EE559 HW6

Motivation

Green learning is an attempt to get rid of heavy data, heavy training, and heavy model. In Kuo's research, he managed to design subspace approximation via augmented kernels (Saak) and subspace approximation via adjusted bias (Saab) transforms to get rid of activation function in convolutional module, and one-pass feedforward FC layer to get rid of backpropogation. Based on Saab and the thought of successive subspace learning (SSL), he further developed PixelHop and PixelHop++, which is a complete and efficient model for image processing. This project will focus on explanation of the working principles and some concepts of Green learning, and implement them to see their performance.

Procedure

Problem 1 are all explanation to Green Learning, all of them are put in Discussion part.

Problem 2 mainly focus on PixelHop++. We first implement Pixelhop++ model with certain parameter, and then tune TH1 to see the change of accuracy and model size. Next we compare performance of PixelHop and Pixelhop++, whose main difference is Saab and c/w Saab. Finally we do error analysis to see why our model makes mistakes and potential way to improve it. All the experiments are conducted on both MNIST and FashionMNIST dataset.

Result

Building PixelHop++ Model

The train of Module 1 cost around 5 min with reduced training set, Module 2 cost around 12 min (get_feat function takes a lot of time), Module 3 on Hop3 feature cost 2 min.

For MNIST:

The test accuracy (10000 training data) is 97.01%, and the test accuracy is 91.68%

Total number of parameters in 3 Hop units is 6150

For FashionMNIST:

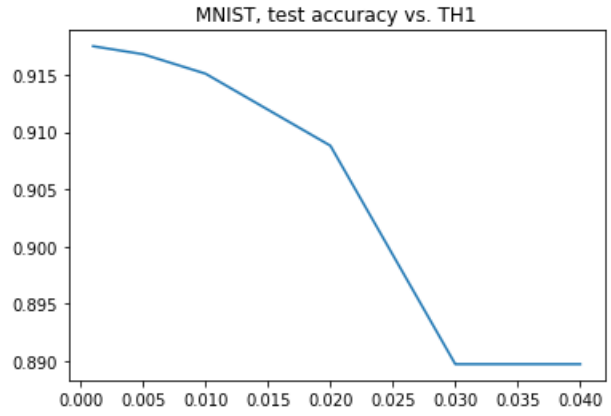
train accuracy 88.56% test accuracy 78.37%, total number of parameter 3300

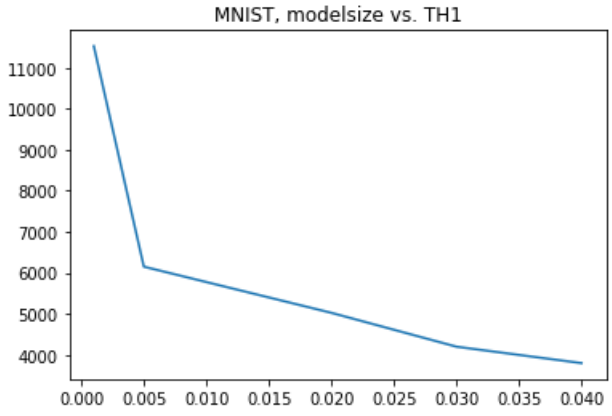
Configuration:

Spatial Neighborhood size in all PixelHop++ units	5x5
Stride	1
Max-pooling	(2x2) -to- (1x1)
Energy threshold for intermediate nodes (*TH1*)	0.005
Energy threshold for discarded nodes (*TH2*)	0.001
Classifier	XGBoost
Number of estimators in classifier	100

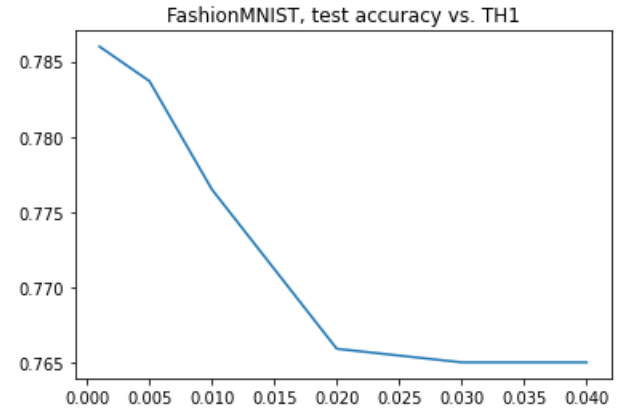
With same TH2, the performance based on TH1 is shown

