Samwoo Seong
7953-6137-66
samwoose@usc.edu

EE569 Homework #6
April 25, 2020

1.

1.1 Abstract and Motivation

In traditional CNN-based structure, filters are determined by back propagation optimization. However, if we understand a role of non-linear activation function employed between convolutional layer, we can derive a transform method and construct a feed forward structure meaning parameters of filters at each layer are determined by statistics of input images without back propagation step. In this practice, we will explore Saab transform and feed forward CNN with the transform. Then, we will compare FF-CNN with the traditional CNN. [1]

1.2 Discussion

(a)

(a-1)

Saab transform is one of many ways to get a representation of an input image in subspace spanned by representation vectors. In Saab transform, it makes a use of PCA to find anchor vectors (i.e., representation vectors of a subspace) that construct a subspace of input $\mathbf{x}$ and transforms it to a corresponding representation in the subspace. Since PCA is statistic-based method, anchor vectors are input data driven.

Moreover, it also utilizes bias term $\mathbf{b}$. At the previous research, Kuo studied role of non-linear activation function and its contribution to the performance of BP CNN. [2] It turns out non-linear activation function resolves sign confusion of output of layer when multiple convolution layers are present in CNN structure. By selecting bias term properly based on the rule given in the paper [1], we can avoid sign confusion without non-linear activation function.

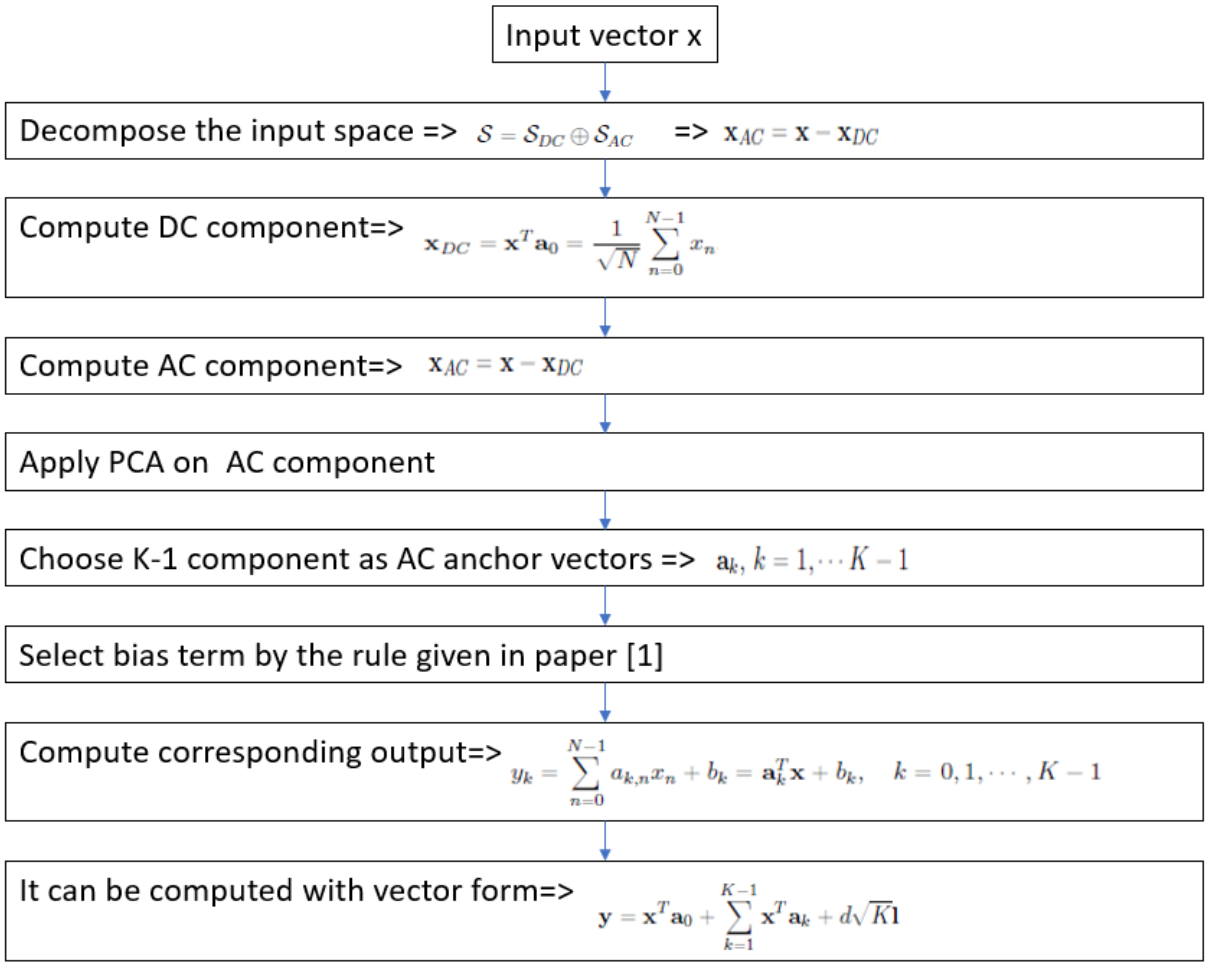


Figure 1. Saab Transform Flow Chart.

