Spring 2018	Report#9 - 03/08/2018	Tianxiao Zhang

I. Tasks achieved Last Week (***, **, *: order of priority)

■ Project: Content-Adaptive Saak Transform

Purpose: Using content-adaptive method to further improve the accuracy of the MNIST test data.

Method: We divide the original images into several blocks and do hierarchy k-means to cluster the corresponding block in each image to construct hierarchy trees. In different trees, we select the nodes with low entropy and in these nodes, we do Saak transform. Then we train svm in each tree and get the probability of test images. After that, we compare the probability for each tree and select the prediction with the highest probability as the label prediction for test images.

Note: The time cost for running the codes should be controlled because what we want to do is beat CNN, not just running the codes without considering the time. If the time cost is long, the method is not that good.

II. Feedback and Interaction

■ ***Prof. Kuo's feedback***

- Analysis why the method did not work well.

■ ***Discussion***

III. Report

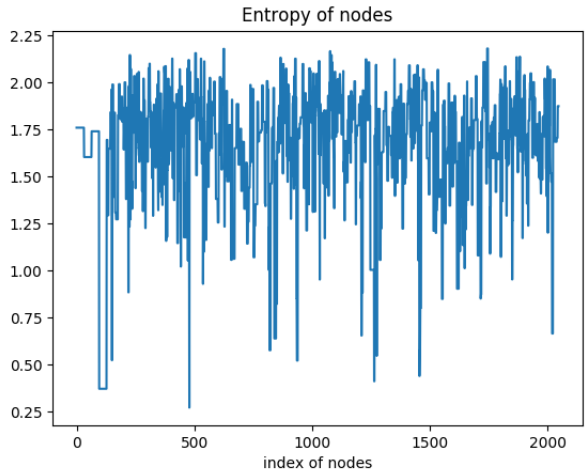
This week, I focused on analyzing why Saak transform cannot have very good results on CIFAR-10 and why it is very hard to further improve the performance of MNIST. Also I further compared the Saak transform and CNN.

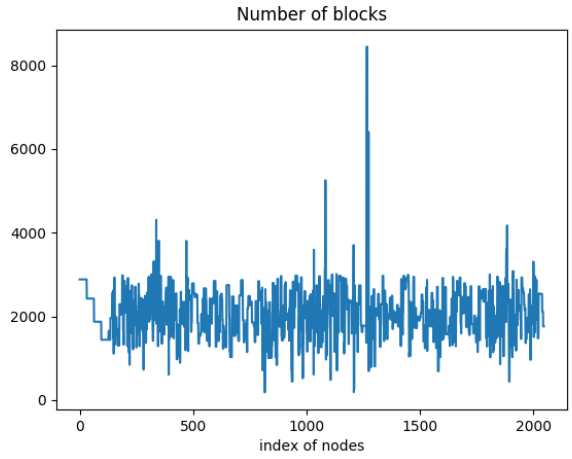
1. The method of more “balanced” tree

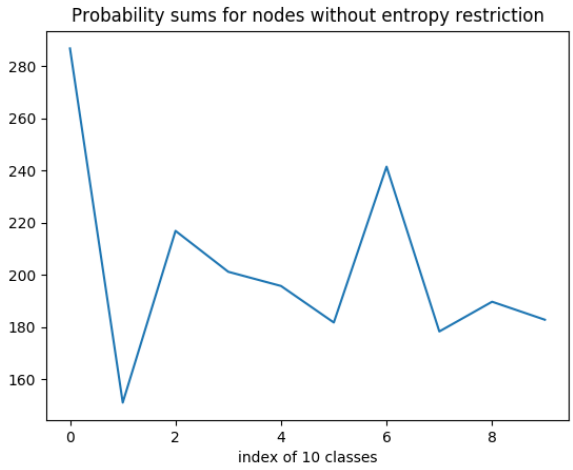
Professor Kuo proposed a new method to make the tree of hierarchy k-means clustering more balanced. First we get non-overlapping 16×16 blocks from the original 32×32 images. Thus we utilize hierarchy k-means to construct a shallow tree. After that, we get overlapping 16×16 blocks with stride 8 from 32×32 images again. Then we put the blocks into the constructed tree. If there are more blocks into one node, then we can construct more layers based on that node. If there are less blocks into one node, then we can constrict less layers based on that node. After that, we can get overlapping 16×16 with stride 4 from 32×32 images again and do the process again. By doing this, we can get more balanced tree and avoid getting the tree with one node with thousands of blocks and another node with dozens of blocks.
After we construct the nodes, we will select the nodes with low entropy value as the salient nodes. Then we need to calculate the conditional probability.

