CS461 Homework 4: LeNet5

Due: December 3rd 11:59 pm

This homework is a group project with up to three members. Please write (1) section #, (2) name, and (3) RUID of all group members.

- member1:
- member2:
- member3:

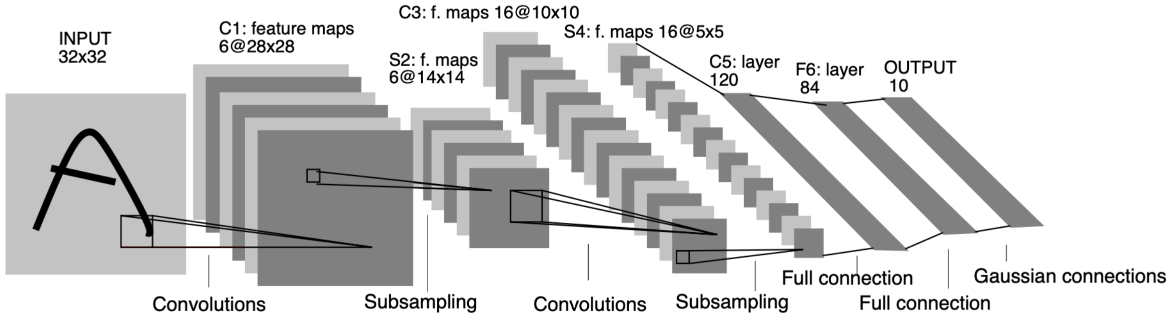

Figure 1: Architecture of LeNet5

In this homework, you will implement LetNet5. LeNet (1989 - 1998) is a series of neural nets developed by LeCun *et al.* to solve the problem of recognizing handwritten digit images. LeNet is historically important in ML for several reasons. First, it formalized a deep-CNN architecture combining the three components: (1) convolutional layers, (2) pooling layers, and (3) fully connected layers. Particularly, the pooling layer was introduced in LeNet for the first time to enhance invariance to shifts and distortions. Additionally, LeNet is the first model to demonstrate end-to-end learning using back-propagation. It trains all parameters simultaneously, enabling the learning of hierarchical features directly from images without the engineered features by hand.

0. Submission

- all codes and LaTex report
- a Box link to the two models: one original (problem1) and one of your design (problem2): The name of the models will be "LeNet1.pth" and "LeNet2.pth" respectively.
- command instructions for inference on your models. For grading, the models will be tested based on the two test sets: (1) the original MNIST test set and (2) the transformed MNIST set TAs will create, including scaling, translation, and rotation.

1. [80 points, LeNet5 Implementation] Please implement the original LetNet5 exactly as described in the paper "Gradient-Based Learning Applied to Document Recognition". Please carefully read the sections I, II, III and Appendix A, B, C. For your implementation, you can use PyTorch or TensorFlow. (PyTorch is recommended if you don't have any specific preference.)

1.1 [Architecture] Implement LeNet architecture as described in Section II. For the RBF parameters, use the data DIGIT. There are several ways to generate the 7 x 12 bitmap image for each digit from the data. Please explain your approach and reasoning behind it.

1.2 [Training]

- Data: Run data.py to collect MNIST data samples. The MNIST database consists of 28×28 images. You will need to resize them to 32×32 to match the regular database in the original paper.
- Optimization: Instead of the second order gradient method (Stochastic Diagonal Levenberg-Marquardt) in the paper, use "steepest gradient method": $w_{k+1} = w_k - c \cdot \nabla J(w)$. A possible step size $c \approx 0.001$.
- use batch size=1
- input image is grayscale each pixel value is [0 - 255]
- Loss function: use equation (9) in the paper. $j = 0.1$ and i denotes incorrect classes.
- At every pass (every epoch), track the error rates. You will need the rates for plotting in the performance evaluation.

1.3 [Performance Evaluation]

- Create a plot of training and test error rate, similar to Fig 5.
- Compute a confusion matrix (10×10).
- For each digit, identify the most confusing example. For instance, which image of the digit 'one' was misclassified with the highest confidence, and to which digit was it misclassified?
- Report your train and test error rates at epoch 20.
- Write your test code (test1.py) for grading.

2. [40 points, Modifying LeNet5 to Handle Unseen Data] You will modify the original LeNet to handle unseen MNIST data. Possible modification examples will be, but not limited to, (0) adding image pre-processing block, (1) data augmentation, (2) adopting the modern blocks such as max pooling, and dropout, ReLU, softmax, or Spatial Transformation Network, or (3) different training schemes.

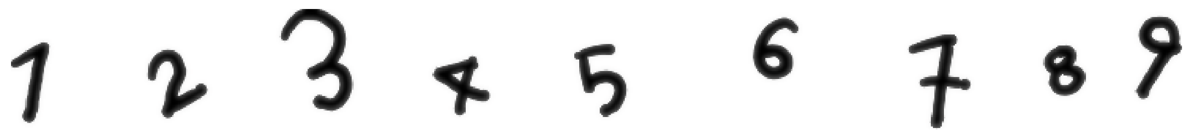

Figure 2: Unseen Data Examples

2.1 Set up multiple plans to build a new neural net robust to geometric transformation or unseen handwritten styles, and train your new models accordingly. Explain your plans based on the components you want to modify and the rationale behind the modifications. You may need to create a transformed validation dataset to estimate the error rates of your new models on unseen data (reference). Your experimental plans and explanations will be graded along with the performance of your final submission model on the test dataset that TAs will newly create. Write your test code (test2.py) for grading.