Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #6
April 28th, 2019

Feedforward CNN Design and Its Application to the MNIST Dataset

I. Abstract and Motivation

Kuo, et, al[1] supposed an explanation for the currently widely used back propagation CNN (BP-CNN) and develop a new CNN-like architecture with no BP needed. This architecture can also achieve a reasonable accuracy of hand writing recognition task. In

II. Approach and Procedures

FF-CNN

FF-CNN's structure is derived from LeNet-5 which is the first CNN. It has two components, convolutional layers and fully connected layers. The underlying mathematical principle of both type of components can be well explained.

Each convolutional layer consists several filters for image convolution operation and a max-pooling layer. To train the weights of filters, all the training images are used. By performing PCA onto all training image small patches collected by moving filter windows, we can obtain a transform matrix to convert the data representation to another filter space. The converted data is convolutional layer's raw output. By discarding some insignificant dimensions, original image's most part of information can be well preserved with less data dimension. We can choose how many dimensions well want to preserve by setting the number of filters. So that, elements in the transform matrix is the filter weights. However, before performing we need to extract the Direct Current (DC) component of each dimension of input patches, and this DC component will also become one additional convolution layer's filter. Other filters obtained by PCA are called AC components. Then, a bias is added to all dimensions of output by individual dimension. The bias is a scalar and is set to the same amount of the minimum value of all output elements of training data. This bias ensures all the output values are positive (in high probability for testing data) and eliminate the sign confusion which is a problem in BP-CNN. The max-pooling layer just pick the local most significant dimension and reduce the output data dimension which also introduce some nonlinear operations in the computation.

The fully connected layer is basically the same if we already have its parameters (weights and biases). However, the training process are different from BP-CNN. It uses K-means to cluster layer's input. The number of clusters are larger than the number of classes of the final output if the layer is not the last layer. It supposes images of same class have similar patterns and these patterns can be represented by previous convolutional layer and can be clustered into same sub class by K-means in high probability. To get the network parameters, how input sub classes are mapped to output sub classes, least

square regression is used based on the training data's true label.

At all, the design of FF-CNN only consists mathematics, and easy for people to understand what each parameter represents for.

Reconstruction

We are required to reconstruct the input sample images from the two stage convolutional layers' output. The max-pooling is canceled and the filter windows moving with no overlapping to ensure the inverse operation is not difficult. There are 3 steps in this task, no label data is used. First, we need to get the filter weights of two convolution layer's by feeding all training images into training process. Then get the output of two convolutional layers by feeding 4 sample images into these layers. Finally, do inverse operations to get reconstructed images. Since some insignificant components are discarded, the expected reconstruction results should be basically the same as the sample images with slightly difference due to the loss of insignificant information.

Doing PCA with all 60,000 training images requires huge amount of memory if no distributed optimization is proposed. Luckily, 10,000 training images is enough to get good filters for convolutional layers and avoid out of memory error. So, for all training process in this report, 10,000 training images is used to train convolutional layers' parameters, and all 60,000 images are used to train fully connected layers' parameters.

Ensemble of Feedforward Design

One way to improve the accuracy of output FF-CNN is to ensemble multiple FF-CNNs with different settings [2]. The key to improve result is to ensure the diversity of FF-CNNs, and [2] gives 3 ways to get the diversity.

After we get the output of all FF-CNNs, we can concatenate their 10D output together to get a new vector. Then perform another PCA to reduce its dimension. Then feed the reduced feature vector into SVM and train an SVM classifier. Thus, the output of this SVM will become the final output classification

This kind of ensemble can improve the accuracy, because multiple FF-CNNs can extract different kinds of features embedded in images, thus the provided information for making final decision is larger, so the final accuracy is higher.

III. Experimental Results

Reconstruction

I have two versions of implementations, one is based on provided Github code, and another is written totally by myself. They all work well. The following result is taken from the first implementation, that is because it is the official implementation of FF-CNN even it seems weird.

