us the image that is horizontally mirrored with a 50% chance. This reflects the possibility of objects facing either direction in real-life images, making the model invariant to horizontal orientation.

- **Normalization:** It is applied to normalize the pixel values of the image. Of course, this is a must for the training to be accelerated as we have to keep the input values approximately equal to normal distribution parameters (mean=0 and std=1). This decreases the overweight ing of certain groups and leads to better solution, which improves the learning efficiency.

These data augmentation techniques are used to create a more versatile and generalized dataset, leading to improved performance on unseen data. By simulating a variety of real-world conditions, these transformations help to focus on the invariant features that define the classes of objects in the CIFAR-10 dataset, irrespective of variations in orientation, position, lighting, or color

In our project, specific components have been selected for optimization, loss computation, and learning rate scheduling. Understanding why these choices are opti- mal for our task can help in tuning our model for better performance. Let's discuss the rationale behind these selections:

- **Loss Function - Cross Entropy Loss:** Cross Entropy Loss is commonly chosen for classification tasks as it evaluates the performance of a model that outputs a probability value between 0 and 1. Cross entropy loss grows as the predicted probability moves away from the actual label, which is ideal for tasks such as classifying the CIFAR-10 dataset, where each image is categorized into one of ten classes. It excels in imposing heavier penalties on incorrect classifications, thereby steering the model towards more precise predictions.
- **Optimizer - SGDM:** Stochastic Gradient Descent with Momentum (SGDM) builds upon the foundational principles of SGD, integrating momentum to enhance optimization. Similar to SGD, SGDM updates parameters using batches of data rather than the entire dataset, offering computational advantages for large-scale problems. Momentum in SGDM enables smoother and more consistent updates by incorporating a fraction of the previous update vector. This helps to mitigate oscillations and speed up convergence, especially in the presence of noisy gradients. SGDM's adaptive learning rate, coupled with momentum, facilitates efficient exploration of the parameter space and can lead to faster convergence compared to traditional SGD. It serves as a versatile optimizer in neural network training, providing a balance between simplicity and effectiveness, often complemented with learning rate schedules for further optimization.
- **Learning Rate Scheduler - OneCycleLR**: The OneCycleLR policy modulates the learning rate over the course of training based on a cyclical schedule that first increases and then decreases the learning rate. This approach is designed to enable faster convergence, reduce training time, and improve the overall performance of the model by allowing it to escape local minima more effectively during the initial higher learning rates phase, and then refine its exploration of the parameter space during the decreasing phase. OneCycleLR is particularly effective in training deep learning models, as it helps in navigating the complex landscape of high-dimensional loss functions. In our project, this scheduler increases the learning rate from a lower bound to a peak before annealing it back to a lower bound, thus balancing the benefits of both warm-up and cooldown phases. This cyclical approach not only aids in rapid convergence but also stabilizes training near the end of the cycle when the learning rate is minimized, enhancing the model's ability to converge to a more optimal solution.
- **Summarizing the hyperparameter choices:**

  Learning Rate: Initialized at 0.01 to ensure stable but effective progression towards convergence.

  Momentum: Set at 0.9 to incorporate momentum in the optimization process.

  Maximum Learning Rate: The maximum learning rate is set at 0.1, allowing the model to explore the parameter space more aggressively before refining its approach as the rate decreases.

  The use of SGDM and OneCycleLR exemplifies an adaptive approach to managing the learning rate:

  SGDM(Stochastic Gradient Descent with Momentum): Efficiently adjusts individual parameter updates based on both the current gradient and the momentum from previous updates, which enhances optimization.

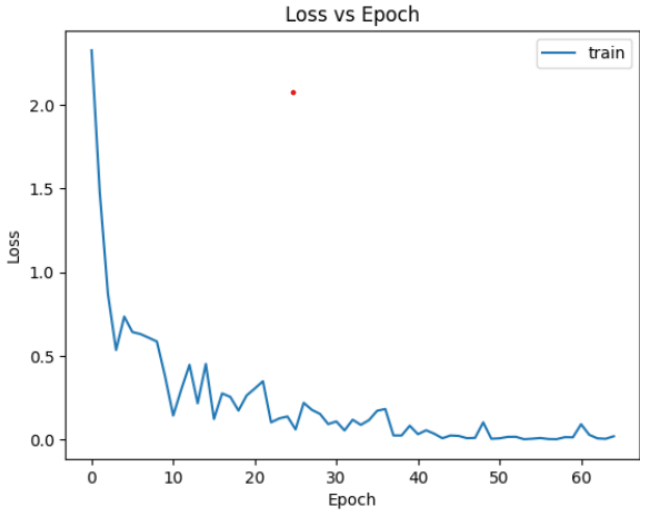
  OneCycleLR: Modulates the overall learning rate cyclically—rising to a maximum and then falling—to combine the benefits of rapid convergence initially and precise fine-tuning later.

