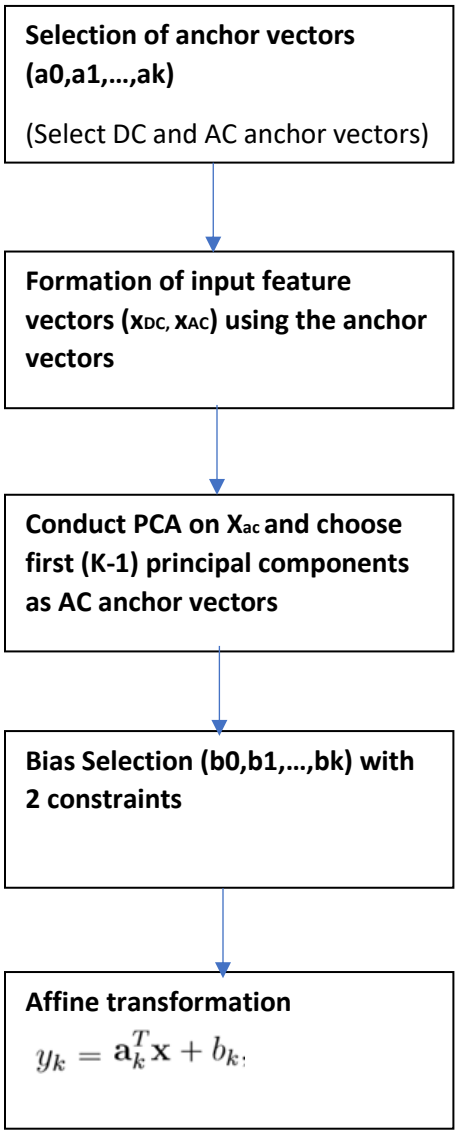
Name: Shreya Kate
USC ID: 2334973997
Email: shreyak@usc.edu

Homework 6

Problem 1: (a)

(1) Flow diagram

Saab transform



The saab transform involves affine transformation.

$$y_k = \sum_{n=0}^{N-1} a_{k,n} x_n + b_k = \mathbf{a}_k^T \mathbf{x} + b_k, \quad k = 0, 1, \cdots, K-1$$

There are 2 main steps in affine transformation, namely, selection of anchor vectors and selection of the bias term.

The anchor vectors are selected in the following way. While selecting the anchor vectors, the bias term is set equal to 0 (bk=0). Next, we need to divide the anchor vectors into 2 parts. One is the DC anchor vector and the other is the AC anchor vector.

The DC anchor vector is:

$$\mathbf{a}_0 = \frac{1}{\sqrt{N}}(1, \cdots, 1)^T$$

The AC anchor vectors are:

$$\mathbf{a}_k,\ k = 1, \cdots K-1$$

The input vector space is represented as follows:

$$\mathcal{S} = \mathcal{S}_{DC} \oplus \mathcal{S}_{AC}$$

For all vectors x, the DC and AC projections are calculated as follows:

$$\mathbf{x}_{DC} = \mathbf{x}^T \mathbf{a}_0 = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n$$

$$\mathbf{x}_{AC} = \mathbf{x} - \mathbf{x}_{DC}$$

The subspaces of AC and DC are orthogonal to each other.

Next, we do PCA on all the xAC and choose the first (K-1) principal components as AC anchor vectors.

We are done choosing the anchor vectors. Next step is to select the bias terms. The selection process has 2 constraints.

The purpose of the bias term is to make the sign of the output positive and it is important for mathematical tractability. It gives a positive output response because it shifts it by a constant value.

The 2 constraints are as follows:

1. We need the output response to be positive, which is described by the equation:

$$y_k = \sum_{n=0}^{N-1} a_{k,n} x_n + b_k = \mathbf{a}_k^T \mathbf{x} + b_k \geqslant 0.$$

2. We have all the bias terms to be equal b0=b1=…=bk

The first constraint leads to the following:

$$z_k = \phi(y_k) = \max(0, y_k) = y_k$$

Because of this result, the ReLU operation is made redundant.

The second constraint makes the design of saab transform simpler and we get the following equation:

$$\mathbf{y} = \mathbf{x}^T \mathbf{a}_0 + \sum_{k=1}^{K-1} \mathbf{x}^T \mathbf{a}_k + d\sqrt{K}\mathbf{1}$$

In this, the first term is assumed to be zero, the second term belongs to subspace of AC and the second term belongs to subspace of DC. Therefore, the third term does not have any impact on AC subspace when multiple affine transforms are cascaded. The saab transform is there in the convolutional layers.

The final bias selection rule is summed up as follows**:**

$$b_k \geqslant \max_{\mathbf{x}} ||\mathbf{x}||, \quad k = 0, \cdots, K-1$$

(2)

- In BP, there is a fixed network architecture, cost function and training and testing data. This is decided. Improvements can be made by changing the architecture or optimizing the cost function. Data labels are needed from the start.