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #6
April 28th, 2019

						
original image	10/40 PSNR=22.980 666936719683	10/100 PSNR=29.460 493501033152	10/160 PSNR=34.100 295916087845	12/40 PSNR=22.870 843698165718	12/100 PSNR=29.592 944639859766	12/160 PSNR=34.655 339139254633
						
original image	10/40 PSNR=20.580 680677706777	10/100 PSNR=26.916 172332096466	10/160 PSNR=31.125 454248408104	12/40 PSNR=20.659 422055905008	12/100 PSNR=26.910 0545334084	12/160 PSNR=32.748 323367079564
						
original image	10/40 PSNR=20.352 495953647026	10/100 PSNR=26.493 573265386917	10/160 PSNR=30.458 211044568566	12/40 PSNR=20.148 281999705815	12/100 PSNR=27.168 108607060372	12/160 PSNR=31.736 201736979055
						
original image	10/40 PSNR=22.043 135190598328	10/100 PSNR=28.341 454773438187	10/160 PSNR=31.187 417590731936	12/40 PSNR=21.963 588196338630	12/100 PSNR=29.036 82606314840	12/160 PSNR=32.541 868520707660

Ensemble FF-CNNs

10 FF-CNNs settings

#	Laws	conv 1 window size	conv 1 num filters	conv 2 window size	conv 2 num filters	fc 1 num classes	fc 2 num classes
1	None	(5, 5)	5	(5, 5)	15	120	80
2	None	(3, 3)	5	(3, 3)	15	120	80
3	None	(5, 5)	5	(3, 3)	15	120	80
4	None	(3, 3)	5	(5, 5)	15	120	80
5	None	(5, 5)	10	(5, 5)	15	120	80
6	None	(5, 5)	10	(5, 5)	30	120	80
7	None	(5, 5)	5	(5, 5)	15	160	80

Zongjian Li

6503-3789-43

zongjian@usc.edu

EE 569 Homework #6

April 28th, 2019

8	L3L3	(5, 5)	5	(5, 5)	15	120	80
9	E3E3	(5, 5)	5	(5, 5)	15	120	80
10	E3L3	(5, 5)	5	(5, 5)	15	120	80

10 FF-CNN training and test accuracies (10,000 images for training convolutional layer due to the limitation of memory. result of first trail of each setting, parameters stored for further usage)

#	1	2	3	4	5	6	7	8	9	10
Training	0.9687	0.9682	0.9718	0.9707	0.9710	0.9759	0.970	0.9702	0.9485	0.9468
Test	0.9705	0.9672	0.9723	0.9713	0.9711	0.9760	0.971	0.9703	0.9484	0.9490

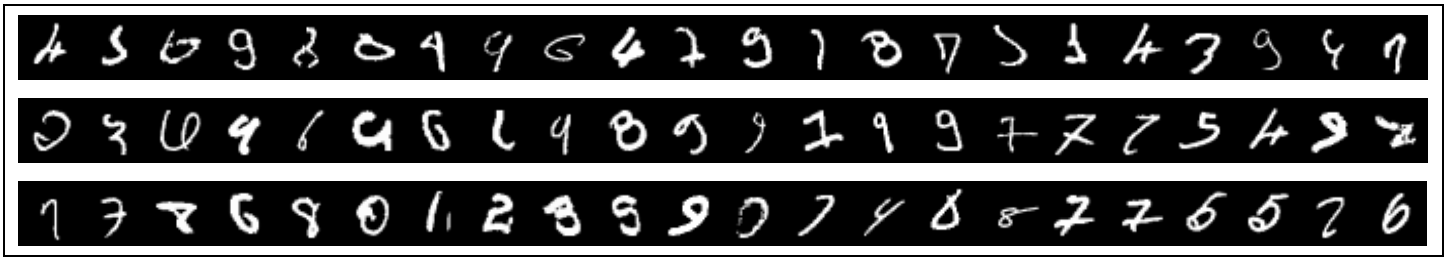
Ensemble systems' setting and training and test accuracies