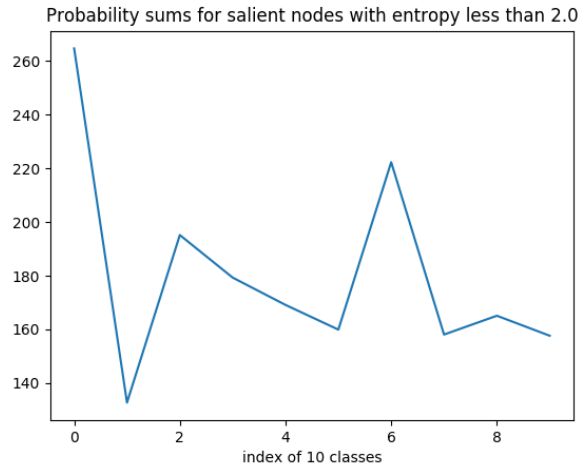
$$\sum_i P(O_j|S_i) = \sum_i \frac{P(O_j S_i)}{P(S_i)} = \sum_i \frac{P(S_i|O_j)P(O_j)}{P(S_i)}$$

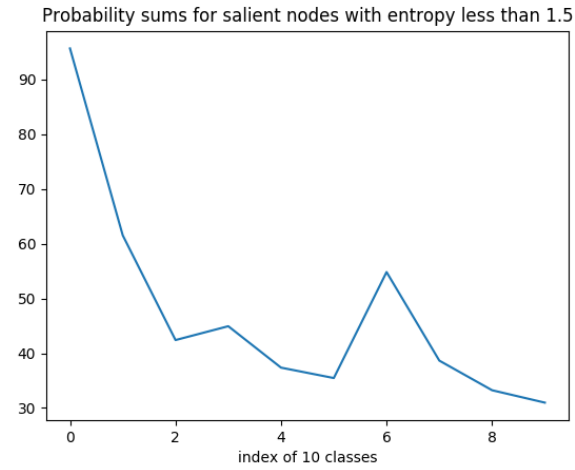
In the equation, S stands for the salient nodes and i represents the index of nodes. O stands for the class and j represents the index of the class.

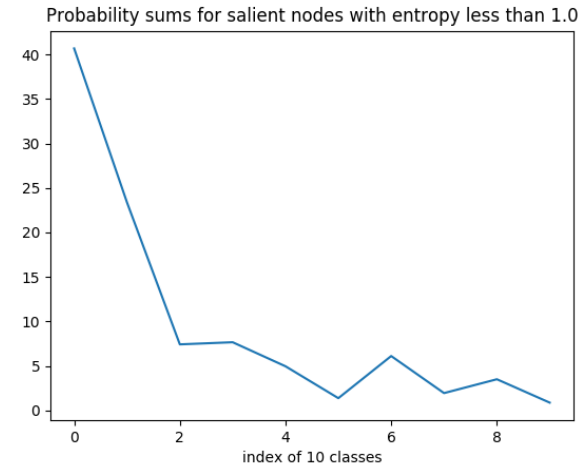












In the results above, I construct 2048 nodes in this tree. Based on different entropy threshold, the probability sums for each class is shown above. I do not think this method will work unless we can cluster the data from the same class into one group. If we cannot get very pure nodes, then the confusion classes will still be confusion classes. However, it seems impossible to put the data from the same class into one group without their labels. But we cannot utilize the test labels because in practice, we do not know the test labels and we need to predict them. I analyze this in detail in part 4.