(a-2)

1-1 Principle of BP-CNN

Fundamental principle used in BP-CNN is to build a structure that can capture patterns of input images by finding filter parameters that cause low values of a defined cost function at each layer. To achieve this end to end optimization process is used. [1]

1-2 Principle of FF-CNN

A structure and goal of FF-CNN are similar to BP-CNN. However, it uses different approach called Saab transform and LSR to find learning parameters at each layer. Those two are data-based techniques which are different from optimization techniques applied in BP-CNN. [1]

2-1 Mathematical tool of BP-CNN

Since BP-CNN focuses on lowering value of cost function at output of each layer, it uses optimization techniques such as Stochastic Gradient Descent. [1]

2-2 Mathematical tool of FF-CNN
Since FF-CNN relies on statistic data from input image, statistic base techniques are used such as Principle Component Analysis, and K mean clustering. [1]

3-1 Interpretability of BP-CNN
The deeper layers go, the harder BP-CNN to be explained because of optimization process which is not easy to be tracked by hand-written equation and strongly connected layers. Thus, it is hard to understand what is really going on at each layer. [1]

3-2 Interpretability of FF-CNN
In FF-CNN, every equation used at each layer can be tracked by hand. Therefore, it provides high interpretability of each layer. [1]

4-1 Modularity of BP-CNN
Since all layers are highly connected to each other, it is impossible to remove and add some layers from BP-CNN structure. In that case, we need to redesign a whole structure. [1]

4-2 Modularity of FF-CNN
Since role of each layer is easily interpretable, it is easy to decouple a certain layer and replace it with another layer. For example, FC layer in FF-CNN can be replaced with traditional machine learning models such as Random Forest, and Support Vector Machine because role of FC layer is to predict labels and role of previous layers is to extract features and representations of input images. [1]

5-1 Robustness of BP-CNN
During recent years, researchers have found that once a structure of CNN is known, it can be easily fooled by adding a little bit of certain noise to input images. It is also known as adversarial attacks. Thus, robustness of BP-CNN is very weak. [1]

5-2 Robustness of FF-CNN
Since FF-CNN has followed exactly the same structure as BP-CNN, it also has low robustness. [1]

6-1 Training complexity of BP-CNN
Typically, when non-convex optimization process is involved, training complexity can be increased drastically because there are many local opima and to find possible global optimum, thousands of iterations are inevitable. Therefore, it has high training complexity. [1]

6-2 Training complexity of FF-CNN
Interestingly, covariance matrix used in PCA method tends to converge with small amount of input images. It means we can train a FF-CNN model without losing performance that a model trained by a large number of data provides. [1]

7-1 Architecture of BP-CNN
BP-CNN is end-to-end optimization and layers cannot be decoupled from each other. Therefore, we need to design a whole structure for new task. [1]

7-2 Architecture of FF-CNN
Since FF-CNN has high modularity (thanks to high interpretability), design of its architecture is relatively more flexible compared to architecture of BP-CNN. It means some layers can be replaced by some other layers depending on tasks. [1]

8-1 Generalizability of BP-CNN
It is hard for BP-CNN to be generalized because of its tight connections among layers. Therefore, we need to design several BP-CNN for multiple tasks. [1]

8-2 Generalizability of FF-CNN
Because of its high modularity, we can simply choose proper methods for different tasks and connect properly. [1]

9-1 Performance of BP-CNN
Researchers have gotten magnificent performances with BP-CNNs at many computer vision applications. [1]

9-2 Performance of FF-CNN
FF-CNN has not shown a lot of state-of-art performance since it has not been studied for long time of period. [1]

(b)
(b-1)
Successive Sub-space Learning also known as SSL contains 4 main parts. [3]

1. Build receptive fields with different ranges by successively constructing neighborhood.
2. Apply a certain transform (e.g., Saab transform in Pixelhop method, and channel wise Saab transform in Pixelhop++ method) on the receptive fields to find their subspace. This subspace will represent original space and provide discriminant power for a trained model to distinguish different classes. In Saab and c/w Saab transform, discriminant power is created based on statistics of input images (i.e., covariance matrix). Additionally, it has an ability to reduce dimension of feature space. These properties will be useful in machine learning part.
3. Create more discriminant ability in the same class with label assisted regression (LAG). It also provides nice dimension reduction of feature space.
4. With features generated through 1,2, and 3 parts, perform classification task (i.e., train a prediction model and predict labels of test input data)

Each part will be further explained at the following discussion.

