Assignment Two

February 19, 2024

1 Deep Learning Course

1.1 Assignment 2

1.1.1 Assignment Goals

- Design and implementation of CNNs.
- CNN visualization.
- Implementation of ResNet.

In this assignment, you will be asked to learn CNN models for an image dataset. Different experiments will help you achieve a better understanding of CNNs.

1.1.2 Dataset

The dataset consists of around 9K images (some grayscale and some RGB) belonging to 101 classes. The shape of each image is (64,64,3). Every image is labeled with one of the classes. The image file is contained in the folder named after the class name.

1.1.3 Requirements

1. **(40 points) Implement and improve a CNN model.**

   (a) We are aiming to learn a CNN on the given dataset. Download the dataset, and use PyTorch to implement LeNet5 to classify instances. Use a one-hot encoding for labels. The dataset is already split into training (90 percent) and validation (10 percent). Report the model loss (cross-entropy) and accuracy on both training and validation sets. (20 points)

   The LeNet5 configuration is:

   - Convolutional layer (kernel size 5 x 5, 32 filters, stride 1 x 1 and followed by ReLU)
   - Max Pooling layer with size 4 and stride 4 x 4
   - Convolutional layer (kernel size 5 x 5, 64 filters, stride 1 x 1 and followed by ReLU)
   - Max Pooling layer with size 4 and stride 4 x 4
   - Fully Connected ReLU layer that has 1021 neurons
   - Fully Connected ReLU layer with 84 neurons
   - Fully Connected Softmax layer that has input 84 and output which is equal to the number of classes (one node for each of the classes).

   (b) Try to improve model accuracy on the validation dataset by tuning the model hyperparameters. You can use any improvement methods you prefer. You are expected to reach at least 65 percent accuracy on the validation set. (20 points)

Here are some improvement methods you can use, of course, you can use others which are not mentioned here:

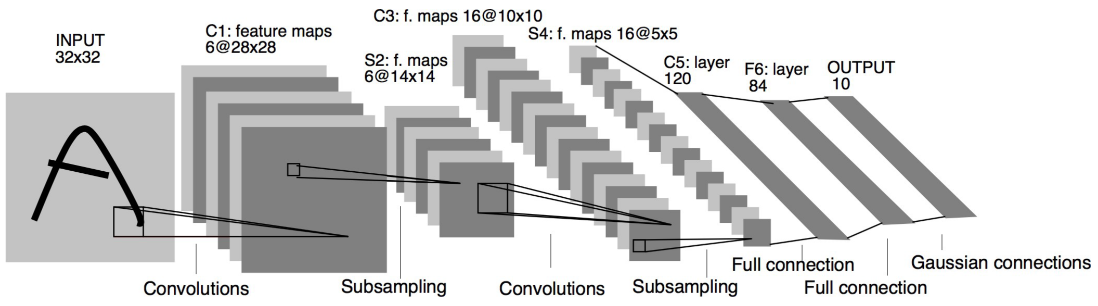
- Dropout
- L1, L2 regularization
- Try improved initialization (e.g., Xavier initializer)
- Batch Normalization

The grading of part (b) is based on the correctness of your implementation (5 points) and the performance of your improvement on the validation set. The validation accuracy and corresponding score are:

- 65% (5 points)
- 67% (8 points)
- 69% (12 points)
- 71% (15 points)

Structure of LENET-5

The following LENET-5 structure is for 10-class dataset. Therefore, the layer size is not exactly the same as ours.



2. **(20 points) Visualize layer activation**

   There are several approaches to understand and visualize convolutional Networks, including visualizing the activations and layers weights. The most straightforward visualization technique is to show the activations of the network during the forward pass. The second most common strategy is to visualize the weights. For more information, we recommend the course notes on "Visualizing what ConvNets learn". More advanced techniques can be found in "Visualizing and Understanding Convolutional Networks" paper by Matthew D.Zeiler and Rob Fergus.