Setting	Training	Test
PCA: 10*10D→60D, SVM: default	0.9818166666666667	0.9818

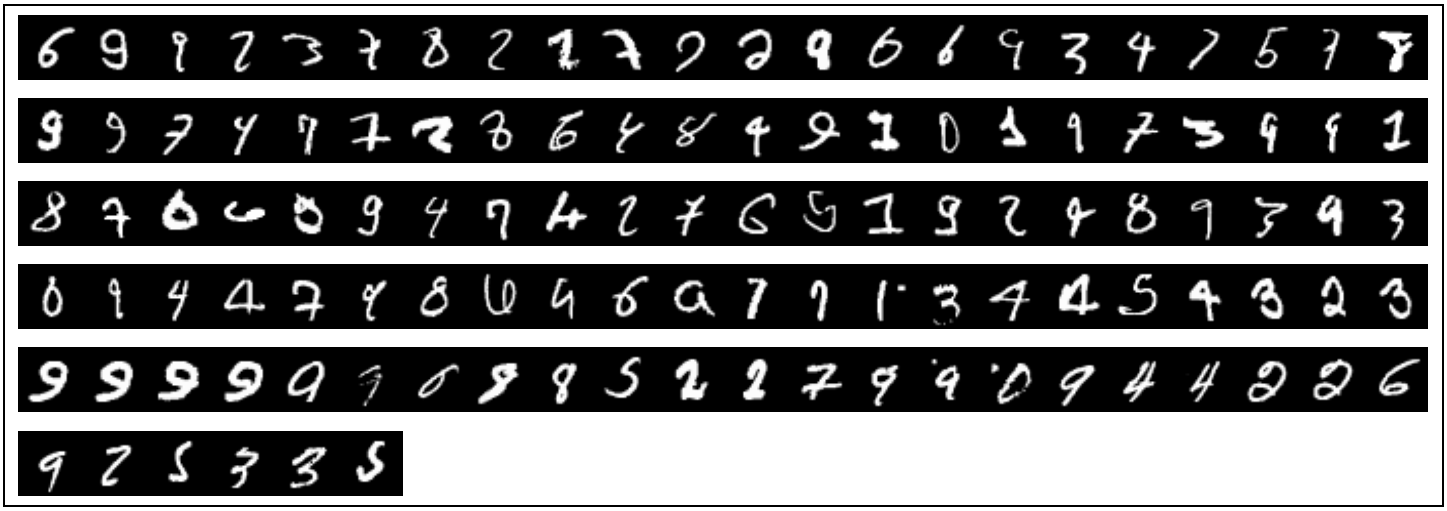
Reference LeNet-5 accuracies (parameters come from the first trail of training my best setting LeNet-5)

Training	Test
0.9982	0.9844

Set1: Test images that cannot be recognized by both CNNs (0.0066%)

