Abstract

The objective of this project is to increase the understanding of machine learning and convolution neural network and how design CNN in efficient way to get more than 97% correct answer after train it with MNIST dataset of image contain digits and label for this digits then test it by input to this CNN to detect a digit in an image. Therefore, fully convolutional neural network used to estimate the digit and non-max suppression to get maximum approach to correct digit.

1. Method

The MNIST handwritten digit database was chosen to train our CNN. MNIST contains 60,000 labeled training examples and 10,000 examples for testing. The training example was divided to 75% for training and 25% for validation while the testing kept untouched to make evaluation on performance of our CNN. Reshape the images data into 4D matrix 28x28x1x60000 (1 is because of gray scale) and convert labels data into categorical in order to be used in CNN. In this way, all image has the same size, so we avoided multi-scale processing of the image.

This image will be the input of CNN. The CNN consist of layers, each layer has number of nodes connected to nodes in next layer with weight w_k while next node has initial value call bias w_0. So the input value multiplied with a weight w_k and add to bias then feeded to active function (sigmoid)

$$p = \frac{e^y}{1 - e^y}$$

This probability used to calculate log-likelihood loss in every nod at every layer. But in CNN contain a huge amount of weights and to training the will take a lot of time and calculation. The full convolutional network will use which is simply CNN but not all layers are connected to each other that will allow us to use it in full images in different size. First layers in full-CNN is image input layer 28x28x1. Next layer is convolution Layer which apply 20 5x5 sliding convolution filters and calculate the multiplication between the input and weight the adding a bias term of the weights and the input, and then adding a bias term. Then batchNormalizationLayer use to normalizes the output of convolution layer. After that to increase the training speed get better sensitivity use ReLU layer and also act like activation function or sigmoid function. And the next layer is maxPooling2dLayer(2,'Stride', 2) which mean down sample the regions by 2x2 and keep the maximum value, the output will still has same number of filter but less dimension's. next three layers are repeating the first three layer but with a smaller number of filters and less dimensions. Last but not least fullyConnectedLayer(10) which act like linear combination of all output by calculating the multiplication between the output from prevision layers and weight matrix and then add a bias. softmaxLayer use to get probability output. At the end pixelClassificationLayer used for semantic segmentation and isolate the background image from digit.

After defining the layers, the training option and condition should be defined like which activation function should be chosen, iteration rate and learning rate, epoch, drop point for iteration and which validation data can be used.

Next step will be to train the network. Therefore, trainNetwork Matlab function used with layers and training option that defined before. Then testing the network by using test image from MNIST dataset and calculate the accuracy by dividing the summation of number of tested accurately data over number of test dataset.

2. Experimental Evaluation

The parameter used for training the CNN

- the input image layer [28x28x1]
- first convolution2dLayer (5,20)
  size of slide window 5x5x20 and learn parameter (5x5)x20+20= 520
  Where 20 is number of filter
- first max pool(2,2) that mean down-samble by halve and filter number stay same
- second convolution2dLayer(3, 10) which mean smaller size of filter and less number of filter 3x3x10 so learning parameter will be (3x3x8)x10+10 = 730
- second max pool(2,2)
- fullyConnectedLayer(10) and learn parameter 4x4x10x10= 1610
- total learning parameter = 520+730+1610 = 2860

After training and testing the accuracy of network by using training and testing dataset from MNIST.mat. The image that provided in canvas was implemented on trained network by using calssify() function to get the right digit in the image. The classify function use two parameter, one is net which is the training CNN result and test_image function.

The test_image.m function is prosses to transfer the image to gray scal then scal it to fit MNIST dataset format which we train our network on it after that separet the digit from background and drow box around that digit and then resize it againe to fit MNIST dataset.

At the end, finding the failed in tested image by compared to labeled test set then finding where the prediction result equal to zero.