Homework Six Report

Zifan Wang

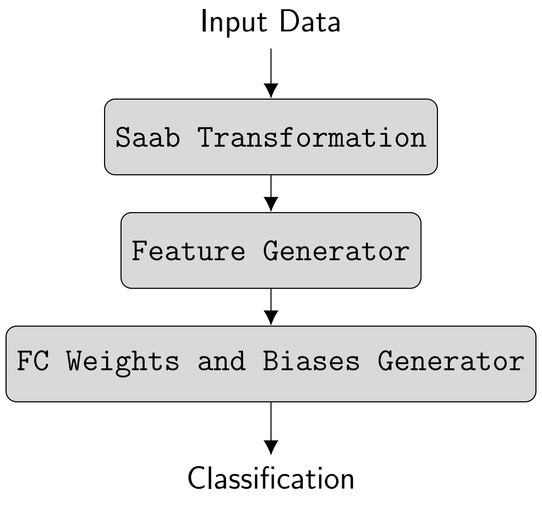
April 28, 2019

1 Problem 1: Understanding of feedforward-designed convolutional neural networks (FF-CNNs)

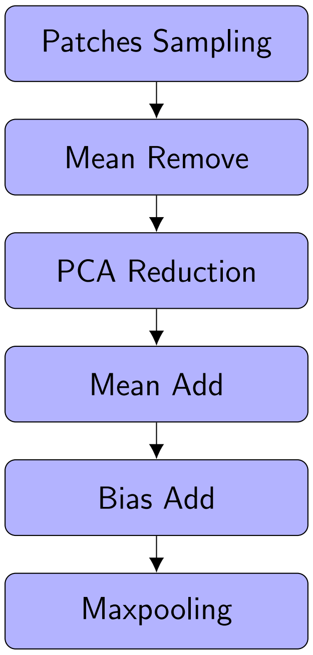
1.1 Abstract and Motivation

Traditional convolutional neural network (CNN) relies on back-propagation which can be viewed as convex optimization to train the parameters. The quality of the trained network heavily depends on the dataset. Take the leNet-5 CNN in the last homework as an example. The network trained based on the MNIST dataset was not able to detect digits whose color were reversed from the MNIST dataset. To reduce the dependence of the dataset, a **feedforward convolutional neural network** (FF-CNN) is developed. Instead of using the back-propagation to model parameters, the FF-CNN applies a data-centric approach to learn these parameters. A new signal transformation, called the **Subspace approximation with adjust bias** (Saab), was developed to construct the convolutional layer. The multiple convolutional layers in FF-CNN is are cascaded by multi-stage Saab transformations. The **linear least squared regressor** (LSR) is used to construct the fully-connected (FC) layer and multiple LSRs are used to build multi-stages FC layers in the FF-CNN [1].

1.2 Approach and Procedures

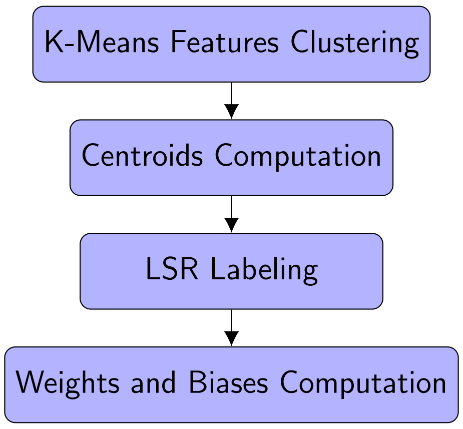


Inspired by the back-propagation convolutional neural network (BP-CNN), the FF-CNN has the same architecture as the BP-CNN which contains mutli-stages convolutional layers and FC layers. The only difference between the FF-CNN and the BP-CNN is that the FF-CNN uses data statistics method instead of back-propagation processes to learn weights and biases. The flow chart above shows the process of FF-CNN. The Saab transformation is the method to construct one convolutional layer. The flow chart below shows the process of Saab transformation:



Multiple patches are generated by input images, window size (k, k), and stride s. For example, the training dataset with size (N, C, H, W) where N is number of training images, C is number of color channels, and H *and* W are image's height and width can be sample into a patch space with size $(N, \frac{H-k}{s}+1, \frac{W-k}{s}+1, C*k*k)$. The benefit of using patches is that patches not only do the pixel-wise comparison in the center pixel, but also it takes center pixel's neighborhood into consideration. This will help Saab transformation be invariant to translation and rotation. The patch space can be categorized into two anchor vectors: DC (direct current) which is the mean of data space and AC (alternating current) vectors. Remove mean from the patch space and flatten it to the 2-D space with size $(N * \frac{H-k}{s}+1 * \frac{W-k}{s}+1, C*k*k)$. PCA, then, will be performed on this data 2-D data space to generate n kernels which are n eigenvectors corresponding to n largest eigenvalues. The energy of these n kernels dominates the energy of whole data space. Thus, they can be used to represent the data space. After PCA, add DC component to the n kernels to construct the kernel in the convolutional layer. That is, in the convolutional layer, kernel is in the shape of $(n+1, C*k*k)$. The largest norm of current features, bias, is added to the sampled patches to generate input images in the next stage. This bias aims to reduce sign confusion and derive new images. Moreover, this computed bias is the bias in the convolutional layer. Same to the BP-CNN a max-pooling process is performed in FF-CNN to preserve important patterns and filter out insignificant ones. Cascade the Saab transformation to construct the multi-stages convolutional layers.
Before computing weights and biases in the FC layers, the input features of the FC layer have to be generated. According to the input data, and kernels and biases calculated

in the Saab transformation stages, the input features are computed by (input data plus biases) multiply with kernels and reshape into 4-D shape.
Last step of FF-CNN is computing the weights and biases in the FC layers. This process aims to relate input data to the classification labels. The following flow chart represents the process of finding weights and biases in the FC layers:



First, K-means clustering is applied to cluster input features and connect these features with labels. Second, based on the results from K-means clustering, n centroids which is the input features of the next stage will be computed, which corresponding the number of neurons in the FC layer in the BP-CNN and be related to the pseudo labels. Last, the LSR is applied to generate weights and bias. The multi-stages FC layers are constructed by cascading those processes.
A FF-CNN will be initialize in Keras based on those kernels, weights, and biases.

1.3 Discussion

Explain the similarities and differences between FF-CNNs and backpropagation-designed CNNs (BP-CNNs)

- Differences:
  The major difference between FF-CNNs and BP-CNNs is the different training method of kernels, weights, and biases. The BP-CNNs uses back-propagation which using stochastic gradient descent to optimize kernels, weights, and biases in each layer. The back-propagation process highly relies on datasets, network architecture, and cost function. In mathematics, the BP-CNNs can be viewed as a non-convex optimization problem. However, the FF-CNNs applies the PCA and the LSR operations to learn those parameters and is more relative to linear algebra and statistics problem. Also, BP-CNNs' training accuracy highly relies on number of training data, but the FF-CNNs do not require large amount of data to get high training accuracy. As a result, the training complexity of BP-CNNs is higher than FF-CNNS.
  Another difference is the vanishing gradient problem. As we all known, when the BP-CNNs become deeper and deeper, the vanishing gradient problem tends to happen. However, using linear algebra and statistics method, the FF-CNNs will less likely to have this problem.
  Moreover, the different mathematics method makes FF-CNNs easy to interpret. During years and years researches in BP-CNNs, researchers can not generalize a

full explanation of BP-CNNs. While, FF-CNNs using different weights and biases generation method are easy to understand.
In addition, the FF-CNNs have more potential architectures to use. For example, the FC layers in the FF-CNNs can be replaced by the random forest and the support vector machine classifiers. However, the BP-CNNs are end-to-end classifiers. The architectures are limited.

- Similarities:
  Both BP-CNNs and FF-CNNs can be used to classify objects. They performs much better than the traditional objects classifiers in the traditional computer vision and image processing fields. The classification performance of these two CNNs are similary when given the same dataset. However, neither BP-CNNs or FF-CNNs are robust against adversarial attacks. Heavily relying on data, the BP-CNNs can be easily fooled by given them the data with different patterns which they never saw before. Same to the FF-CNNs, they also try to map the given input data to the classification label. Powerful adversarial attacks can fool the classification systems in both networks. This because both network all learn some noises which are not necessary to be used to classify objects. When a powerful adversarial attack happens, the learned noises will actually fool the network.