Deep learning also known as DL and SSL have something in common. 1. Both methods build multiple receptive fields to obtain various information from input images. 2. Both methods pay attention to spatial and spectral information of the input images. 3. Both methods use spatial pooling to build different receptive fields and it is also for dimension reduction purpose as well. [3] However, SSL has many aspects that are different from what DL has. [3]

1. DL has a lot of filter parameters that have to be learned while SSL has relatively small number of parameters. For DL to perform well it needs that many parameters because the parameters provide an ability to handle rich patterns in images. On the other hand, SSL pays attention to statistics of the images. Therefore, a number of parameters is fixed.
2. Since DL has end to end optimization structure, when new data set is introduced, a whole system needs to be re-designed. For SSL, it can simply design Saab transform based on the new data set.
3. Similarly modifying architecture of DL is almost impossible since all layers are strongly coupled. However, several units can be added or deleted depending on given tasks in SSL.
4. Like many optimization-based models, it is hard to mathematically keep track of every single layer. On the other hand, every step can be mathematically traced which provide high interpretability.
5. Since SSL has a nice property that covariance matrix converges quickly as a number of input images increases, it can use a small number of samples without losing performance in Saab/channel wise Saab transform part. However, In DL, every layer needs all data samples to perform well.
6. Dimension of feature space reduction is achieved by Saab or c/w Saab transform in SSL. In DL, number of filters will determine dimension of feature space.
7. A whole DL needs to be designed for each task meaning output of each layer only can be used for a specific task. However, SSL can multiple extract features depending on different tasks and use existing rest of modules.
8. It is hard for one DL structure to handle multiple tasks because DL is designed for one specific purpose. However, SSL can easily manage several tasks by adding proper modules at right places.
9. SSL proceeds forward once and finishes training prediction model. DL needs tons of iteration to optimize filter parameters.
10. Since SSL doesn't need labels at part 1 and 2 of its structure, it shows less performance drop as a size of train data set decreases while DL shows relatively significant performance drop.
11. As we mentioned earlier, DL model is vulnerable to adversarial attacks, however SSL can filter out added small changes of incoming images which are principle of adversarial attacks because it uses statistics of the images.

(b-2)

As a different point of view, we can divide SSL framework into 3 modules for both Pixelhop and Pixelhop++. [3], [4]

<Pixelhop>

1. Function of Module1 in Pixelhop: At this module, it constructs neighborhood of target pixel with a certain window size. It allows us to capture the spatial and spectral

information of that area with Saab transform (i.e., with neighborhood construction and the transform we can compute attribute of surrounding area of a target pixel). Spatial pooling is performed successively in this module. This is for creating different receptive fields, so we can find diverse patterns from input images.

2. Function of Module2 in Pixelhop: Once we get spatial and spectral information from the module 1, we want to capture rich features from this. To do so, we can apply mean, maximum, minimum aggregation method. After the aggregation, we can go further to increase discriminant power of features amongst images of the same class by label assisted regression.
3. Function of Module3 in Pixelhop: After module1 and module2 we end up with rich features of input images. By concatenating these features properly, we can form a train or test data set which is suitable for machine learning models. At this module, we simply train a machine learning model with train data set and predict labels of test data set, and check its performance.

<Pixelhop++>

1. Function of Module1 in Pixelhop++: Main function of this module is similar to module1 in Pixelhop. Difference is that it uses channel wise Saab transform. It will be explained more at following discussion.
2. Function of Module2 in Pixelhop++: Main flow of this module is quite similar to module2 in Pixelhop. Only difference is that it uses cross entropy to select features instead of multiple aggregations.
3. Function of Module3 in Pixelhop++: Role of this module is exactly the sample as module3 in Pixelhop.

(b-3)

Both Pixelhop and Pixelhop++ methods build neighborhood of target pixel and approximate original space to subspace. First, we need to construct neighborhood of a target pixel. Purpose of neighborhood construction is to compute attribute of surrounding areas of a target pixel. To do so, we can follow the next steps. [3]

Step1. Choose window size and target function. Here red square is a target pixel and blue squares are neighbor pixels selected by 5x5 size of window.