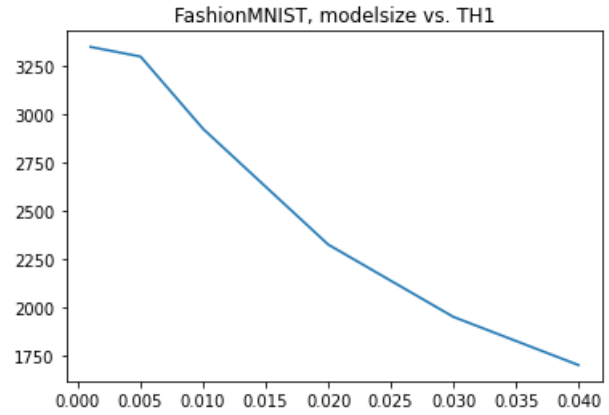
For MNIST dataset:





For FashionMNIST dataset:





~~

raw data

TH1=[0.001,0.005,0.01,0.02,0.03,0.04]

MNIST:

train accuracy [0.9704, 0.9701, 0.9677, 0.9613, 0.9504, 0.9504] test accuracy [0.9175, 0.9168, 0.9151, 0.9088, 0.8897, 0.8897] model size [11525, 6150, 5775, 5025, 4200, 3800]

FashionMNIST:

train accuracy [0.8873, 0.8856, 0.8752, 0.8686, 0.8529, 0.8529] test accuracy [0.786, 0.7837, 0.7765, 0.7659, 0.765, 0.765] model size [3350, 3300, 2925, 2325, 1950, 1700]

~~

Comparison between PixelHop and PixelHop++

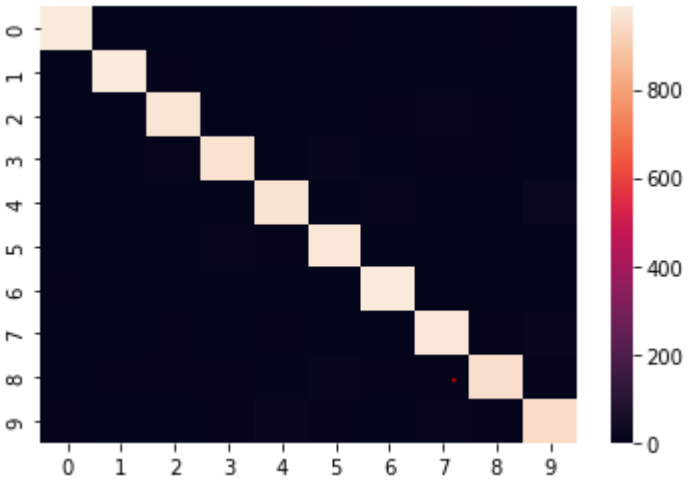
I use the same configuration as problem2(a) on PixelHop (c/w Saab -> Saab).

On MNIST dataset, PixelHop achieves train accuracy 95.93%, test accuracy 90.59% with total number of parameter 88825. (By comparison, PixelHop++ is 97.01%, 91.68%, 6150)

On FashionMNIST, it achieves train accuracy 87.96%, test accuracy 77.93%, with total number of parameter 24675. (By comparison, PixelHop++ is 88.56%, 78.37%, 3300)