2 Image reconstructions from Saab coefficients

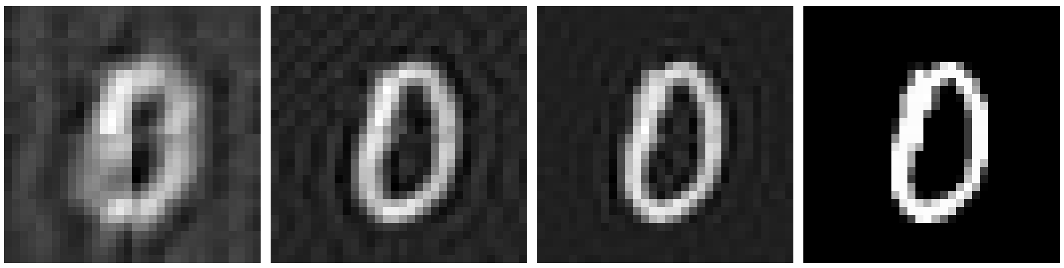
2.1 Abstract and Motivation

One way to examine the performance of FF-CNNs is to reconstruct images from Saab coefficients. In this section, a two stage Saab transformations with all 4 by 4 kernels' size and stride 4 will be used to see the performance of Saab transformations. The only variable in this section is the number of kernels in each Saab transformation.

2.1.1 Approach and Procedures

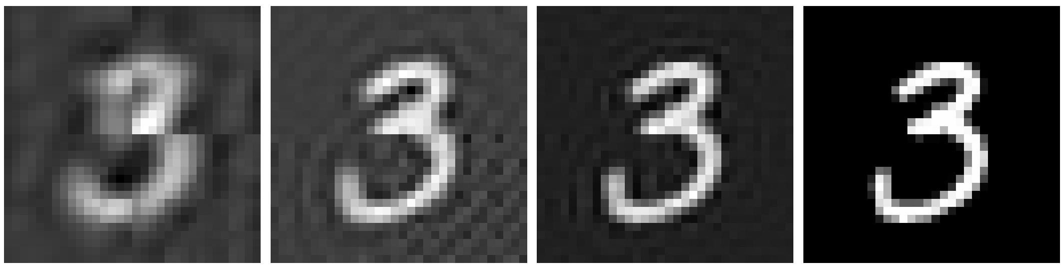
Based on the code from davidsonic in github [2], a reversed Saab transformation is developed. The method of Saab transformation is dicussed in the Section 1. To reversed Saab transformation, the only thing need to do is that apply *pinv* method in numpy library to inverse the projections. The followed result section shows different reconstructed images with different number of kernels in each layer.

2.2 Result



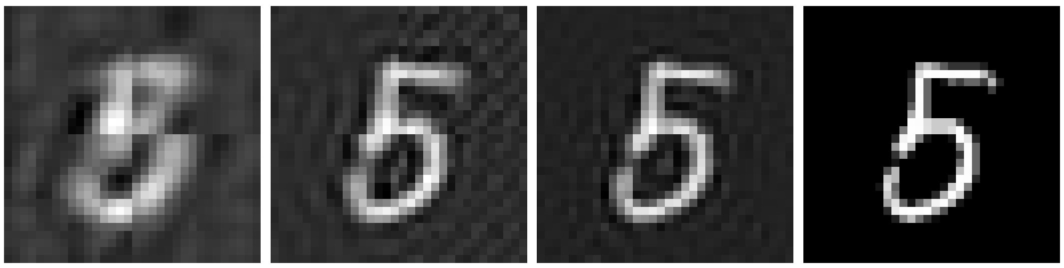
(a) $(6, 16)$,PSNR=15.7 (b) $(10, 60)$,PSNR=21.6 (c) $(10, 100)$,PSNR=25.2 (d) $(16, 256)$,PSNR=219