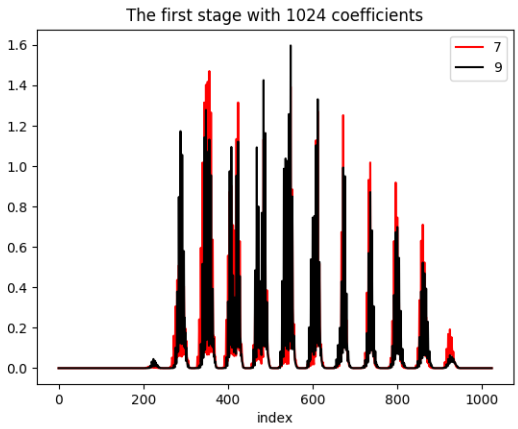
2. Some other methods

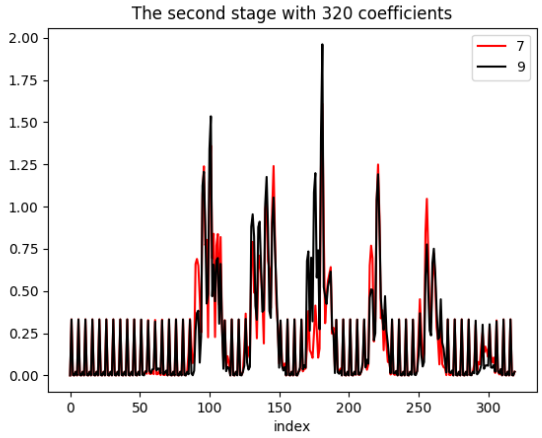
We also have proposed some other methods. For example, in the first stage, we do the 4×4 non-overlapping Saak transform; In the second stage, we do 4×4 overlapping Saak transform and finally in the last stage, we do Saak transform according to the blocks left. Thus we have 3 stages in this new Saak

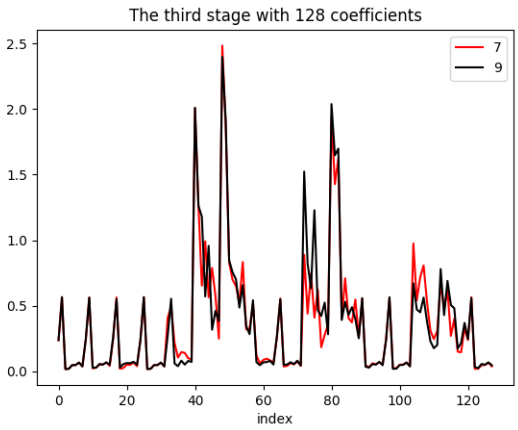
transform. But this did not work. I will explain this in part 4.

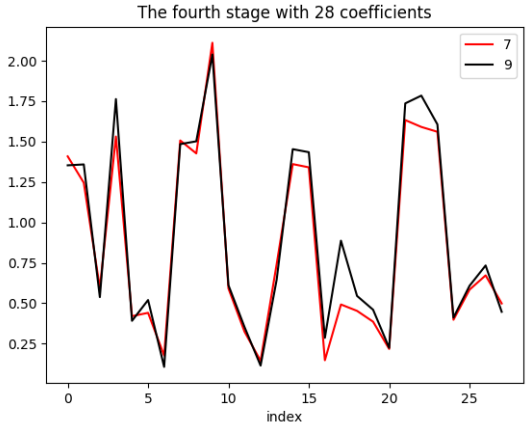
3. Analyzing the coefficients in each stage

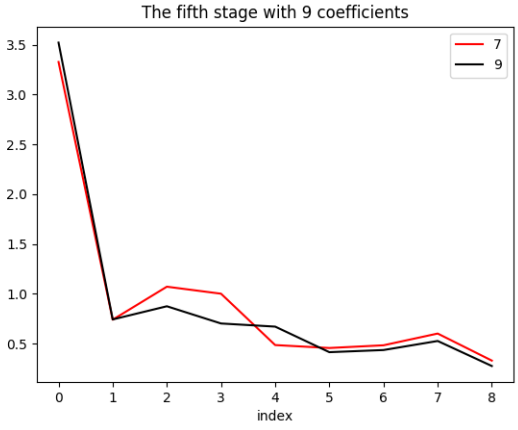
For example, “7” and “9”, how the coefficients will be in each stage? I averaged the coefficients in each stage for “7” and “9” respectively. I did not order the coefficients and just show the coefficients naturally because we need to compare the coefficients in the same place in the F-test and svm. By the way, the coefficients are absolute values.











In general, there are some differences between the coefficients of “7” and “9” even though the differences are very small. However, the figures shown above are the averaged results. For most of “7” and “9”, the prediction is correct. However, there are some special cases which cannot be predicted correctly just using statistics. Thus we need to consider these special cases separately from the rest. Almost all experimental results show that it is hard to further improve the performance of MNIST just using statistics unless we can utilize the test labels to cluster the test images and do Saak transform in each cluster. However, we cannot use the test labels because in practice, we need to predict them instead of use them. So we need to analyze the special cases separately because we do not the mechanism like back-propagation in CNN to correct the wrong labelled images. I will analyze the reason in detail below.