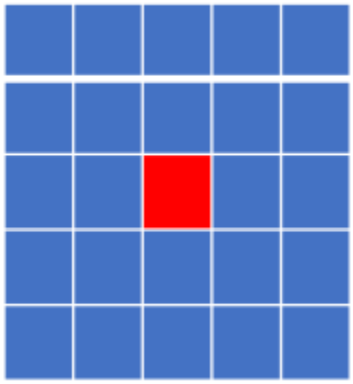


Figure 2. Selected region.

Step2. Construct neighborhood using the selected region. => (K x 5 x 5) dimension

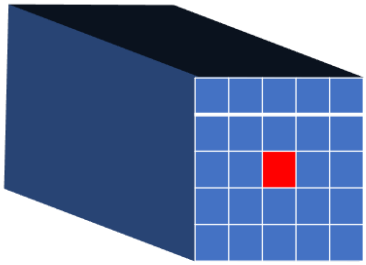


Figure 3. Neighborhood structure.

Step3. Reshape it (i.e. flatten the neighborhood) => ((25 x K) x 1) dimension

Figure 4. Reshaped neighborhood structure.

Second, we need to apply a certain transform to reduce the dimension of reshaped neighborhood because its dimension increases rapidly. At this stage, Saab transform is employed in Pixelhop, while c/w Saab transform is employed in Pixelhop++. [3], [4]

<Pixelhop>

Step4 in Pixelhop. Apply Saab transform and compute its representation projected to its subspace.

Figure 5. Dimension reduced reshaped neighborhood structure after Saab transform.

<Pixelhop++>

It turns out correlation K channels are very low meaning they can be separated without losing generality. Since we can perform Saab transform channel wise, we will be able to not only save computation source but also introduce tree decomposition method for feature selection at this stage. This method selects intermediate nodes (i.e. channels that will go to the next units) based on their energy level. In our programming threshold 1 is used for choosing intermediate nodes and threshold 2 is used for selecting leaf nodes. It indicates that we have more control over reducing feature dimension in Pixelhop++.

(b-4)
LAG unit plays two major roles. First role is that it provides ability to reduce the dimension of feature space that have grown rapidly through module 1. Second role is to capture more details in the same class by performing K means clustering among the same class. So that the trained system will be able to predict class more accurately. For instance, when images are labeled as a cat, they could capture different breeds. By LAG unit, this situation could be handled better than a trained model without LAG unit. LAG procedure can be summarized as follows.[3], [4]

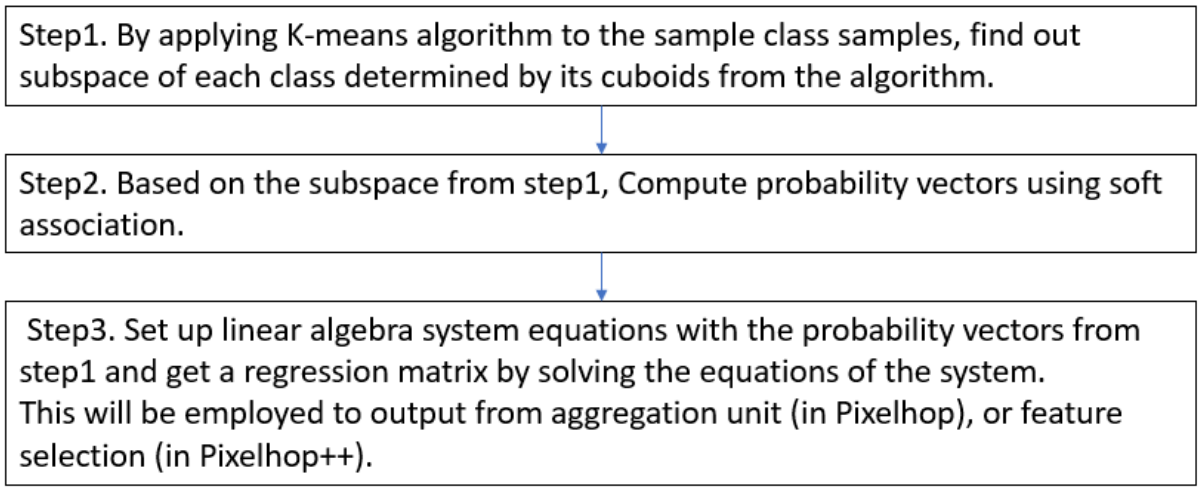


Figure 6. LAG unit procedure.

2.1 Abstract and Motivation
Once we have found that it is possible to extract meaningful feature vectors with multiple layers of statistic-based transform [1], we no longer have to follow the structure of the traditional CNN. Instead, we can build a new structure that extracts data driven features using Saab transform. After the feature extraction, we can tune those features further to perform classification task well such as feature selection, feature reduction, and cross validation which are widely used in traditional machine learning field. Furthermore, this type of approach has provided mathematical transparency of the training structure while traditional CNN hasn't.

