CNN Implementation

Jaehyun Lee
Korea University
jaehyun1030@naver.com

Abstract

In this project, we implemented a convolutional neural network (CNN) using PyTorch to classify images from the CIFAR-10 dataset. The model is based on a modified ResNet-50 architecture, where we filled in the forward function and integrated the training/testing pipeline. We trained the model using the cross-entropy loss and the Adam optimizer. After tuning hyperparameters and completing 500 iterations, our final model achieved a test accuracy of ***81.18%****, which surpasses the target benchmark of 80%. This result demonstrates the effectiveness of residual connections and deep CNNs for image classification tasks.*

1. Introduction

Convolutional Neural Networks (CNNs) have achieved significant success in computer vision tasks such as image classification, object detection, and segmentation. Among various CNN architectures, ResNet has demonstrated remarkable performance by introducing residual learning, allowing for the training of much deeper networks.

In this project, we implement and train a ResNet-50 architecture on the CIFAR-10 dataset, a widely-used benchmark for image classification. The goal of this assignment is to understand the internal structure of ResNet-50 and verify its performance through proper training and evaluation.

2. Implementation

In this project, we implemented a convolutional neural network based on the ResNet-50 architecture to classify images from the CIFAR-10 dataset. The implementation was completed by modifying the provided skeleton file `resnet50_skeleton.py`, focusing on both residual block behavior and overall model architecture.

Residual Block. In `question1`, we filled in the forward function of the `ResidualBlock` class. We constructed a bottleneck block consisting of three convolutional layers (1x1, 3x3, 1x1) with batch normalization and ReLU activation. A shortcut connection was added directly to the output, and downsampling was conditionally applied based on the input.

ResNet Layers. In `question2`, we completed the ResNet-50 architecture by implementing `self.layer2`, `self.layer3`, and `self.layer4`. Each layer contains multiple residual blocks with increasing channel sizes and stride-based downsampling. This hierarchical structure enables deeper feature extraction as the network progresses.

Global Pooling and Classification. We also implemented `self.avgpool` using adaptive average pooling to reduce the spatial dimensions before the final classification layer. The output is then passed through `self.fc`, a fully connected layer that produces logits for 10 CIFAR-10 classes.

Training Setup. The network was trained for one epoch using a batch size of 128. Cross-entropy loss was used along with the Adam optimizer. We used the provided pretrained weights and completed 500 steps of training. The final test accuracy achieved was **81.18%**.

3. Results

We trained the ResNet-50 model on the CIFAR-10 dataset for one epoch using a batch size of 128 and the Adam optimizer. The training loss decreased consistently over the course of 500 steps, demonstrating stable convergence.

The final model achieved a test accuracy of **81.18%**, successfully surpassing the target benchmark of 80% specified in the assignment. This result validates the effectiveness of residual connections and deep architectures for image classification tasks.

```
Epoch [1/1], Step [100/500] Loss: 1.4893
Epoch [1/1], Step [200/500] Loss: 1.0451
Epoch [1/1], Step [300/500] Loss: 0.8894
Epoch [1/1], Step [400/500] Loss: 0.7871
Epoch [1/1], Step [500/500] Loss: 0.7256
Accuracy of the model on the test images: 81.18 %
```

You must get about 80% accuracy for testing.

Figure 1. Training loss and test accuracy over 500 steps. The model achieved 81.18% accuracy on the test set.

4. Discussion

During this project, we successfully implemented key components of the ResNet-50 architecture, including bottleneck residual blocks and layered residual stages. One of the challenges was understanding how downsampling and channel expansion were handled inside residual blocks, particularly when aligning input and output dimensions for the skip connection.

We observed that even with only one epoch of training, the model showed strong performance, thanks to the depth of ResNet-50 and the use of pretrained weights. The skip connections greatly facilitated gradient flow, which likely contributed to stable training despite the model's complexity.

While the final test accuracy of 81.18% is a solid result, we believe that further tuning of the learning rate, training epochs, or data augmentation strategies could yield even higher performance. In future work, we would also explore training from scratch and comparing performance against shallower networks like VGG.

References