EE569: Homework #6

Yifan Wang
wang608@usc.edu #3038184983

April 8, 2019

1 Understanding of FF-CNNs

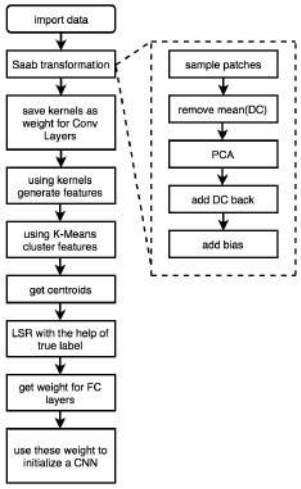

Figure 1: Procedure

1.1 FF-CNNs

Feedforward design convolutional neural network (FF-CNN) aims to find weights through one feedforward pass rather than through back propagation, then using weights as a pre-trained network weight to apply on test sample directly. Detail procedures are as following ($Figure1$):

Weights of convolutional layer come from Saab transformation. Train data with shape (N, C, H, W) can be separated into many small patches by using window size (k, k) and stride s. For one single image $(H \times W)$ $P = (\frac{H-k}{s}+1) \times (\frac{W-k}{s}+1)$ patches can be derived, so that whole data set's dimension is (N, P, c, k, k). Before doing PCA on these data, both patch mean and sample mean should be removed, and reshape it become a $2D$ tensor with shape $(N \times P, c \times k \times k)$. Doing PCA on this data to get M kernels (eigenvectors corresponding to M largest eigenvalues) each with shape $c \times k \times k)$. Then put DC part back to these kernels (total $M + 1$). To compute features for next stage, bias (largest norm of current features) is added to these images which is explained in the paper to avoid sign confusion and using the kernel to project features into new features. Deriving new feature patches which can be used for next stage Saab transformation. Repeat this process resulting multi-stage Saab transformation which is basically doing affine PCA on AC part on small patches (which contains DC part) again and again.

After get final features which do not have any information related to labels. To connect them with provided train labels, K-Means is used to find clusters and connect them with labels. Instead of directly cluster features

into number of classes needed, multi-stage k-means are used to mimic fully connected layers to give it more flexibility. After using K-Means, some pseudo labels are generated and using least square regression to get weight and features (centroids) for next stage. Repeat this process, weights for fully connected layer can be derived from least square regression's weight.

To use this network, weight derive from above procedure are used to initialize a normal CNN. And just use the feedforward path to make predictions. This is due to the fact convolutional layer can be regarded as a kind of dot product in small patches and FC layers can be regarded as a kind of linear space transformation.

1.2 Difference with BP based CNN

Major difference between FF-CNN with BP-CNN is it do not need back propagation part to train network which is initialized by random numbers as weights and optimized by using SGD. FF-CNN using multi-stage subspace PCA , K-Means, and least square regression to get weights and use them to initialize weights in the same network architecture directly.

For traditional CNN which uses back propagation to find optimized weights performs good however it is difficult to understand especially when network is deep (also difficult to initialize and optimize (DenseNet and ResNet partly solved this problem)). Even though lots of effort has be made to explain how it works, this remains a field where applications grow faster than theories. While for FF-CNN, it would be relatively easy to explain. But it needs to compute $pinv$ or norm which would take super long time when dimension is high. Besides, CNN needs lots of data to give better performance, while this method is said to give nice performance with far few data. It do not need any label to generate weight for a convolutional layer so that features may be suitable for all kinds of data.

Both network are not robust to attack which I explain it as side effect of not being able to get accurate number of parameters need. Most network trends to have more parameters than fit actual data distribution needs (detail in `https://github.com/yifan-fanyi/interpret_CNN`). Extra parameters give model more freedom which can be dominated by noise. Besides, these weight matrix are likely not well in condition (large condition number), so that, some small change in input which is not visible to human can be amplified layer by layer with a super large error in output layer (a flavor of butterfly effect).

2 Image reconstructions from Saab coefficients

In this problem input images have shape $(32, 32)$ and two stage window each has size $(4, 4)$ with stride 4. Suppose in first stage M_1 kernels and M_2 in second stage (include DC kernel) are kept. Let's analysis from forward pass, in first stage, 64 patches can be driven after reshape, a patch stack with size $(64, 16)$ is extracted from one image. After choosing first M_1 kernels, a project matrix with shape $(16, M_1)$ ($M_1 \leq 16$) is derived. Project patches to this sub-space, patches are reduced to $(64, M_1)$. To get patches for second stage, processed patches should be reshape back to $(8, 8, M_1)$. Doing the same process again, we can get $(4, 16M_1)$ flattened new patches for second stage PCA. Project matrix (kernel) for second stage is $(16M_1, M_2)$. Final feature would be $(4, M_2)$. To get reverse transformation, what we should do is to using $pinv$ of project matrix and carefully reshape patches. Bias should be included in above calculation as well (just change plus to minus).