2.2 Approach and Procedures
(a)
To build a PixelHop++ model, we need to carefully connect each unit such as neighborhood construction, channel wise Saab transform, max-pooling, feature selection, LAG unit, classifier, and predicted object class. Since we have discussed roles of all units in depth above, we can jump right into procedure. The procedure is illustrated nicely by Kuo et al. [4]

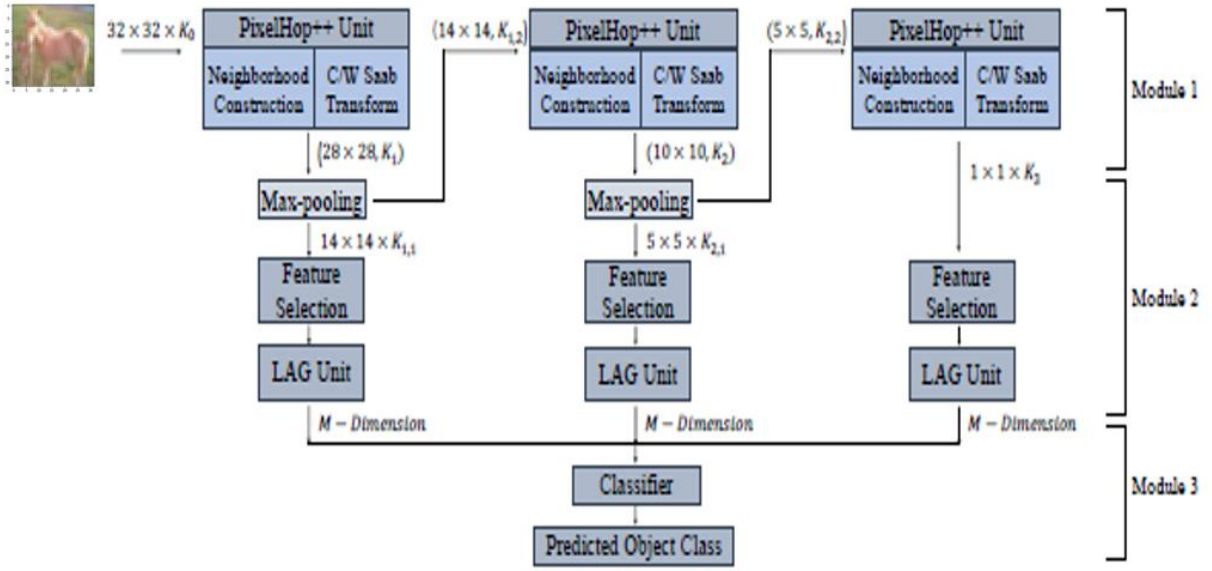


Figure 7. Procedure of the PixelHop++ model [4]

Step1. Calcute attributes of input images by pixelhop2.py (Module1)
Number of train images : 6000
Number of test images: 10000

Step2. Apply maxpooling onto the attributes accordingly (Module2)
Numbers of train images : 20000, 12500, 6250, 3125, 1562
Number of test images: 10000

Step3. Select features that give low value of cross entropy (Module2)
Numbers of train images : 20000, 12500, 6250, 3125, 1562
Number of test images: 10000

Step4. Apply LAG to reduce feature dimension and gain more discriminant power (Module2)
Numbers of train images : 20000, 12500, 6250, 3125, 1562
Number of test images: 10000

Step5. Concatenate features from all 3 units and form training data or test data set depending on input images (Module3)
Numbers of train images : 20000, 12500, 6250, 3125, 1562
Number of test images: 10000

Step6. Train a random forest model with training data and predict labels of training data and test data. (Module3)
Numbers of train images : 20000, 12500, 6250, 3125, 1562
Number of test images: 10000

Figure 8. Flowchart of the PixelHop++ model implementation

Spatial Neighborhood size in all PixelHop++ units	5x5
Stride	1
Max-pooling	(2x2) -to- (1x1)
Energy threshold (T)	0.001
Number of selected features (N_S)	1000
α in LAG units	10
Number of centroids per class in LAG units	5
Classifier	Random Forest (recommended)

Figure 9. Chosen hyper parameters

	Used provided python scripts
Module1	pixelhop2.py
Module2	cross_entropy.py, lag.py

Table 1. Used provided python scripts for each module.

(b)
Compute a confusion matrix based on true labels of test set and predicted labels of test set.

