III. Report

1. Clustering blocks with hierarchy k-means

In this experiment, first we utilized average-pooling to change 32×32 images to 16×16 images. Then we divided the 16×16 images into 8×8 blocks with step 4 (overlapping) and we could get 9 8×8 blocks. All these 9 blocks are local blocks, so we need one more global block. What I did was average-pooling 16×16 images to 8×8 images as the global block. Then we have 10 blocks for each image. We can construct trees for each block using hierarchy k-means and finally we will have 10 trees.

After we get 10 trees, we need to calculate the class entropy in each node. If the entropy of that node is low, we will utilize that node. And if the entropy of that node is high, we will not utilize that node. Thus in each tree, we have useful nodes and non-useful nodes. After that, we put all test images into all trees and if the test blocks are divided into the non-useful nodes, we will not use that test blocks. But if the test blocks are divided into the useful nodes, we will put these test blocks into svm and get the probability for each class. After that, we compare the probability of all trees and select the highest probability as the prediction.

In this experiment, the Saak transform is used for 8×8 blocks, thus we have 3 stages. Because the results using coefficients from all 3 stages are better than the results using only the last stage, so the results shown below are using coefficients from all 3 stages. I will not show the results using coefficients only from the last stage.

By the way, there is also one case we need to consider. What if the prediction says this test blocks is “0” but the the distribution of that node says that the number of “0” is close to 0? So we nee to consider the weights of the distribution. In the results shown below, if the results consider the “weights”, I will put “weighted” behind.

The results for 32 nodes:

Node entropy	**< 0.5**	**< 1.0**	**< 2.0**	**All nodes**
Test accuracy (weighted)	0.1967	0.6672	0.9181	0.9188
Test accuracy (non-weighted)	0.1963	0.6693	0.9289	0.9300

The results for 64 nodes:

Node entropy	< 0.5	< 1.0	< 2.0	All nodes
Test accuracy (weighted)	0.4733	0.7679	0.9202	0.9204
Test accuracy (non-weighted)	0.4733	0.7709	0.9283	0.9284

The results for 128 nodes:

Node entropy	< 0.5	< 1.0	< 1.5	All nodes
Test accuracy (weighted)	0.5970	0.8569		0.9223
Test accuracy (non-weighted)				0.9259

The results for 256 nodes:

Node entropy	< 0.5	< 1.0	< 1.5	All nodes
Test accuracy (weighted)				0.9246
Test accuracy (non-weighted)				

The results for 512 nodes:

Node entropy	< 0.5	< 1.0	< 1.5	All nodes
Test accuracy (weighted)				
Test accuracy (non-weighted)				0.9218

2. Results analysis

I did not have enough time to fill all the blanks in the tables above, but we can get some information from above.

When we utilized all nodes in each tree, we can get the best results. It is easy to understand. If we did not use some nodes, maybe there is some important information in that node because we decided which nodes to be used on the entropy of the nodes. And there is some uncertainty if we just depend on the entropy of nodes. In addition, we combined the test results from all trees together and selected the best prediction. However, there is one case that after we combine the test prediction from all trees, the results do not contain all test images. Thus we will have no information on the test images which are not contained in all trees.

However, if we utilized all nodes in each tree, even though there are some “bad” nodes in each tree, but we will compare the results from all 10 trees and get the best one as the prediction. So if some nodes are “bad”, they will not be selected as the prediction in the final comparison. Thus there is no need to select the nodes in each tree and the result that using all nodes in each tree can make us get the best result is reasonable.

Moreover, the results without weighted probability seem better than the results with weighted probability, so we do not need to utilized weights on the test probability.

Also, it seems that adding more nodes in each tree cannot improve the test accuracy too much. I think the reason is that we utilize blocks to cluster. The blocks are only the partials of the original images, we will get more confusion images using the blocks than using the original images to do the Saak transform. In the original Saak transform, the distribution of the train images is spread because the images contain all 10 classes. However, using k-means to cluster the blocks, maybe in the final nodes, only blocks from confusion images are in the nodes. Thus maybe this will generate more confusion because the blocks are so similar and it is even harder to separate the confusion images using Saak transform under this scenario. What’s more, even though we utilize the hierarchy k-means algorithm, the tree is still unbalanced, because some nodes may contain thousands of blocks, while some nodes may contain hundreds of blocks or even less.

By the way, when the number of the nodes is increasing, the time cost is also increasing rapidly. Time control is also a key point we need to consider.

3. Improvement of Clustering blocks with hierarchy k-means

I came out a way to do the Saak transform after we cluster the blocks and construct 10 trees. We can do Saak transform in each node and get the coefficients for each block. Then we combine the coefficients from the same image from all trees together. For example, it one block from one image can generate 100 coefficients, because we have 10 trees, thus we can have 1000 coefficients for each image. After that we utilize the all train coefficients to train the svm and predict the test images.

Number of nodes	**32**	**64**
Test accuracy	0.9524	0.9444

4. Future work

The results above are already good enough because the test accuracy is more than 90%. However, the the results for the original Saak transform is 98.5% and we want the result to be higher than 99.5%, thus the results are not good enough when compared to those.

Professor Kuo proposed a new method to make the tree of hierarchy k-means clustering more balanced. First we get non-overlapping 8×8 blocks from 16×16 image which is got by average pooling. Thus we utilize hierarchy k-means to construct a shallow tree. After that, we get overlapping 8×8 blocks with stride 4 from 16×16 image 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 8×8 with stride 2 from 16×16 image again and do the process again. By doing this, we can get more balanced tree and avoid getting the tree with one node has thousands of blocks and another node has dozens of blocks. I will try this method in future work.

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