  Epochs: Chosen as 65 to provide sufficient iterations for learning while avoiding overfitting and unnecessary computational costs.

  This setup of epochs, adaptive optimizer, and dynamic learning rate scheduler effectively balances training dynamics, promoting both speed and accuracy in model performance.

Results

In Figure 2, we plot the training and test loss curves against each epoch. Our ResNet model converged at a training loss of 0.0084 and a test loss of 0.0093.



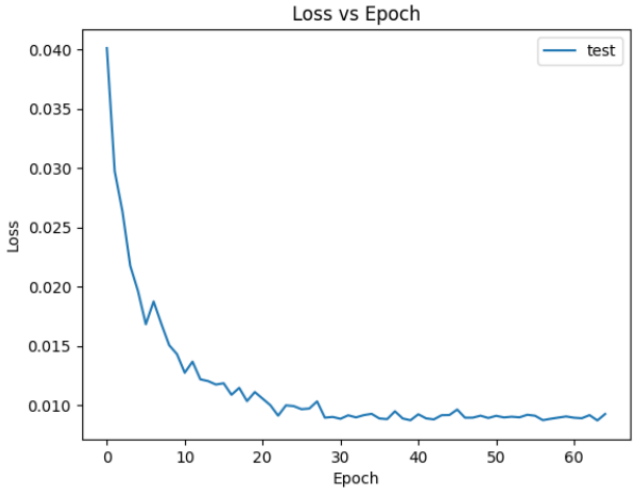


Figure 2: Loss curves

In Figure3, we plot the training and test accuracy curves against each epoch. Our ResNet model converged at a training accuracy of 99.41% and a test accuracy of 93.31%.

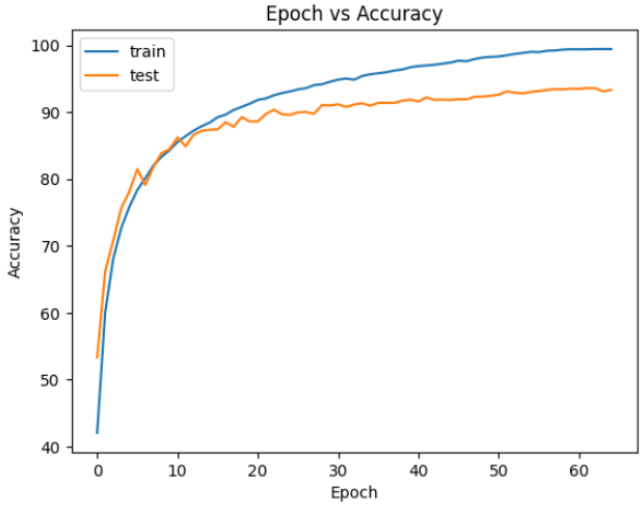


Figure 3: Accuracy curves

Our model trained for 65 epochs as we observed convergence at that point. We don't want our model to overfit to the training data so we didn't train it any longer and stopped at 65 epochs.

- The model employed is a customized ResNet architecture utilizing BasicBlock units designed specifically for handling the CIFAR-10 dataset. This configuration effectively balances depth with computational efficiency, consisting of several layers of residual blocks that enhance the model's capability to learn from complex image data.
- The total number of trainable parameters in the model is 4479946. This streamlined parameter count ensures that the model remains computationally efficient while still capturing sufficient detail to perform high-accuracy classifications.
- The final test accuracy achieved by the model on the CIFAR-10 dataset is 93.31%. This high level of accuracy indicates that the model is exceptionally well-tuned and capable of generalizing from training data to unseen test data effectively.

References

[1] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. CoRR, abs/1512.03385.

[3] Shuvam Das (2023). Implementation of ResNet Architecture for CIFAR-10 and CIFAR-100 Datasets.

[4] Liu, W. et al. (2021). Improvement of CIFAR-10 Image Classification Based on Modified ResNet-34. In: Meng, H., Lei, T., Li, M., Li, K., Xiong, N., Wang, L. (eds) Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery. ICNC- FSKD 2020. Lecture Notes on Data Engineering and Communications Technologies, vol 88. Springer, Cham. https://doi.org/10.1007/978-3-030-70665-4 68

[5] Thakur, A., Chauhan, H., Gupta, N. (2023). Efficient ResNets: Residual Network Design. arXiv preprint, arXiv:2306.12100.