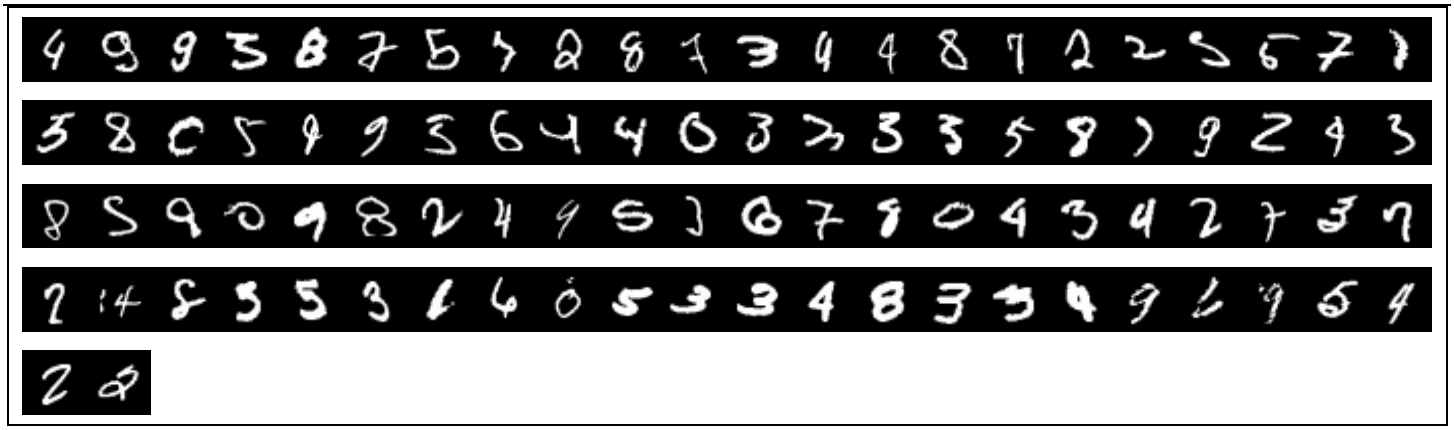


Set2: Test images that can be recognized by LeNet-5 but cannot be recognized by Ensemble FF-CNN system (0.0116%)



Set3: Test images that can be recognized by Ensemble FF-CNN system but cannot be recognized by LeNet-5 (0.0090%)

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #6
April 28th, 2019

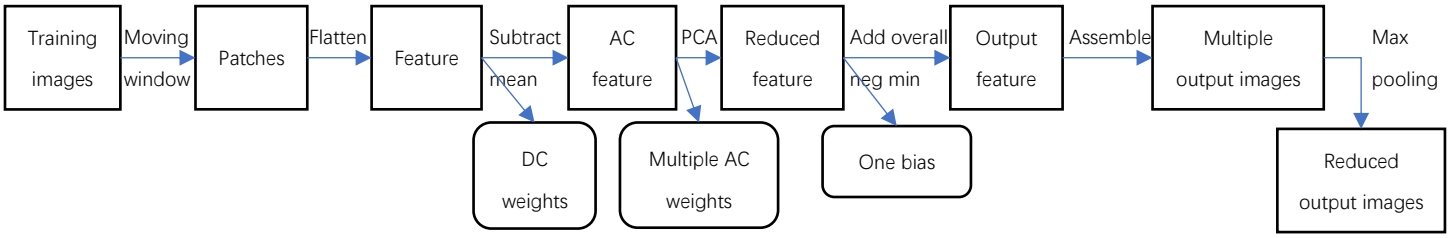


IV. Discussion

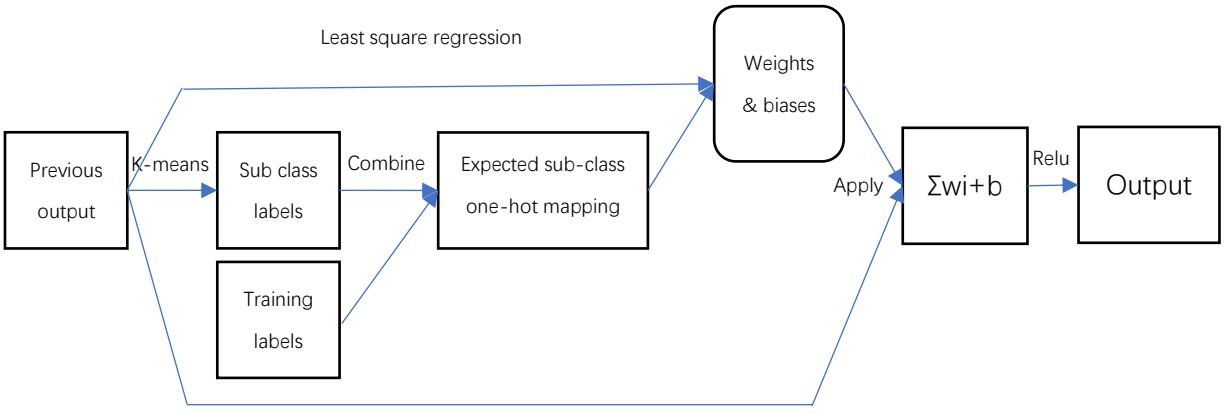
[Problem (1)-1] Summarize FF-CNNs with a flow chart and explain it in your own words.

