Ve593

Jia Shi 716370990049(Second Year Undergraduate)

Liao Xinhao 516370910037(Third Year Undergraduate)

Fan Zheliang 516021910518(Third Year Undergraduate)

Prof. Weng

In the project 4, our team numpy library to construct a Convolutional Neural Networks (aka CNN) to realize the classification of hand-written picture in the Mnist dataset and the object classification of Cifar10 dataset.

For the most implementation method for CNN online, people use frameworks like Tensorflow, Caffe, Keras or PyTorch, which is easy to realize with the given structure. For a complex model like CNN, it's a challenge for our team to implement it from the ground with Numpy. After all, we get to know all the detailed implement and the specific working process for CNN, which strengthen our understanding to this model a lot.

We design our CNN model, construct the base, add some advanced feature and then apply to two dataset.

For our CNN, we have the structure of conv1 - activation1 - pool1 -conv2 - activation2 - pool2 - fullconnection -softmax, which is a basic but efficient structure. Besides that, we also try some famous structure like AlexNet, GoogLeNet, LeCunLet on our Model, some shallow but efficient model. However, this part is only for self learning, we didn't include that in the final submission.

Beside the basic CNN network, we also implement some advance features. First, softmax, we implement this at the end of our model, which is a useful classifier that connect to the output directly. We also have a little research on this part. We change softmax with SVM(which use a lot in the R-CNN model), however, there's not much difference in the performance(both around 96%). Thus, we choose the simple to implement softmax in the final version.

Second, we implement several popular activation function in the CNN, ReLu, LeakReLU and Elu. When it run on the dataset, there's not much

difference between their performance. Our team assume that it's because our model is very shallow and thus the discrepancy is not apparent.

Third, instead of using the regular calculation, we implement the im2col algorithm. This make the regular Convolution calculation into a direct calculation of matrix, which is more efficient when encounter large dataset.

On the Mnist dataset, we implement a early stopping when the accuracy rate reach 95%. Compared to the ANN model we implement in Project 3, the CNN model is faster to reach a high percent of accuracy, but sometime it's less stable. For example, the accuracy will change like—40-50-60-80-40-89…which may encounter sudden drop. In general, we have a average training accuracy of 96.6% and a validation accuracy of 93%. It's very fast to deal with dataset like Mnist.

On the Cifar dataset, different from Mnist, it have three level for each data, which is RGB. This slow down the training process a lot. Thus, we also implement a structure that can support multi-layer of data input. Instead of 28*28*1, we now have 32*32*3. It take much longer time than the Mnist dataset and the performance is not so good. In general, after spending more than 5 or 6 hours of training, we only have a average training of 55.4%(which is pain) and validation average of 47.3%. In our research, we found that it did take a long time for training Cifar dataset, and the performance is around 70-75 percent. In the future, we are considering changing gradient decending method to adam, Nesterov or Momentum instead of the regular SGD to improve accuracy rate.