   Please visualize the layer activation of **the first conv layer** and **the second conv layer** of your above CNN model (after completing Q1), on the following 2 images:

   - accordion/image_0001
   - camera/image_0001

   Visualizing a CNN layer activation means visualizing the result of the activation layer as an image. Specifically, the activation of the first conv layer is the output of the first (conv + ReLU) layer during forward propagation. Since we have 32 filters in the first conv layer, you should draw 32 activation images for the first conv layer. Please display multiple images side by side in a row to make your output more readable (Hint: matplotlib.pyplot.subplot).

3. **(40 points) ResNet Implementation**

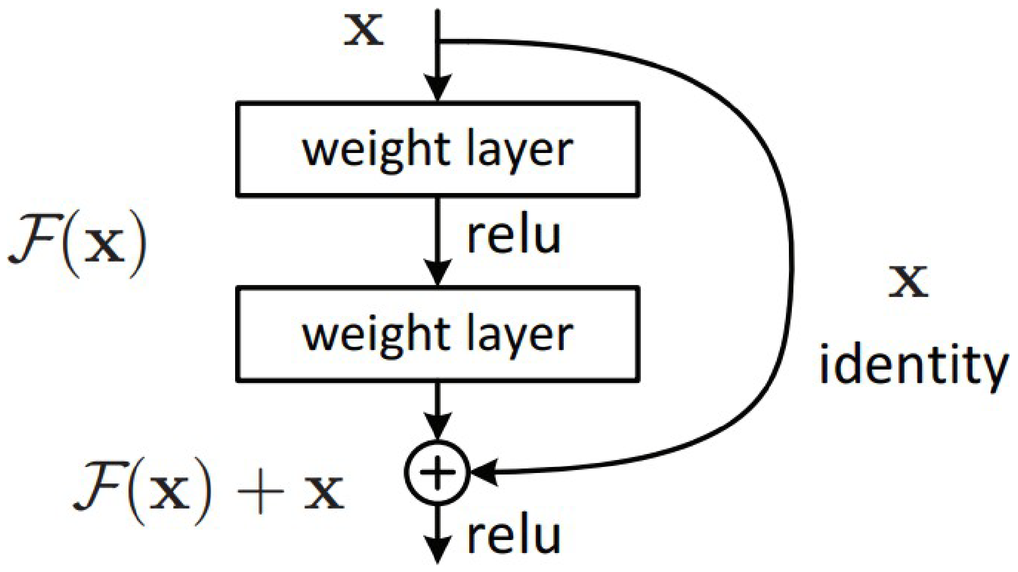
   Use PyTorch to implement ResNet 18 to classify the given dataset. Same as above, please use a one-hot encoding for labels. The dataset is already split into training (90 percent) and validation (10 percent). Report the model loss (cross-entropy) and accuracy on both training and validation sets. See the paper Deep Residual Learning for Image Recognition for a detailed introduction to ResNet.

   The grading of this part is mainly based on the implementation and performance on validation set. If you need more resources to complete the training, consider using Google Colab.

   The ResNet 18 configuration is:

   - conv_1 (kernel size 7 x 7, 64 filters, stride 2 x 2)
   - conv_2 (max pooling layer with size 3 x 3, followed by 2 blocks.Each block contains two conv layers. Each conv layer has kernel size 3 x 3, 64 filters, stride 2 x 2)
   - conv_3 (2 blocks, each contains 2 conv layers with kernel size 3*3, 128 filters)
   - conv_4 (2 blocks, each contains 2 conv layers with kernel size 3*3, 256 filters)
   - conv_5 (2 blocks, each contains 2 conv layers with kernel size 3*3, 512 filters)

   A block has the structure:


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

1.1.4 Submission Notes

Please use Jupyter Notebook. The notebook should include the final code, results, and your answers. You should submit your Notebook in (.pdf or .html) and .ipynb format. (penalty 10 points)

1.2 Your Implementation

1.2.1 Implement and improve a CNN model

[1]:
```
# implement your Lenet5 here
```

1.2.2 Visualize layer activation

[2]:
```
# implement your visualization here
```

1.2.3 ResNet Implementation

[3]:
```
# implement a ResNet model here
```