Figure 1: Digit 0 Reconstruction



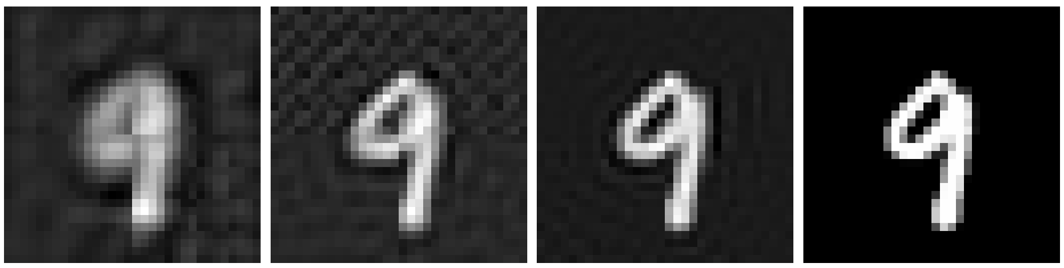
(a) $(6, 16)$,PSNR=16.7 (b) $(10, 60)$,PSNR=22.6 (c) $(10, 100)$,PSNR=26.3 (d) $(16, 256)$,PSNR=222

Figure 2: Digit 3 Reconstruction



(a) $(6, 16)$,PSNR=16.6 (b) $(10, 60)$,PSNR=21.6 (c) $(10, 100)$,PSNR=24.7 (d) $(16, 256)$,PSNR=222

Figure 3: Digit 5 Reconstruction



(a) $(6, 16)$,PSNR=18.2 (b) $(10, 60)$,PSNR=22.7 (c) $(10, 100)$,PSNR=27.9 (d) $(16, 256)$,PSNR=222

Figure 4: Digit 9 Reconstruction

2.3 Discussion

The figures 1, 2, 3, and 4 show the reconstruction results. The d setting which uses $(16, 256)$ kernels which uses all kernels in the PCA. This setting return the highest PSNR value because it preserve all energies. From the setting a to the setting d, we can notice that when number of kernels increases, the PSNR values become higher and the reconstructed images are more similar to the input one. We do not have to use all the kernels in the network. Because the c setting which has 10 kernels in the first convolutional layer and 100 kernels in the second convolutional layer reconstructed images well enough. This is because 10 kernels' energies domain the energy of the first layer and 100 kernels' energies domain the energy of the second layer.

3 Handwritten digits recognition using ensembles of feedforward design

In this section, performance of individual FF-CNN and ensemble FF-CNN will be evaluated. Using the same configuration of leNet-5, this FF-CNN has two convolutional layers. The first convolutional layer has kernel size $(5,5)$, stride 1, and 6 kernels. The second convolutional layer has kernel size $(5,5)$, stride 1, and 16 kernels. Two FC layers with 120 and 80 neurons respectively. A output layer with 10 neurons.

3.1 Individual FF-CNN

Five tests are performed in this section.

	1	2	3	4	5	Mean	Variance
Train Loss	1.5843	1.5853	1.5846	1.5845	1.5851	1.585	1.6e-7
Train Accuracy	0.9688	0.9687	0.9689	0.9680	0.9684	0.968	9.8e-8
Test Loss	1.5786	1.5791	1.578849	1.5777	1.5791	1.58	2.9e-7
Test Accuracy	0.9702	0.9693	0.970300	0.9698	0.9684	0.969	4.8e-7

Table 1: Individual FF-CNN Performance

The individual FF-CNN has average 0.968 training accuracy and 0.969 testing accuracy. It has lower training and test accuracy than the BP-CNN. This is because the error during the PCA and K-means clustering. That means the PCA will loss some important information. Also, K-means is a not supervised learning and therefore we cannot make sure each label is correctly been pseudo labeled.

3.2 Ensemble FF-CNN

To improve the performance of FF-CNN, an idea of ensemble ten different FF-CNNs is adopted. Separate training data into 10 groups and each has 5000 images. Same to the individual FF-CNN, five tests are performed in here.

	1	2	3	4	5	Mean	Variance
Ensemble Accuracy	0.9751	0.9753	0.9755	0.9754	0.9753	0.975	1.8e-8

Table 2: Ensemble FF-CNN Performance

The mean ensemble accuracy shows that the ensemble FF-CNN performs better than the individual FF-CNN. To further improve the FF-CNN's performance, we can increases the diversity of the baseline FF-CNNs. The paper [3] proposes three different ways to increase this diversity: flexible parameters in convolution layers, subsets of derived features, and flexible input image forms. In this section, I use the thrid method flexible input image forms to increase diversity. That is, I apply nine $3 * 3$ laws filters to images. The table 3, 4, and 5 below is the performance results:

Acc	Raw	L3L3	L3E3	L3S3	E3L3	E3E3	E3S3	S3L3	S3E3	S3S3
Train	0.976	0.975	0.960	0.935	0.966	0.960	0.929	0.942	0.938	0.891
Test	0.963	0.963	0.946	0.905	0.953	0.945	0.902	0.923	0.908	0.858

Table 3: Ensemble FF-CNN Performance with Diversity Increase First Test

Acc	Raw	L3L3	L3E3	L3S3	E3L3	E3E3	E3S3	S3L3	S3E3	S3S3
Train	0.974	0.975	0.959	0.937	0.967	0.963	0.928	0.942	0.937	0.889
Test	0.964	0.963	0.947	0.904	0.954	0.942	0.897	0.921	0.908	0.858

Table 4: Ensemble FF-CNN Performance with Diversity Increase Second Test

Acc	Raw	L3L3	L3E3	L3S3	E3L3	E3E3	E3S3	S3L3	S3E3	S3S3
Train	0.975	0.975	0.957	0.934	0.966	0.963	0.929	0.943	0.938	0.889
Test	0.963	0.968	0.949	0.905	0.958	0.946	0.898	0.925	0.912	0.856

Table 5: Ensemble FF-CNN Performance with Diversity Increase Third Test

The following table 6 shows the training and ensemble results of three test above.

	1	2	3	Mean	Variance
Training Accuracy	0.9836	0.9826	0.9830	0.983	1.7e-8
Testing Accuracy	0.9714	0.9727	0.9723	0.9721	2.9e-7

Table 6: Ensemble FF-CNN with Diversity Increase Performance

The ensemble system with diversity increase has average 0.983 training accuracy and average 0.9721 testing accuracy. The ensemble system without diversity increase has average 0.975 training accuracy. The idea behind why ensemble system performance better than the individual one is that the ensemble system rate different feature with different scores. Some features in the images are more important in classify objects. As a result, both of my ensemble systems apply a confidence score system to rate different features. This system can be used to improve the performance of FF-CNNs.

3.3 Error analysis

Comparison between these three FF-CNNs and BP-CNNs can be helpful for knowing the performance of FF-CNNs. The following table shows this comparison.

	Individual	Ensemble	Ensemble(DI)
Training Acc	0.968	0.975	0.983
Testing Acc	0.969	0.970	0.972
Number of Different	295	291	289
%Same Errors	0.6944	0.699	0.701
%Different Errors	0.205	0.220	0.206

Table 7: FF-CNN v.s. BP-CNN

The lower training accuracy and testing accuracy in the FF-CNN are because the PCA do not fully preserver all kernels. Both FF-CNN and BP-CNN has error results. This is because during the learning processes, both methods learn some noises and these noises make the prediction wrong. In addition, some images in the MNIST dataset are confused. Compare between individual FF-CNN and ensemble FF-CNN. We can notice that the ensemble FF-CNN performance better than the individual FF-CNN. This is because ensemble FF-CNN average multiple good models.

3.4 Discussion

- BP-CNNs:
  To improve BP-CNN, one method is that ensemble different models. Combine the predictions from multiple modes. The good performance can be achieved by combining the predictions from multiple well-performed modes rather than a single model. Proved in the section 3.2 the ensemble FF-CNN has better performance than the individual one. Same to the BP-CNN, the better performance can be achieved by ensemble multiple well-performed models.

- FF-CNNs:
  To improve FF-CNN, one method is that replace PCA by latent Dirichlet allocation (LDA). The LDA method uses labels to reduce data space dimension. However, the PCA uses energy or defined number of kernels to reduce data space dimension. I believe this method would help because this is a classification problem. Based on the labels to compute kernels would return a better results.

References

[1] C.-C. Jay Kuo, Min Zhang, Siyang Li, Jiali Duan and Yueru Chen. *Interpretable Convolutional Neural Networks via Feedforward Design.* Journal of Visual Communication and Image Representation, March 2019

[2] davidsonic *https://github.com/davidsonic/Interpretable_CNN*

[3] Yueru Chen, Yijing Yang, Wei Wang and C.-C. Jay Kuo. *Ensembles of feedforward-designed convolutional neural networks.*Jan 2019