Training flow charts

For each convolutional layer



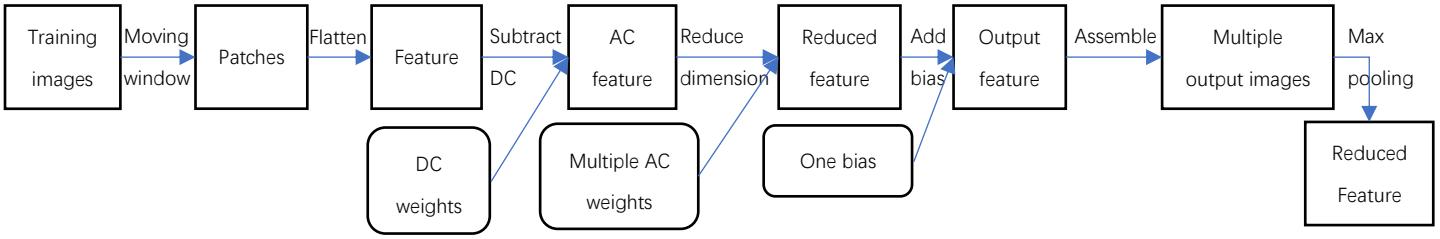
For each fully connected layer



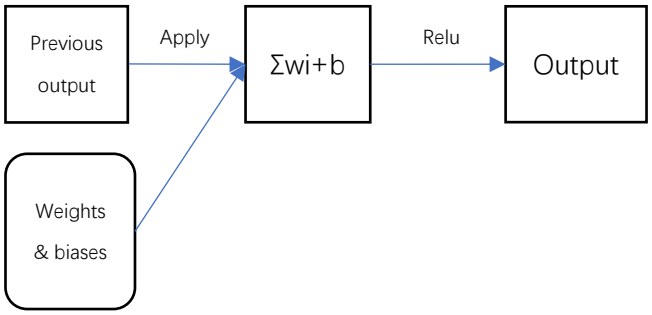
Testing flow charts

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #6
April 28th, 2019

For each convolutional layer



For each fully connected layer



Explain

Training and testing processes are different.

For training process, training convolutional layer and fully connected layer are different.

For training convolutional layer, all channels of training images are used to generate patches by moving filter window. For the first convolutional layer, there is only one channel. After we get all small patches, values are flattened to get features vectors. Then the mean of each feature is calculated as DC component. Subtract DC from feature, we get AC only feature. Then, apply PCA using all training AC feature to get a transformation matrix. This matrix can transpose feature from on special/spectral spaces to another. Since we want to reduce to feature dimension, non-significant components in the transformation matrix are discarded, only remaining number of set filter kernel components as AC components. After we get a reduced dimension feature, a bias is selected as negative minimum value among all elements in all training reduced feature vectors. That is because of the two FF-CNN's bias assumptions. Then, add this bias to all elements in feature vectors to make all elements are positive values. This procedure eliminates the sign confusion, so that Relu is not needed. Then, we assemble feature vectors back to a special representation. At last, max-pooling is used to further reduce the feature dimension.

For training fully connected layer, all training samples belonging each true final classes are further divided into several sub classes by K-means. This procedure showing in given paper and Github code is different, which make it hard to select one to explain in detail. However, After the K-means, for each training sample, we know a label for one overall sub-class and its true label. Then we can know our expected one-hot output of the fully connected layer. We perform least square

regression using input feature (with an addition column with all ones to calculate bias) and the expected output to compute the weights and biases. After we get these, we can compute the layer's true output by matrix multiplication operation. At last, Relu is used onto the output.

