Machine Learning II

Individual Report

Tianyi Wang

The Street View House Numbers (SVHN) Dataset Project

Introduction

As we explained in the data description section, the dataset consists of three parts: training dataset, test dataset, and the extra training dataset. The main part of our project, we choose to use 73,257 images in training dataset to train our model. In addition, we also applied the full extra training dataset, which has 531,131 images, on CNN model 4 as references to compare how the accuracy will be different than our main models. In our project, there are lots of techniques for refinement and many hyper-parameters need to be tuned, and then observe the loss and accuracy on the training dataset. After checking the results, we can decide which approach may improve the accuracy on the test dataset. For the CNN models, we mainly adjust in three different ways: add layers, increase or decrease the number of hidden units, change the probability of dropout and learning rate. We also apply the CNN model 2 on Caffe.

Description of your individual work

There is how we split our work to each person:

Tianyi	Yina	Jue
MLP1 CNN1	CNN2 CNN3	CNN4 CNN5

There have some feature representations for SVHN dataset. Previous approaches to class labels or digits from digital images are using hand-crafted features such as Histograms-of-Oriented-Gradients (HOG) and an off-the-shelf cocktail of the binary image (stacked sparse auto-encoders and the K-means-based system of) (Netzer, 4). These two are popular options when one tries to solve digit classification problem by using standard machine learning procedures. However, the problem is in order to use either one of them, one must binarize the input images grayscale. Recent scholar introduces a new approach that using ConvNets. The ConvNet architecture has numbers of convolution module, and it followed by pooling module and a normalization module (Sermanet, 1)

In our project, our team plans to implement the image classification using the aforementioned algorithms. We are going to compare the MLP model with different layers convolutional neural network models, and then we are going to apply Caffe on the convolutional neural network model 2. Regarding the accuracy and efficiency based on the performance of the models to be evaluated, and we are optimizing different models in each scenario by tuning the algorithms.

Describe the work

- Multilayer Perceptron Networks:

MLP network has two linear layers and one ReLU layer. The first linear layer has input size as 3*32*32 and output size as hidden_size (=500). The second linear layer has input size as hidden_size and output size as the number of classes (=10).

- Convolutional Neural Network (CNN):

Model 1: the base convolution network has two feature stages and one fully connected hidden layer. Each convolution stage contains a convolution module, followed by the max-pooling module. The convolution kernel has the size of 5 * 5. The first convolution layer produces 16 feature convolution filters, while the second convolution layer outputs 32 feature filters. Both of convolution layers have zero-padding and zero-stride on the input. Both of convolution layer apply the same max pooling window with the size as 2 * 2 with stride 2 * 2. The fully connected layer has input-feature = 800 (32 * 5 * 5) and output-feature = 10.

Results

Based on our results, the convolutional neural network has better performance. We used a two layers MLP, which takes the image pixels vector as input, then the hidden layer with 200 nodes by using ReLU activation function, and finally the output layer with 10 nodes, which are the number of classes in our data. The loss of the MLP model maintains between 0.3 to 0.9. The highest single class accuracy can reach 90% (class 2), and the lowest single accuracy is around 67% (class 3). This simple MLP model can reach the 80% overall accuracy with 10 epochs

training process, and the result doesn't' t change a lot when we are adding more training iterations, try different batch sizes and number of hidden sizes.

The most basic CNN model we built contains two convolutional layers with max-pooling layers, which can get the 0.85 accuracy and 0.84 f1-score by 10 epochs training process. We experiment more epochs, but the result does not improve obviously. For the second model, we add the ReLU layers followed each convolutional layer and max-pooling layers. In this one, the f1-score improved to 0.87. Besides, the loss is really similar to the MLP model. The third model we add the dropout layer behind each ReLU layer with 20% dropout ratio. By using the 0.2 dropout rate, the result is close to the previous model, but when we tried 0.5 dropout rate, the performance becomes worse in f1. In model 4, we add three more fully connected layers and the two of them with ReLU functions. This model gets the best performance within those six different models, which gets 91% accuracy and 0.9 f1-score. The training loss bounces and maintains between 0.2 to 0.6, which is better than the results of MLP and previous CNN models. In the last model, we changed the ReLU function to tanh function. The accuracy decreased to 88%, since ReLU function can greatly accelerate the convergence of stochastic gradient descent compared to tanh function.