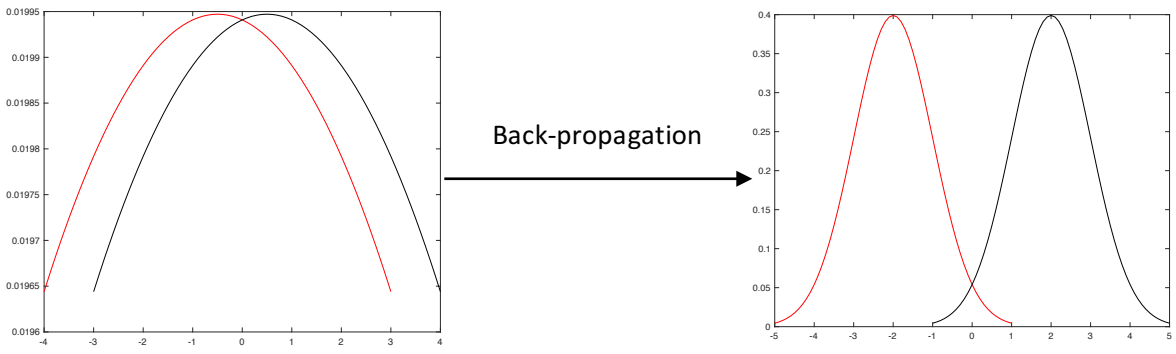
4. Analysis of Saak transform applied to MNIST and CIFAR-10

As we know, Saak transform is very robust because its performance is hard to fluctuate. Why? The reason is simple. First, the filters are fixed once the image set is fixed. As we know, Saak transform utilizes PCA to generate the filters of the images. Thus there is no much flexibility to change the filters. Second, the layers (stages) are fixed once the size of the images is fixed. It seems impossible to add or delete layers (stages) in Saak transform unless we do zero-padding to the images or we slice the images to smaller blocks. However, we can easily change the filters through back-propagation in CNN and also we can add or delete the layers in CNN as long as we want. So from this point of view, CNN is more flexible than Saak transform. And what’s more important, CNN can further improve its performance by changing the number of layers and the filters. But Saak transform cannot do the same things to further improve its performance. This is one very important reason we still cannot get very good results on Saak transform.

Now let me analyze two datasets: MNIST and CIFAR-10. For MNIST, now we can get the performance of 98.5%, which is very good. However, we want to get more than 99.5% accuracy for test images, and we have already tried so many methods, why it is so difficult to improve the performance of MNIST? We can analysis this from the train accuracy. The train accuracy of MNIST for the original Saak transform is not 100%. The MNIST dataset is very special, and the recognition of MNIST is the recognition of the shape of each image. **We need to recognize the digital number as a whole**. If we slice the images into different blocks, it is meaningless because each block can tell us almost nothing about the content of the image. Now let’s talk about the original Saak transform. Most of the images from the same class

have similar property and finally we can have 10 clusters for 10 classes and the intra distance of each class is relative small. As I mention above, the train accuracy of the MNIST dataset is not 100%. Thus even we label some train images their correct labels, but the Saak coefficients put them to other clusters instead their own clusters, thus the train accuracy is not 100%. These wrong labelled images are special cases. I did so many experiments, it seems that the best accuracy for test images using statistics is 98.5%. It is almost impossible to correct the wrong labelled images just using statistics. **We need to analyze these special cases separately from the rest**.

For CIFAR-10, because each image is different, for instance, all dog images, they have various shapes, colors, and also the images are taken from different angles. Thus each image is relatively independent and using Saak transform, it is very hard to have 10 clusters just like MNIST, and the distance of the Saak coefficients for each image is relatively large from each other. So each image is like its own cluster, thus no one will have confusion with others. Thus the train accuracy for CIFAR-10 is 100%. And test images also have the same property as the train images. Because every test image is different, the Saak coefficients for test images will "randomly" locate in the gap between the Saak coefficients for train images. Because there are no clusters in CIFAR-10, so the test images will be recognized as their nearest train labels. That's why the test accuracy is not that good for CIFAR-10. However, CNN can make the images from the same class cluster together which means that the intra distance of the same class will be reduced and the inter distance for different class will be increased through back-propagation. We can utilize the figures to show the process.



However, we cannot do this process because the filters of Saak transform are fixed. But the method mentioned by professor Kuo is meaningful for CIFAR-10 because CIFAR-10 contains objects from real world. We need to pick up the most important things from the objects to recognize the objects belonging to which classes. For example, for human being, we can easily recognize who is that guy by seeing the face of that person. If we remove the head of one person and left the body, can we recognize

who is that person? It is almost impossible unless there are some marks on the body. But if we only see the head of one person, we can easily recognize who is that person. So the head of one person is useful to recognize the person and the body of one person is useless to recognize the person. And almost all objects from the real world have the same property as the head and the body of one person. Thus slicing the images of the objects from the real world and focusing on the most important part is very meaningful and it is a correct direction to further improve the performance of Saak transform. And another reason for this method will be good for CIFAR-10 is that the performance of Saak transform on CIFAR-10 is not that good. Thus it is reasonable to utilize this method to further improve the performance of Saak transform on CIFAR-10. However, **we need to find a mechanism to separate the classes from each other just like the back-propagation in CNN**. Or it seems hard to make the results on CIFAR-10 very high.