For testing process, testing images first go into convolutional layers then fully connected layers.

For each convolutional layer, small image batches are generated from layer's input images by sliding window. Then subtract and record their feature DC component, then transform into the successive lower dimension space (AC components) by multiply the transform matrix. Concatenate AC and DC components as the output, then add the trained bias for all elements. For each fully connected layer, since we already have the weights and biases, we just multiply the weights then add biases to get layer's output. Then apply Relu.

At last, we have a 10D vector, soft-max is used to pick the category with the highest probability as the final result.

[Problem (1)-2] Explain the similarities and differences between FF-CNNs and backpropagation-designed CNNs (BP-CNNs).

Similarities	Differences	
	FF-CNN	BP-CNN
Data driven: They both need training data to train parameters in network.	White box: All parameters are explainable. we can know why and how it works.	Black box: If something wrong, hard to know how to fix the network, since we do not know which parameter to tune
Convolutional Layers + Fully connected Layers: Conv layers extract features, and FC layers learn the mappings.	Perform well even with small training data set: Only FC layers need labeled training data, so fewer parameters are needed to be trained. Lower compute resource requirement.	Require large training data set: Must train parameters in Conv layers and FC layers, the searching space is too large.
Not robust against attacks: Since the training data cannot cover all false cases, and parameters are constants after training, targeted attacks are hard to prevent.	Only statistics and linear algebra: Basic mathematics, no BP. Transform between special space and spectral space.	Convex optimization problem: Search problem, BP needed

Zongjian Li
6503-3789-43
zongjian@usc.edu

Need to tune hyper-parameters: We need to set the architecture and evaluation function case by case.	Pluggable modules: Since we know how it works, so easy to modify CNN's structure. Such as FC layers can be replaced by other ML classifiers.	End-to-end training: Easy to use. Cannot be separate into parts and replace after trained.
	No Relu and labels needed in convolutional layers	Widely studied: May ready to use tools and conclusions.
	Good generalizability: Extract the most significant feature automatically.	

Relationship Between Convolutional Layer Setting and Reconstruct Results

We can see that if we preserve more components after PCA, the output has more dimensions and the image information loss is insignificant. So, with larger number of kernels, we can reconstruct images with higher PSNR and better visual clarity. If the number of kernels is small, the edges of digits are blurred, since the high frequency components are discarded, and edges are considered as high frequency information. If the first layer discarded too many components, it is hard to recover the lost information even if there are many kernels in second layer.

With a same setting, different images have different PSNRs. The first image always gets the highest PSNR and the third image always gets the lowest PSNR. I think that is because trained weights are good at extract features in the first image and not good at extract features in the last image.

Ensemble Strategy Explanation and Result Analysis

I ensembled 10 FF-CNNs, the first one is the default one, and #2-#4 changes its convolutional filter sizes, #5 & #6 changes its convolutional filter number, #7 changes the number of subclasses of fully connected layer, #8-#10 added a laws filter preprocessing to images. I change the parameters slightly different to the default one the see each parameter's influence of individual FF-CNN's accuracy. I choose these parameter modifications based on the methods in [2], it gives three kind of strategy to provide diversity. However, in my trails of different FF-CNN settings, the difference of accuracy is not very large. I picked several FF-CNNs with relatively high accuracy to assemble my final ensemble system shown above. Laws filter preprocessing FF-CNNs do not give high accuracies, I still added some of them (totally 9) into the ensemble system to provide potential diversity. I compared my ensemble result before and after adding laws preprocessing, the accuracy is almost the same. Since there are many hyper-parameters that can be adjust lead to too many combinations, the fine tuning process will be as same ignoring as that in BP-CNNs.

Zongjian Li
6503-3789-43
zongjian@usc.edu

For those FF-CNNs with different network structure, too small filter size will give lower accuracy, more filter kernels and sub-classes also increase the accuracy. That means, we can improve the accuracy by adding more parameters in this network, with same amount of training data.

