Assignment Nº 2: ResNet-50 with PyTorch

Daniel Jader Pellattiero
고려대학교
Seoul, Republic of Korea
danieljaderpellattiero@outlook.com

Abstract

This assignment consists of implementing, training and testing a simplified ResNet-50 architecture using PyTorch. The network needs to be trained and tested on the CIFAR-10 image classification dataset with the aim of achieving at least 80% *classification accuracy on the test set.*

1. Implementation Overview

This section presents the core implementation components of the assignment.

The first part describes the design of the *bottleneck residual block*, which is the fundamental unit of the ResNet framework for *residual learning* purposes. This includes both standard and downsampling variants of the block, with particular emphasis on channel alignment and identity shortcuts. The second part outlines the implementation of a simplified ResNet-50 network, hierarchically organised using *stacked residual blocks*.

1.1. Bottleneck Building (Residual) Block

The `ResidualBlock` class implements a bottleneck-style block with three convolution layers arranged in a $1 \times 1 \rightarrow 3 \times 3 \rightarrow 1 \times 1$ pattern, each followed by a ReLU activation. A shortcut link adds the input directly to the block's output. Two configurations are supported: with and without downsampling. In the downsampling case, the first 1×1 convolution uses a stride of 2, and a parallel projection aligns the shortcut dimensions. Otherwise, all strides are set to 1, and the shortcut defaults to identity, unless the input and output channels are different - in which case a 1×1 convolution ensures compatibility.

```
if self.downsample:
  self.layer = nn.Sequential(
    conv1x1(in_channels, mid_channels, stride=2, 0),
    conv3x3(mid_channels, mid_channels, stride=1, 1),
    conv1x1(mid_channels, out_channels, stride=1, 0))
  self.downsize = conv1x1(
    in_channels, out_channels, stride=2, 0)
else:
  self.layer = nn.Sequential(
    conv1x1(in_channels, mid_channels, stride=1, 0),
    conv3x3(mid_channels, mid_channels, stride=1, 1),
    conv1x1(mid_channels, out_channels, stride=1, 0))
  self.make_equal_channel = conv1x1(in_channels,
      out_channels, 1, 0)
```

1.2. Simplified ResNet-50 Architecture

Key implementation details of the simplified ResNet-50 architecture are presented in this section.

- **Layer 1:**
  - 7×7 convolution, 64 channels, stride 2, padding 3
  - Batch normalization and ReLU activation
  - 3×3 max pooling, stride 2, padding 1
  - Output size reduced to 8×8
- **Layer 2:**
  - 3 residual blocks, $64 \rightarrow 256$ channels
  - Downsampling in the last block
  - Output: $256 \times 4 \times 4$
- **Layer 3:**
  - 4 residual blocks, $256 \rightarrow 512$ channels
  - Downsampling in the last block
  - Output: $512 \times 2 \times 2$
- **Layer 4:**
  - 6 residual blocks, $512 \rightarrow 1024$ channels
  - No downsampling
  - Output: $1024 \times 2 \times 2$
- **Final layers:**
  - Average pooling with 2×2 kernel
  - Fully connected layer for 10-class classification

2. Experimental results

The model was initialized from a pre-trained checkpoint (`resnet50_epoch285.ckpt`) and trained for an additional epoch using the Adam optimizer, with a learning rate of 1×10^{-3} and cross-entropy loss. The learning rate was configured to decay by a factor of 3 every 20 epochs. Training and evaluation were performed on the CIFAR-10 dataset, and final test accuracy was computed in inference mode using the test set, reaching 83.41%.