2.3 Experimental Results (Partially open source is used)
#NOTE
Since I was experiencing severe memory issues during pixelhop2.fit() and pixelhop2.transform(), maximum number of data samples are 6000 train images, and 20000 train images respectively. In the future work, I better use cloud web service for bigger number of data samples
(a)
(a-1)
<Model size>
Module1
=>1st PixelHop++ unit has 5x5 size of window, 3 channels => 5x5x3 = 75 parameters
=>2nd PixelHop++ unit has 5x5 size of window, 42 channels => 5x5x42 = 1050 parameters
=>3rd PixelHop++ unit has 5x5 size of window, 273 channels => 5x5x273 = 6825 parameters

Module2
=>1st LAG unit has 50 dimension of output, n1 features after feature selection
= > 50 x (4116+1) = 205,850
=>2nd LAG unit has 50 dimension of output, n2 features after feature selection
= > 50 x (3412+1) = 170,650
=>3rd LAG unit has 50 dimension of output, n3 features after feature selection
= > 50 x (265+1) = 13,300

Module3
Random Forest Classifier has been used.
Number of estimators:
Bootstrap = true

=>Total number of parameters of the model used in HW6 = 397,750

<Training time>
Module1 (number of train images: 6000)
(1) Fit stage
56 seconds
(2) Transform stage
16 minutes

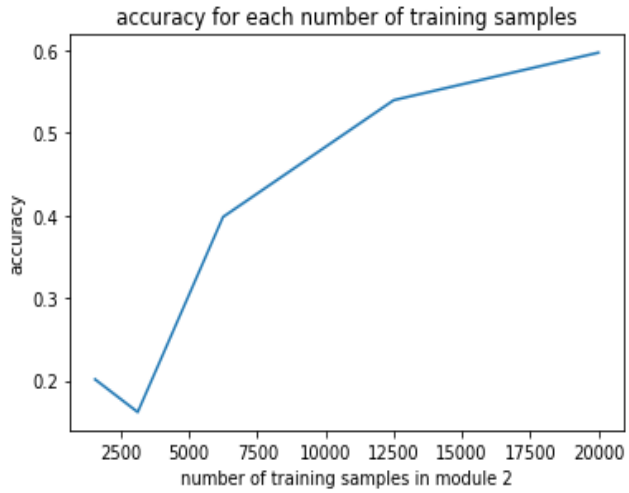
Module2 (number of train images: 20000)
(1) Feature selection stage
23 minutes
(2) LAG unit
0.4 seconds