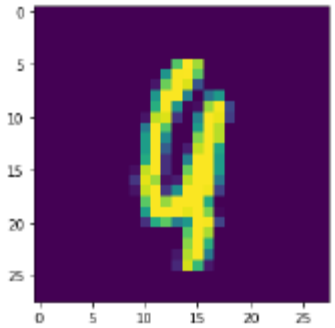
Error analysis

For MNIST

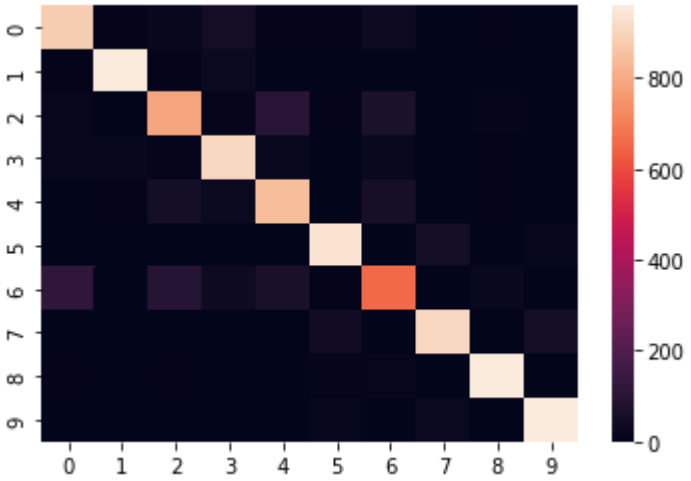


The class with lowest error rate is class '0' and "1". The class with highest error rate is class "9"

The accuracy is quite high. For the most confused pair "9" and '4', for example, some handwritten are really hard to confirm it's one or the other. I picked a example from the dataset and the ground truth of it is "9"



For FashionMNIST



The class of "Bag" has the lowest error rate. The class "Shirt" has the highest error rate.

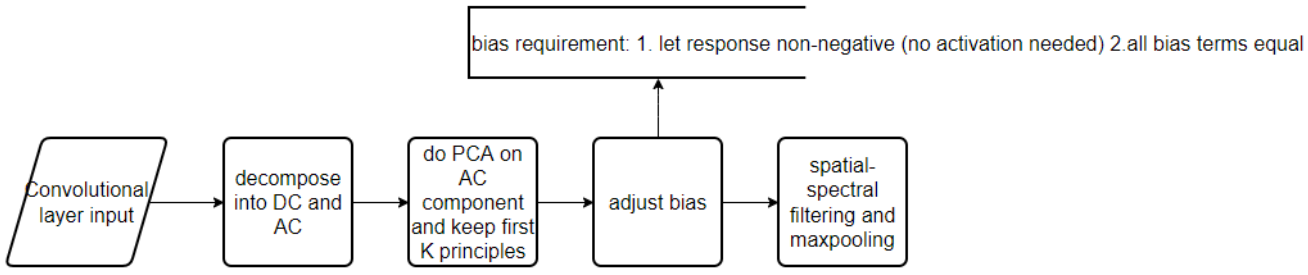
The most common confusion pairs, for example is group 0 and 6, which corresponds to T-shirt and shirt. They are easy to be confused just because they are really similar in reality. Similar is the other pair like pullover and shirt. Human will still be confused sometimes.

Discussion

Problem 1. Origin of Green Learning

Saab transform

Subspace approximation with adjusted bias's major purpose is to get rid of activation function, and reduce dimension in a lightweight design in the convolution module. The main task is to find proper anchor vector a_k and bias b_k for $y_k = {a_k}^T x + b + k$. First it decompose input vector space to DC and AC, by projecting the input to DC anchor subspace and AC anchor subspace. All manipulation are done on AC component. Next do PCA on AC part and keep the first K principle components as features. Then we compute the bias. It is expected to overcome sign confusion problem in Saak by making y_k greater than 0. Also all bias terms are equal in order to make the design simpler. Finally do maxpooling. The flow chart below summarize the Saab procedure.



FF-CNN and BP-CNN

Their similarity is, they are both data driven model, which requires certain amount of training. Their operating principle is similar, the convolutional layers extract features and reduce dimension, the FC layers do classification or regression.

The major difference is FF-CNN do not have back propogation part. While BP has to iterate over and over, FF-CNN can come straight to the point, and the whole process is interpretable. Also FF-CNN have less parameters (e.g. equal biases), which makes the model smaller. Thus, the model is simpler and easier to train.

