Assignment 1

MNIST (100 points)

In this problem, we will implement a model to classify digits using the MNIST dataset. The dataset is available at https://git-disl.github.io/GTDLBench/datasets/mnist_datasets/ or https://pytorch.org/vision/stable/generated/torchvision.datasets.MNIST.html. Please use both training and test sets. You can train your model for 20 epochs. Use a batch size of 64 to train the model. Please submit a report and code for this assignment. You can use Pytorch or Tensorflow in this assignment.

K Nearest Neighbors (KNN) (25 points)

We will implement a K-nearest neighbor model that finds the most similar image given a test image. We can use the *scikit-learn* Python library to find the KNN. We will use the sum of absolute difference (SAD) on all the pixels to measure the similarity between two images. What will be the accuracy if we find the nearest neighbor? What will be the accuracy if we find K neighbors? When we find K, we choose the label that appears most among the KNN images (if there is a tie, we pick the one with the smallest aggregated SAD). Plot a curve of accuracy versus K from 1 to 10.

Multilayer Perceptron (MLP) (25 points)

Treat each image as a 784d (28×28) vector. Train a multilayer perceptron model to classify images. Use two hidden layers (not including the input layer and the output layers). Each hidden layer is a fully connected layer followed by ReLU (Rectified Linear Unit $f(x) = max(0, x)$). The final layer should output 10 numbers indicating the scores of the 10 classes. Try cross-entropy loss for the classification[1]. We can experiment with different neurons in the hidden layers. What is the accuracy of such a model? Plot a curve of accuracy versus the number of neurons: 4, 8, 16, 32, 64, 128, and 256. We assume both hidden layers have the same neurons.

[1] In Tensorflow, we can use https://www.tensorflow.org/api_docs/python/tf/nn/softmax_cross_entropy_with_logits. In PyTorch, we can use https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

Operation	3×3 conv	3×3 conv	3×3 conv	3×3 conv	3×3 conv	Avg pool
Activation	LReLU	LReLU	LReLU	LReLU	LRelU	N/A
Dilation	1	2	4	8	1	N/A
Receptive field	3×3	7×7	15×15	31×31	33×33	N/A
Feature channels	32	32	32	32	10	10

Table 1: The CAN model. The average pooling is a global average pooling that outputs a 10-dimensional vector[4]. LReLU represents Leaky ReLU.

Convolutional Neural Networks (CNN) (25 points)

Implement the LeNet-5 convolutional neural network. You can follow this tutorial for the details of the LeNet:

`https://pub.towardsai.net/the-architecture-implementation-of-lenet-5-eef03a68d1f7`

Please use input image size 28×28 in your model, although the input image size in the original LeNet-5 model is 32×32. What is the accuracy of the LeNet-5 model?

Context Aggregation Networks (CAN) (25 points)

We can also implement a simple CNN model with dilated convolutions. We can have a model with a large receptive field with the same spatial resolution in the hidden layers. See the network architecture in the Table 1. For your information, the dilated residual network[2] is a similar model for image classification.

What is the accuracy of such a CAN model? What will be the accuracy if we use a different number of feature channels instead of 32?

[2] Fisher Yu, Vladlen Koltun, and Thomas A. Funkhouser, "Dilated Residual Networks," CVPR 2017