Results of reconstruction with different Saab coefficients are show in $Figure2$. It can be found that if using all kernels, PSNR would be super high since all energies are preserved. Otherwise each stage would give a discount to total energy. If using like this without max-pooling or over-lapping. To some extent, it can be regard as one single stage PCA with patch size $(16, 16)$ stride 16 (slightly different in patch bound, some correlation among patches would be removed if using two stage). However, without any overlapping trends to lose some information on patches bound which can be found in $Figure2(e, j, o, t)$. Patches would not consist with each other after inverse transformation especially on boundaries, since PCA destroy correlation information. SNR can also be estimated by how many energies preserved, similarly $PSNR$ is related to energy preserved as well. More energy preserved (more kernels), higher $PSNR$ and SNR. Image quality seems nice even when using only small part of kernels each stage, since most energies are concentrated in these principle components.

(a) (16, 256), PSNR=303.45 (b) (16, 128), PSNR=30.46 (c) (8, 128), PSNR=28.64 (d) (8, 64), PSNR=24.28 (e) (8, 16), PSNR=17.55

(f) (16, 256), PSNR=302.20 (g) (16, 128), PSNR=28.40 (h) (8, 128), PSNR=26.53 (i) (8, 64), PSNR=22.27 (j) (8, 16), PSNR=15.49

(k) (16, 256), PSNR=303.98 (l) (16, 128), PSNR=28.12 (m) (8, 128), PSNR=25.27 (n) (8, 64), PSNR=21.95 (o) (8, 16), PSNR=16.73

(p) (16, 256), PSNR=303.06 (q) (16, 128), PSNR=29.56 (r) (8, 128), PSNR=26.30 (s) (8, 64), PSNR=23.31 (t) (8, 16), PSNR=16.25

Figure 2: Influence of different kernel number

3 Test FF-CNNs

3.1 Single FF-CNN

		#1	#2	#3	#4	#5	Overall	Variance
train	acc	0.9701	0.9695	0.9699	0.9702	0.9702	0.9700	6.9e-8
	loss	1.5820	1.5826	1.5822	1.5824	1.5819	1.5822	6.5e-8
test	acc	0.9712	0.9707	0.9709	0.9708	0.9714	0.9710	6.8e-8
	loss	1.5763	1.5773	1.5764	1.5769	1.5772	1.5768	1.6e-7

Table 1: test accuracy on LeNet-5 structure in MNIST

For one FF-CNN trained on all 60000 MNIST images, kernels for convolutional layers would be the same along different run. Test accuracy can be found in $Table1$. Weight would slightly change in FC layers due to different initialization of K-Means. It can be found train and test accuracy is around 0.97, while train and test loss is much larger than a BP-CNN (around 0.05). This fact mainly comes from that FF-CNN can not give high

confidence to one prediction. If print softmax activation for each class of one single image, it can be found the highest activation would be about 0.2 to 0.3, while other are around 0.09 which means that FF-CNN is also not sure about prediction. However, in BP-CNN the softmax activation for the largest class can achieve more than 0.9. So that according to definition of loss function, the loss would much lower. To some extent, it shows features from FF-CNN are more general while BP-CNN gives more specify feature according to training data.

3.2 Ensemble FF-CNN

To get a ensemble FF-CNNs as homework require, 10 single FF-CNNs are needed. This method is to use bagging like how several BP-CNN ensemble. Basic idea is to separate whole training set or whole features into small parts. Then using these individual part to train several weak learners. Finally, using voting method or channel PCA from homework instruction to get final predictions among these weak learners.

ensemble	$\#1$	$\#2$	$\#3$	$\#4$	$\#5$	Overall	Variance
Separated Data	0.9759	0.9761	0.9761	0.9756	0.9765	0.9760	8.6e-8

Table 2: Ensemble LeNet-5 structure using MNIST

First method I tried is to separate training data. It is separated into 10 balanced subset with 5000 images where each label has 500 images used to train a single network. Second idea is using different features to train a network and combined as it instructed. In this homework 9 response from Laws filters and raw image are used to train 10 network each used one feature.