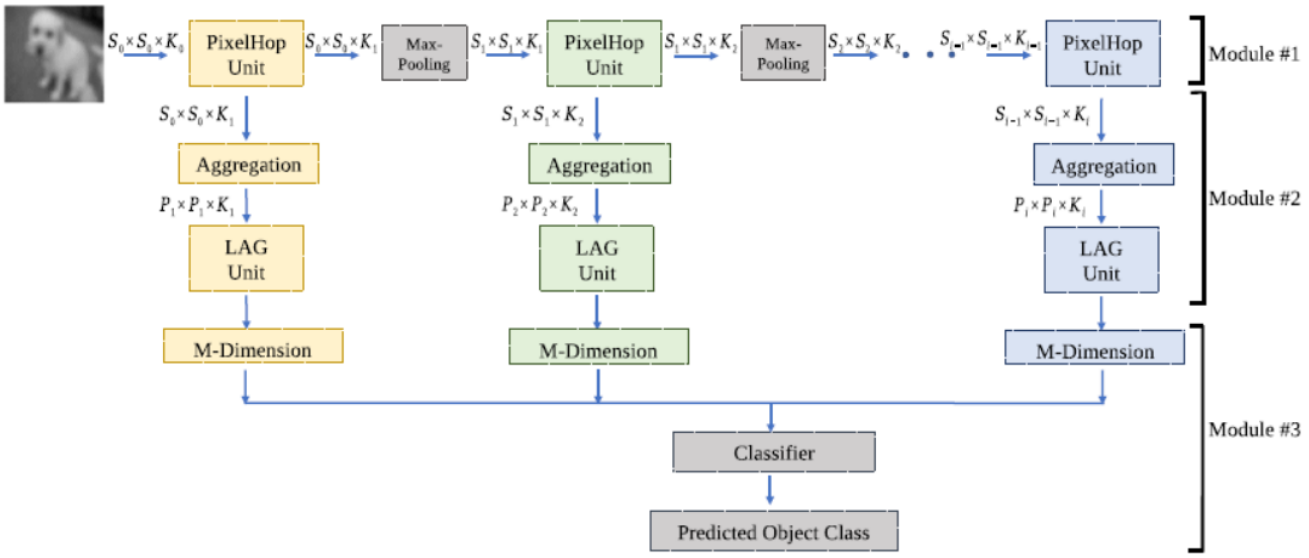
PixelHop and PixelHop++

PixelHop is a image-based object classification method based on SSL. SSL (successive subspace learning) consists of four major components, and I'll explain them with the example of PixelHop design.

1. successive near-to-far neighborhood expansion. The process start from a very localized region, and gradually expand to further region. This process use consecutive Pixelhop unit, which can see larger and larger area, and the attributes can propogate to latter units (that's why it's called Hop).
2. Unsupervised dimension reduction. To reduce the increasing dimension in the PixelHop unit, maxpooling and Saab transforms are used.
3. supervised dimension reduction through LAG (label-assisted regression). It is a similar design of FC layers in FF-CNN. It generate some pesudo labels and, together with actual label, to do least square regression. A feature vector will be output at the end of the FC layers.
4. feature concatenation and decision making. Every PixelHop unit will generate a feature vector. We concatenate them into a matrix and use it as the feature representation of the whole image.

SSL and DL are similar in terms of growing receptive fields, trading spatial for spectral dimension, and spatial pooling according to Chen(2020).

Meanwhile, they are different in many ways. The most obvious aspects I can think of are: SSL has one-pass feedforward design, while DL has to do BP. SSL has much more flexible architecture. It can expand or shrink by just adding or subtracting units. SSL model is white box and is interpretable while DL is black box. SSL also performs better than DL when there's only small number of training data. There're more differences in Chen's article.



Block diagram of PixelHop method

> Chen, Yueru, and C-C. Jay Kuo. "Pixelhop: A successive subspace learning (ssl) method for object recognition."

The first module is for near to far neighborhood expansion and unsupervised dimension reduction. The main component is PixelHop unit. It captures information from local to global, and use Saab transform to prevent dimension from explosion. MaxPooling is used to reduce spatial redundancy in attribute representation.

The second module is Aggregation and LAG. While maxpooling is used to reduce dimension that send to next Pixelhop unit, for current Pixelhop unit, multiple aggregation method like maxpooling, meanpooling, minpooling are used in order to extract various feature. LAG has just been talked about above.