=>Total training time: 39 minutes

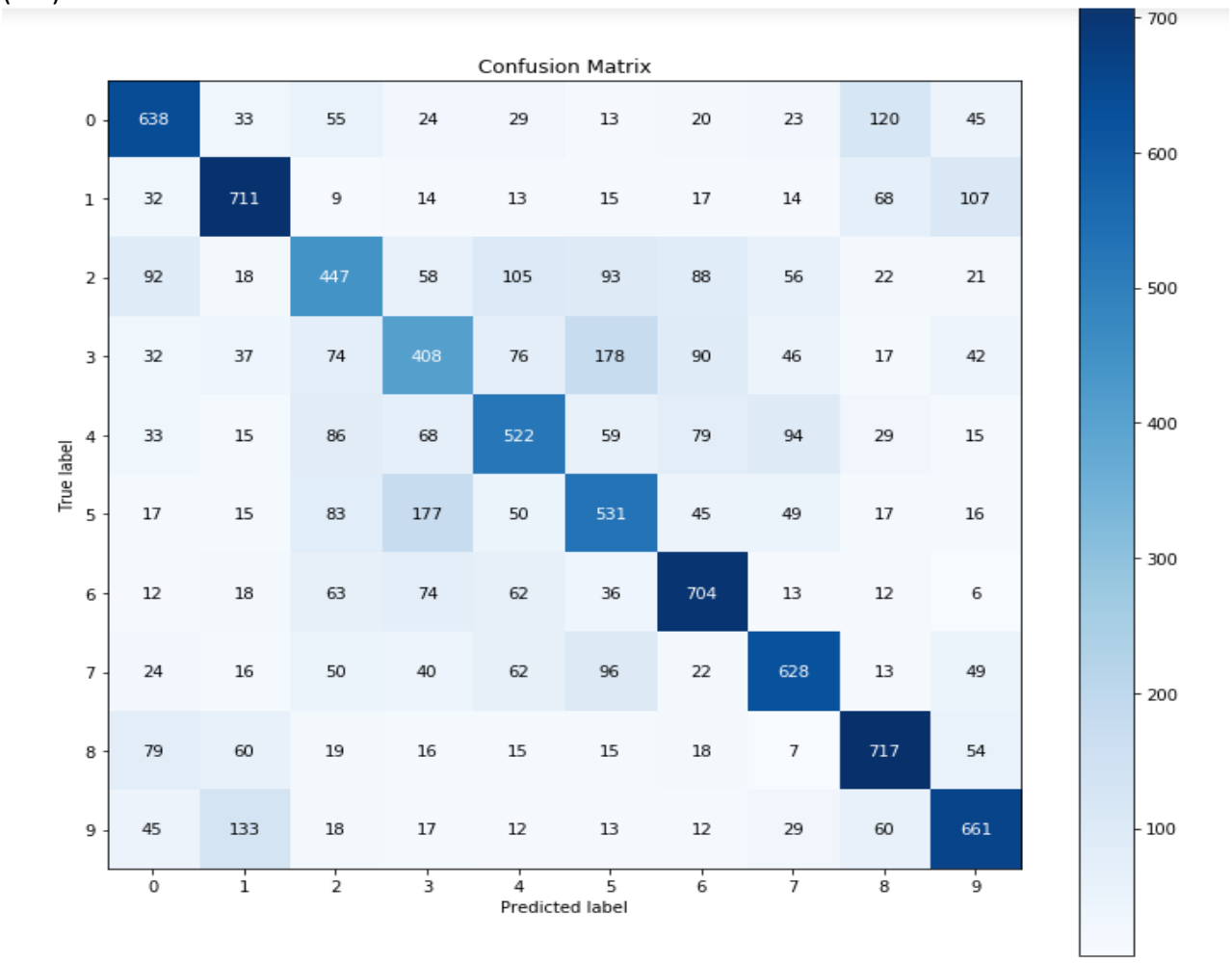
Train accuracy:
1 = 100%

(a-2)
Test accuracy:
0.5967 = 59.67 %

(a-3)

