Effect of NN Architecture Configuration and Backpropagation

1. In this assignment the objective is to try to understand the role of network configuration when the number of parameters are kept the same for different configurations.

   You have to design three different fully connected neural net where each of them have the same number of parameters (call it N) and the same number of layers (call it L). L includes all hidden layers and the output layer.

   In the first net (call it UniformNet) each consecutive connecting layers (except the input and the first hidden layer) will have the same number of parameters. In case you can’t keep it the same, adjust them in the last hidden layer only.

   In the second net (call it PyramidNet), the first two connecting hidden layers will have the largest number of parameters and the it will decrease gradually with the layer depth. Take the rate of decrease ½. That is if the first two connecting hidden layers has m parameters, then the next will have m/2 parameters and so on.

   In the third net (call it InvPyramidNet, the first two connecting hidden layers will have the smallest number of parameters and the it will increase gradually with the layer depth. Take the rate of increase as 2. That is, if the first two connecting hidden layers has m parameters, then the next will have 2m parameters and so on.

   Train and test these three nets on MNIST data and measure the accuracy on the test set. Set the number of parameters N to 100000 and number of layers L to 10. For simplicity, set the number of neurons in the first hidden layer to 50 for all the three cases. Use batch normalization and dropout (with 0.3). Use adam optimizer with learning rate 0.0001. Use Xavier initialization. Use RELU activation and in the last layer use softmax. Error function is cross entropy. You may not find networks with exactly 100000 parameters. In that case slight +/- in terms of number of parameters is fine.

   You must submit a output file (.txt) with accuracy values for all three nets.

2. Implement backpropagation from scratch and use it to train a fully connected neural net with mini-batch vanilla gradient descent. Test your implementation on MNIST data. To check correctness of your implementation, also implement the same net using standard libraries and train with the exactly same settings.

   Your output file (.txt) must contain the accuracy of the above two implementations on MNIST data.

The deadline is: 20.02.22 (11.55 IST)

Submit in moodle.

Put all your files into a folder, zip it and upload.