However, for MNIST, I do not think this method will work and the results have already shown this. First, the performance of the Saak transform on MNIST is already very good. Thus next **we need to focus on the 150 special images which cannot be recognized correctly**. Second, we cannot slice the MNIST images to blocks because **MNIST images need to be recognize as a whole** which is totally different from the images taken from the real world like CIFAR-10, CIFAR-100 and ImageNet. What will the MNIST images be when we divide the images into different blocks? For some images, like “7”, and “9”, would become “1”, or like “3” would become “2” after we divide the images. But most of the blocks contain almost nothing because we cannot tell which classes they belong to because we cannot recognize the digital number by seeing only a part of the image. Thus we will have more confusion classes. For example, if the confusion classes are “7” and “9” before, after dividing the images, maybe “7” will be confused with “1” and “9” will be confused with “5” or “6”. That’s why the performance of Saak transform on MNIST after blocking the images cannot beat the original performance. Now let me talk about the clustering. If we cannot cluster the same class into the same group, the clustering will be meaningless and the result cannot be higher than original result. **Only clustering the same class into the same group can we reduce the intra distance in each class and enlarge the distance between different classes**. I have shown the results before when I utilize the train labels and test labels to cluster the train images and test images based on their classes, and that is the only case that the test accuracy is higher than 99.5%. However, we cannot utilize the test labels to cluster the test images because in practice, we do not know test labels and we need to predict them. But later on, no matter how I clustered the images, the images cannot be clustered based on their classes unless we utilized their labels. So dividing

the images and clustering the images seems not very effective on MNIST and we need to treat the MNIST images as a whole because there is no important part on MNIST.

All in all, if MNIST follows the method applied on CIFAR-10, or if CIFAR-10 follows the method applied on MNIST, we will go to the wrong direction because these two datasets are different in almost every aspect. So for CIFAR-10, we can follow the new method of hierarchy k-means. But for MNIST, we have to focus on the 150 wrong labelled images instead of statistics because using statistics, we have already gotten the accuracy of 98.5%. If we want to further improve the performance and get more than 99.5% accuracy for test images using statistics, it seems that clustering the test images based on their classes is the only method. But we cannot cluster the test images based on their classes without the test labels. Thus it is almost impossible to increase the test accuracy to more than 99.5% using statistics. So later on, we have to focus on the special 150 images and analyze them separately from the rest. And maybe changing the filters is also a good idea because the property of fixed filters is not flexible to so many images and later on we need to solve ImageNet. So Saak transform needs to be more flexible on the filters and number of layers (stages) so that it can adjust to more various data. The purpose of using PCA on Saak transform is that we want to losslessly recover the images and PCA can realize this because the filters generated by PCA are orthogonal. As we do not need to losslessly recover the images, maybe we can change the filters and be more flexible because there are more challenging datasets waiting for us.

5. One unsolved problem in Saak transform

I found a problem in Saak transform which is that if we do not do filter augmentation and just utilize the coefficients generated into the next stage, we can get the same accuracy result as that using filter augmentation. According to the paper “On data-driven Saak transform” and the paper “Understanding convolutional neural networks with a mathematical model”, the augmentation is to avoid the confusion between positive-positive and negative-negative and also between positive-negative and negative-positive. But I can get the same results between the Saak transform with augmentation and the Saak transform without augmentation.

I thought maybe the coefficients are all positive, but I checked that to find the coefficients are both positive and negative. Thus how to explain this phenomenon?

References

[1] C.-C. Jay Kuo, “Understanding convolutional neural networks with a mathematical model,” the Journal of Visual Communications and Image Representation, Vol. 41, pp. 406-413, November 2016.

[2] C.-C. Jay Kuo, “The CNN as guided multi-layer RECOS transform,” the IEEE Signal Processing Magazine, Vol. 34, No. 3, pp. 81-89, May 2017.

[3] C.-C. Jay Kuo and Yueru Chen, “On data-driven Saak transform,” arXiv preprint arXiv: 1710.04176 (2017).

IV. Plan for the next week (***, **, *: order of priority)

- Analysis the property of the new method mentioned above.

V. Milestone