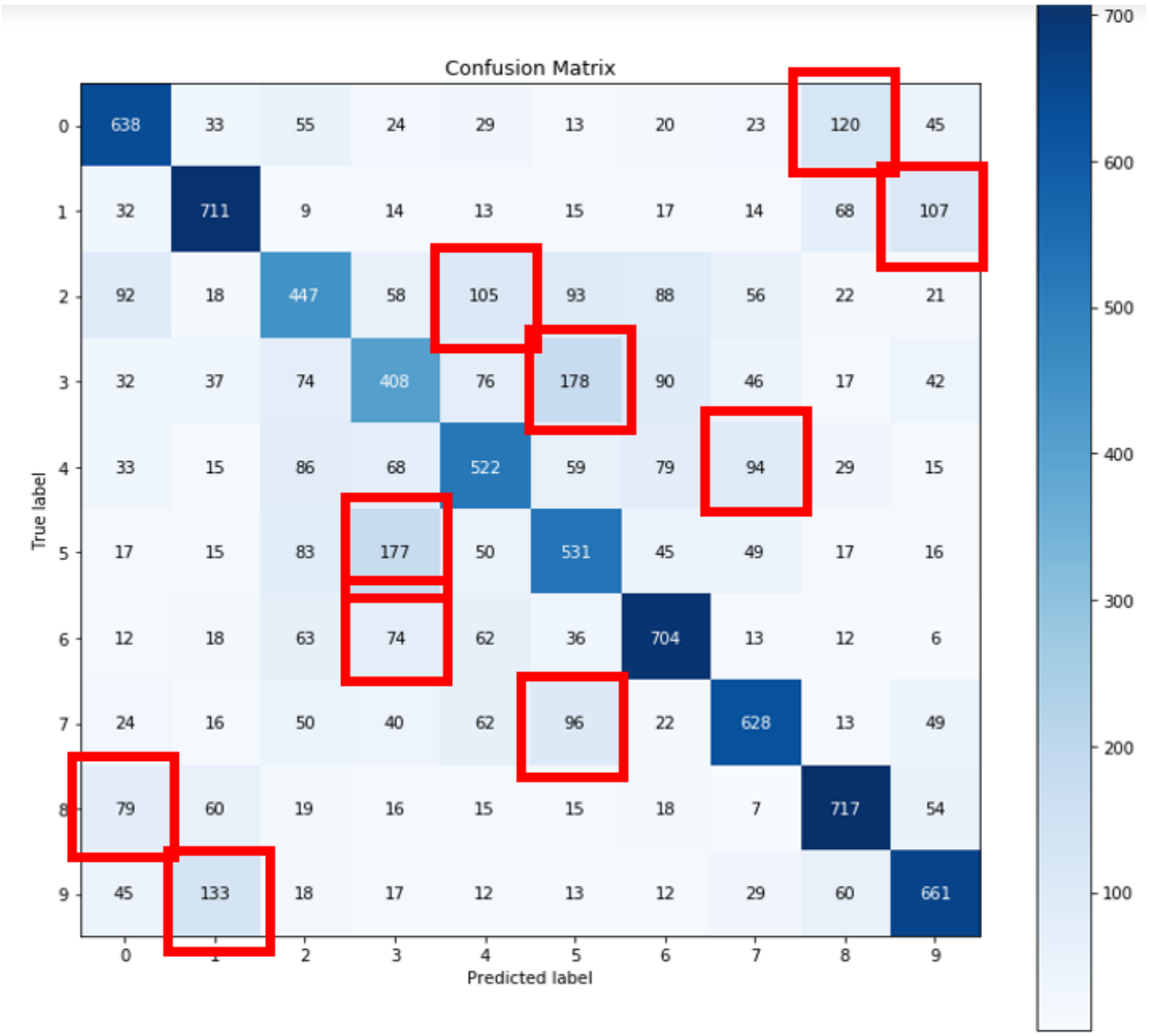


(b)
(b-1)



=>Ship class yields the lowest error rate.
=>Cat class is the most difficult one.

(b-2)



Class reference
{'plane': 0, 'car': 1, 'bird': 2, 'cat': 3, 'deer': 4, 'dog': 5, 'frog': 6, 'horse': 7, 'ship': 8, 'truck': 9}
Confusing groups:
(0,8), (1,9), (2,4), (3,5), (4,7), (5,3), (6,3), (7,5), (8,0), (9,1)
=>Most confusing group
(5,3) = (dog, cat)

=>Second most confusion group
(9,1) = (truck, car)

2.4 Discussion

(a)

(a-1)

Accuracy on train set is 100% while accuracy on test set is only 59%. There seems to be clear overfitting in my random forest model. I was trying to overcome overfitting by cross validation, but I couldn't find a set of proper hyperparameters. In the future work (may be problem 3), I will put more effort on this part, so that I would avoid over fitting. One possible explanation is that since I couldn't use all 50000 train samples for training at module 2 due to limited computational resource, it could lead the model to overfitting.

Even though I couldn't use 10000 train samples and used only 6000 train samples at module 1, it won't have significant performance drop compared to the case that DL model use 6000 train samples instead of 10000 samples. This is because covariance matrix used in PCA in Saab transform converges fast as a number of train images increases. It means at some point, covariance matrix would be the same a covariance matrix computed by a large number of train images. To be more specific, each training sample has a lot of patches and the train images and the patches have correlation structure.

(a-2)

My model has 59~60 % accuracy on test data set. This might not be the best performance that SSL can achieve. Fine hyper-parameter tuning needs to be done in our future work.

(a-3)

As we can expect, performance of the trained model drops as a number of train data sample decreases. However, this study will become more valid if we compare the results with results from LeNet-5 under the same condition (i.e., using the sample number of images for training) In future work, we will show that the rate of performance drop for PixelHop++ will be slower than rate of performance drop for LeNet-5.

This is because, since we don't need labeled data to train module 1, it provides ability for model to be trained better as a number of labeled data decreases while DL model does not because it is highly relying on labeled data set.

(b)

(b-2)

It turns out dog and cat classes are the most confusing classes to each other. This is because both classes have similar characteristics such as fur, similar face, and body structure.

Figure 10. Similar cat and cog images.[5]

Additionally, truck and car seem to be confusing classes as well due to their similar body struction.

Figure 11. Similar truck and car images [5]

(b-3)

We can utilize LAG unit to resolve the confusion that occurs among the classes mentioned earlier. E.g., cat & dog, car & truck. If we think about this situation, in our trained model, mis predicted cat is seen as dog and if we could create more diversity in dog class with LAG unit by setting more seeds, cat will less likely be predicted as a dog class. We can apply the same strategies for other classes as well. However, we need to pay attention to the dimension of the final feature vector because high dimension of feature vector could lead to overfitting.

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*, vol.60, pp. 346–359, 2019.

[2] Kuo, C., 2016. Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, 41, pp.406-413.

[3] Yueru Chen and C-C Jay Kuo, "Pixelhop: A successive subspace learning (ssl) method for object recognition," Journal of Visual Communication and Image Representation, p. 102749, 2020.

[4] Yueru Chen, Mozhdeh Rouhsedaghat, Suya You, Raghuveer Rao, C.-C. Jay Kuo, "PixelHop++: A Small Successive-Subspace-Learning-Based (SSL-based) Model for Image Classification," https://arxiv.org/abs/2002.03141, 2020

[5] CIFAR-10 dataset (online) Available: https://samyzaf.com/ML/cifar10/cifar10.html