The third module is feature concatenation and classification. It use all the features from every Pixelhop unit and apply a regular ML procedure to do classification.

For PixelHop++, it has similar neighborhood construction and subspace approximation steps, that is taking the union of a center pixel and its eight nearest neighborhood pixels, and then do Saab transform. The difference of PixelHop and PixelHop++ in these steps is the input tensors dimension. PixelHop++ 's spatial neighbourhood size is constant (5 * 5), while PixelHop is 25 * K_i. This is based on the observation that Saab coefficients has weak spectral correlation, so the joint spatial-spectral input tensor 25 * K_i can be decomposed to 5 * 5

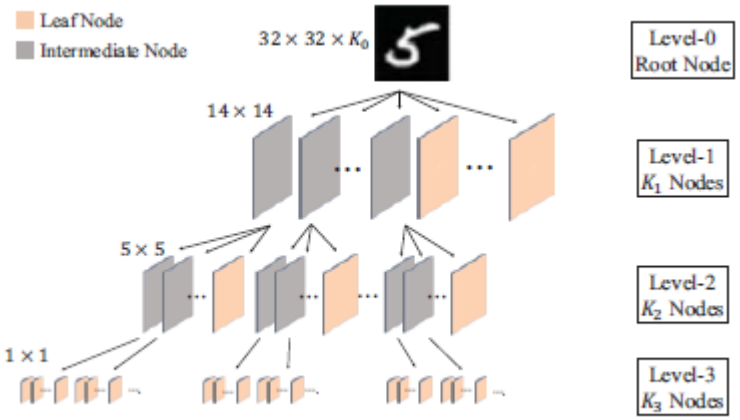
Saab transform and the channel-wise (c/w) Saab transform are almost the same, except that cw Saab transform works on colored images and treats each channel separately (do Saab on each channel). That adjustment makes the input tensor size much smaller.

Problem 2

PixelHop++

The input size for the first hop unit is supposed to be 32 * 32, but MNIST and FashionMNIST are 28 * 28. So there's a implict requirement that the first PixelHop unit should have proper pad. In our case, pad=2.

TH1 is for passing the hop nodes to next hot nodes, and TH2 is for discarding those nodes with least energy. In the figure below, if the energy did not reach TH1, then it stops propogate to next hop unit and become a leaf node. In this part, we are asked to fix TH2 and tune TH1. We can see as TH1 increase, the accuracy decrease, but just to slight extend. While the reduction of model size is significant. That means Pixelhop++ is very economic and can have good performance even with small size.



> Chen, Yueru, et al. "Pixelhop++: A small successive-subspace-learning-based (ssl-based) model for image classification." *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020.

Comparison between PixelHop and PixelHop++

We can see Pixelhop++ slightly outperform Pixelhop. But meanwhile there're a lot of students get the inverse result. Maybe the two datasets are too simple so the performance difference is not large. Slight chances cause by choice of hyperparameter may easily reverse the outcome.

What is remarkable is that Pixelhop++ has much smaller model size (about 10x smaller) due to the decompose of input tensor in each unit. This means PixelHop++ is much more efficient.

Error analysis

The assignment requires to use all 60000 training data, but it crash during p2.fit() due to RAM (12G). So I return to 10000 reduced set.

We have seen in Result section that the errors mainly comes from some confusing data, and other models, like LeNet-5, has very similar errors. To improve accuracy, suppose we do not modify the dataset, we can possibly adjust the thresholds and thus more features will be extracted. For those similar classes, maybe features with low energy will help. But definitely, it will increase model size and training time, so it's a tradeoff.

Reference

- Chen, Yueru, and C-C. Jay Kuo. "Pixelhop: A successive subspace learning (ssl) method for object recognition." *Journal of Visual Communication and Image Representation* 70 (2020): 102749.
- Chen, Yueru, et al. "Pixelhop++: A small successive-subspace-learning-based (ssl-based) model for image classification." *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020.
- Kuo, C-C. Jay, et al. "Interpretable convolutional neural networks via feedforward design." *Journal of Visual Communication and Image Representation* 60 (2019): 346-359.