For laws filter preprocessed FF-CNNs, only filter kernel L3L3 gives a high accuracy finally or lowest accuracy decay comparing with the default FF-CNN. Since L3L3 does not change the input image too much. [2] gives a same accuracy result. Although their ensemble system can give a higher accuracy, since our task is to ensemble 10 different FF-CNNs, so I just cannot put all 9 FF-CNNs into my final system, so I picked three of them.

I did not try the feature subset scheme in [2], since it is a little harder to encode the network and I do not have many available slots to add this kind of FF-CNN modifications.

An interesting thing is that, the ensemble system has a higher accuracy, higher than any of 10 FF-CNNs' accuracies. Ensemble system's inputs are these FF-CNNs, how can it get a higher accuracy. I think that is because different FF-CNN extract and evaluate on different *view of fields* of images – it has diversity of feature. Their extracted features totally different. Some information resists in these combinations of features rather than a set of features with similar "shapes" of feature. So, after I ensemble these FF-CNNs, the hidden information can be discovered by SVM. Thus, the final accuracy is higher.

Analysis of Difference of LeNet-5 and Ensemble System's Error

I cannot use all 60,000 training images to train conv weights, so some useful feature may hide beneath unselected training images. So, the accuracy is lower than that in papers [1][2]. However, the accuracy is much higher than I expected, with such a simple and interesting network architecture.

Under the above situation, I compared the result both from my ensemble system and a well-trained raw LeNet-5 BP-CNN. Both systems have similar accuracy. There are 4 sets of images, I listed images that is not well recognized by either or both systems. I found most of these images are not difficult for me to recognize, so why these systems cannot recognize them even they have may training data? I think that is because they do not understand digits' shapes, they just consider pixel values. So, I think they did not really understand what they learnt.

Some images cannot be recognized by both kind of CNN, has disproportionate strokes. Their feature stroke is too small, so it is hard to be captured by CNNs. For other two sets, I just cannot see the difference of their common feature between sets.

We can see that LeNet-5 get very high accuracy on training dataset, and similar accuracy to the ensemble system's on testing dataset. So, BP-CNN do not have good generalizability and tends to overfitting training data.

Improvement Proposal and Justification

Since training convolutional layer weights needs PCA, and it is hard to fit all matrix into memory, I propose to do distributed SVD rather than plain SVD to solve the memory issue, so that FF-CNN can be improved since I can load more

training data. The are already several papers talk about how to calculate SVD in smaller matrix operations. Some ML tools have provided ability to solve very large matrix SVD, such as Apache Spark MLlib and Apache Mahout, we can use them to accelerate training. Since FF-CNN based on PCA, and PCA is based on SVD, and SVD can be solved on these systems faster and easier with large training data.

Furthermore, the sample FF-CNN codes' structure is not designed well, it always makes people confusing. I think if the published sample code for a paper is easy to read with good document and concise statements, thoughts in that paper will be easier to be understood and arouse the interest of others. Here are my improvement suggestions. Some functions are implemented in redundancy, separate them into several functions and use numpy built-in functions rather than slow python native implementations. Some variables' names are not straight forward. There are too many hardcoded constants that make the program hard to be generalize although FF-CNN has good modularity ability. And, there are too many meaning less console outputs. At last, it is written in mixing of Python2 and Python3, which is absurd. I tried to rewrite the code in a better way in my homework, however, the architecture problem is something I do not have time to solve.

To improve both BP-CNN and FF-CNN, we can do data augmentation to generate more training images from existing data. So that networks can learn more detailed features of digits, especially from some special training images.

References

[1] Kuo, C. C. J., Zhang, M., Li, S., Duan, J., & Chen, Y. (2019). Interpretable convolutional neural networks via feedforward design. Journal of Visual Communication and Image Representation.

[2] Chen, Y., Yang, Y., Wang, W., & Kuo, C. C. J. (2019). Ensembles of feedforward-designed convolutional neural networks. arXiv preprint arXiv:1901.02154.