	raw	L5E5	L5S5	L5W5	L5R5	E5L5	E5E5	E5S5	E5W5	E5R5
1^{st} Stage	0.79	0.92	0.89	0.83	0.79	0.91	0.79	0.74	0.66	0.67
2^{nd} Stage	0.86	0.86	0.77	0.69	0.66	0.84	0.70	0.67	0.64	0.66l
Train Acc	0.9704	0.9527	0.9303	0.9047	0.8685	0.9614	0.9549	0.9376	0.9178	0.8790
Test Acc	0.9714	0.9534	0.9310	0.9033	0.8659	0.9608	0.9535	0.9383	0.9138	0.8779

Table 3: Some results on 10 sub-network of ensemble method $\#2$, $kept_kernel = (5, 15)$, $patch = (5, 5)$, $stride = 1$

For first method, test accuracy for each network is around 0.9640, while train accuracy is around 0.98. This implies when provided small training dataset, network trends to overfit as well. On the other hand, test accuracy does not drop so dramatically compare with dramatic shirking in size of training data. It shows features extracted by FF-CNN is more general which BP-CNN can get more features specify to train data. So that BP-CNN would give good performance only when probability distribution of train and test data match nicely. FF-CNN may give moderate performance when train-test not match well.

Ensemble 10 network together can increase test accuracy to 0.9760 ($Table2$) using method one (using voting method to ensemble). For one single test, 267 test images out of 10000 test data have conflict predictions between BP-CNN and FF-CNN. Among these conflicts BP-CNN has accuracy of 0.7310, while FF-CNN only achieve 0.2380. Both network give wrong prediction to the rest 0.0310. Different may account for performance gap between BP-CNN and FF-CNN. Explanation might be BP-CNN would gain more features which have larger discriminate power and more specify to train data according to above discussion, since there would be cases feature which has most discriminate power does not have large energy power. However, FF-CNN is based on energy, some of detail may be discarded and it can not extract other more features unless it is provided with pre-processed feature images.

For second method, train and test results are shown in $Table3$ and final test accuracy of 3 trials are shown in $Table4$. In this test, C-PCA is used on 10 concatenated output followed by a least square regression as [3] used. It would perform almost the same as method one. For some features from Laws filter would have relative low accuracy compare with others (lower energy preserved, lower accuracy), while ensemble them together

	#1	#2	#3	Overall
train acc	0.9755	0.9756	0.9751	0.9754
test acc	0.9763	0.9759	0.9764	0.9762
#diff pred vs. BP	247	250	249	248
BP acc in diff pred	0.7085	0.7120	0.7068	0.7091
FF acc in diff pred	0.2550	0.2480	0.2610	0.2547

Table 4: Ensemble FF-CNN using more features

gives moderate performance. Conflict predictions among BP and FF CNN are similar to previous one. In that case, FF-CNN

3.3 Improvement

To make some improvement to single FF-CNN, there are two opposite ways regarding to labels. One is that it would be a kind of waste when generating kernels for convolutional layers without using labels provided. In that case, LDA may be used to replace PCA, which can solve the problem that some features with strong discriminate power while have small energies would be discarded by PCA. On the other hand, considering the high cost of get all data labeled, semi-supervised FF-CNN can be developed based on non-label related features and K-Means used in FF-CNN. It can be used to combine with BP-CNN to generate some semi-supervised CNN.To slightly increase accuracy of ensemble FF-CNN, train accuracy can be used as a discount rate to softmax output of each sub-network and emphasis more on wrong predicted images like boosting.

Here are some my personal ideas relate to this paper. It would be nice if we can design a good classifier with in a interruptible manner. However, more test should be carried out on larger images and larger scale networks. Another quite important issue is when simply using lots of stages of Saab transformation, it would suffer from vanishing energy problem (similar to vanishing gradient problem). Some other method should be found to make network deeper. It comes from when selecting small part of kernels each stage which gives large discount rate to total energy. Choosing lots of kernels where computing $pinv$ and K-Means would take super super long time. For example, using default setting of raw code ($\#kernel = (5, 15)$, $stride = 1$, $window = 5 \times 5$, $\#label = (120, 84, 10)$), it takes 4 minutes to get kernels and more than 8 minutes to get weight on a AWS c4.8xlarge instance using 60000 MNIST training data. Similar running time issue would be tough problem when input image is large.

It seem quite contradict to compare with these two approaches. Ideas in FF-CNN would be helpful to understand how CNN works. However, one can not beat a BP based CNN by just mimic its behavior which is highly optimized. On the other hand, highly optimized means once test data's probability distribution change slightly, test accuracy would drop dramatically. While some low level feature gained from FF-CNN would be more robust to such kind of change.

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] `https://github.com/davidsonic/Interpretable_CNN`

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