Furthermore, we also used the extra dataset as the training data, which contains 531,131 images and more distinct averagely. We tried the MLP model, the first CNN model, and the fourth CNN model by using the extra data as the training set. The fourth CNN model can reach 0.94 f1-score and the loss maintain between 0 to 0.5 which is so much better than the other results, since there are more training data which lowered the occasionality, and also the images are more distinct with the clear and simple background.

Otherwise, we transfer the second CNN model in Caffe framework, which gets the same results, but saves tons of training time (70.45s training time in Caffe). We also plot the confusion matrix (both normalized and non-normalized versions) of our results. Each row of the confusion matrix represents the instances in a predicted class while each column represents the instances in the true class. The confusion matrix can easily show if the system mislabeled one as another. From the confusion matrix (and the plot). Since our results are reasonable (79%~94%), so the plot looks like colored-diagonal. Also, we can easily calculate the precision and recall by this matrix. In addition, we plot the ROC curve for each class of our model. The closer the curve follows the left border and the top border of the ROC space, the more accurate the model. The closer the curve comes to the diagonal line of the ROC space, the less accurate the model. The

area under the curve is a measure of test accuracy. Compared the ROC plots of our MLP model and the fourth CNN model, the AUC of each class in MLP model is higher than 0.80, and the AUC of each class in CNN is higher than 0.9. The AUC of each class of the fourth CNN model by using the extra training data is closer to 1 (higher than 0.95). Thus, the fourth CNN model is more accurate, especially with the very large extra training data.

Summary and Conclusions

With training dataset, the state-of-the-art performance on the test dataset with the lowest accuracy of 80% for MLP and the highest accuracy of 91% for CNN model 4. With the same CNN model 4, the performance on the test dataset with training dataset (~ 70k) has the accuracy of 91% and the performance on the test dataset with the extra training dataset (~ 531k) has the accuracy of 94%. The results above show that the CNN has higher accuracy and lower loss than the MLP model. With implementing the same CNN model, the extra training dataset has better outcomes than the training dataset, because the extra training dataset has simpler and less blurry images, which means the digits in the image will be easier to be detected. Because the training dataset contains some of the images that are even harder to recognize at the first glance. It also comes from a significantly harder real-world problem of recognizing digits and numbers in natural scene images.

The other difficult part of this project is to decide the proper hyper-parameters such as the learning rate. In the scholar research, a case with the application of advanced optimization techniques and applying 7 layers that achieved over 93% accuracy. A comprehensive review of these examples would be meaningful future work to consider. In addition, when comparing the processing of the MLP model and multiple CNN models, we have learned that the running time of the CNN training model will be shorter when compared to the MLP model. The CNN model can reach the same performance (80% accuracy) by using only 14s (One epoch training of our CNN model). Even though the multilayer network can work on lots of input sources, convolution network works better on images. In our project, since training and testing datasets are made of images, the convolution network will be better. Even more, the convolution network has weighed sharing, so the processing time will be shorter than that of the multilayer network.

A couple of interesting next steps to be pursued further research. For example, from the research "*Generalizing Pooling Functions in Convolutional Neural Networks*", they find that “mixed” (with one mix per pooling layer) is outperformed by “gated” with one gate per pooling

layer. Changing the different Pooling layers may change the results for our relatively large SVHN dataset. We are also thinking if we add more layer for our network, maybe we can improve our results too.

Percentage of the code we used

We search on the website in order to look at how to change the original dataset to ImageNet LMDB version, and this is the only file that we use from the website since we don't know how to transfer the data. As a result, the percentage will be 8.16%.

References

Lee, Chen-Yu. “Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree.” *Eprint ArXiv*, 10 Oct. 2015.

Netzer, Yuval et al. “Reading Digits in Natural Images with Unsupervised Feature Learning.” (2011).

Sermanet, Pierre et al. “Convolutional neural networks applied to house numbers digit classification.” *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)* (2012): 3288-3291.