  In FF, the covariance matrix is used to define spatial-spectral transformations in the convolutional layers and LSR is used for the FC layers. Here, data labels are not needed in the convolutional layers and are needed in the FC layers. This is a PCA-based technique. PCA is used for dimension reduction.

- The mathematical tool used in BP is stochastic gradient descent and FF uses linear algebra and statistics.
- The BP-CNN are very difficult to explain mathematically, whereas the FF-CNN design can be explained very well by mathematics.
- BP-CNN is end-to-end connected, i.e. we cannot separate that into two blocks of convolutional layers and FC layers. The FF-CNN can be seen as 2 blocks, one is the

convolutional layers and the other is FC layers. This means that the FF-CNN can be separated into feature extraction and classification modules.

- The network architecture in BP-CNN is known and fixed so it is more prone to adversarial attacks. In FF-CNN we can change the algorithm at the FC layers so we can avoid the adversarial attacks to a certain extent.
- The training complexity in BP-CNNs is high since all the training data is needed for the training process, we have to go through the training data several times since almost 50-100 epochs are needed for training. However, in FF-CNN all the training data is not needed. Only a small fraction of the data is seen to give convergence. So the training complexity is lower.
- We can change the classifier in FF-CNN, that is, we can change the classifier from SVM to RF etc. In BP, this option is not there since the network is end-to-end connected. FF-CNN is flexible whereas BP-CNN is not.
- To do multiple tasks on the same data, we need different CNN networks for BP. Whereas, if we use FF-CNN, we can keep the convolutional layers the same and change the FC layers depending on which task we need to execute. So, FF-CNN is generalizable.
- Currently, the performance of BP-CNN is superior to FF-CNN. Research is ongoing in FF-CNN.

Problem 1: (b)

(1) SSL is Successive subspace learning. As the name suggests, it involves taking a subspace from the feature space and learning on that subspace and not the entire feature space.

An SSL has the following four things:

1. It has various successive operations to get a close-up view and a view from a distance, of the image under consideration. It uses filters and max pooling techniques to implement this. It is called PixelHop. Each hop leads to looking at the image from a farther view.

2. It uses PCA to reduce the dimensionality. This is unsupervised learning as no labels are used in this step. Saab transform is used to eliminate the sign confusion problem. Due to this non-linear activation function is not needed.

3. It uses Label Assisted Regression (LAG) to do supervised learning, i.e. use the labels to reduce the dimensions further.

4. In the last step, the features are concatenated from the various views (close-up and the far views) and then classification is done using some classifier. This leads to the formation of 1 feature vector.

DL and SSL : Similarities

- Both DL and SSL have neighborhoods or receptive fields that are gradually growing.

- Both sacrifice spatial dimensions for spectral dimensions. That is, both have growing spectral dimensions whereas their spatial dimension goes on reducing in each convolutional layer (DL) or hop (SSL).
- Both use spatial pooling or max pooling for spatial dimension reduction.

DL and SSL : Differences

- DL has a fixed network architecture and the number of parameters is large. It is overparameterized. The number of parameters is more than the number of training samples.
  In SSL, this is not true. The model is not fixed. It can be changed at any stage. The filters can be reduced or added at any stage.
- In DL, it is difficult to accommodate new data. Whereas in SSL, it is easy because it is a non-parametric model. We can check if the model can accommodate new data, otherwise we can add saab filters. This can be done since the dimension reduction here is unsupervised.
- DL has BP so we need to select a cost function. In SSL, there is no BP and we can apply ensemble learning methods. DL is difficult to explain since it is not clear what exactly happens in the model. In SSL, the mathematics behind feature extraction and all steps is explainable.
- SSL is a feedforward system and the dimension reduction and feature extraction is done in one pass. It uses supervised and unsupervised learning to do so. DL uses BP to extract features. It has end to end optimization.
- Due to BP, DL's complexity for training the data is high. SSL's complexity is comparatively less for a small number of convolutional layers.
- DL method is easily exploitable since its path is known and it has BP. One can create fake images and fool the system. It is highly prone to adversarial attacks. Since SSL is a feedforward network, its path is not known to attackers and hence it is tougher for adversarial attacks to occur.
- DL requires a large amount of labelled data. SSL does not require such a large amount since the pixelhop layers use unsupervised learning. Therefore, SSL performs better under weak supervision.
- DL needs label along with input images to extract features. SSL has two types of features, one are task dependent which are extracted using label while the others are task independent which are extracted by using PCA and saab transform.
- DL uses cost functions to extract the features before classification whereas SSL uses LAG units for the same purpose.

(2)

SSL has 3 modules. The functions of each module are explained below:

Module 1:

In this module there are I pixelhop units in cascade. The input image goes in the first pixelhop unit and it undergoes PCA followed by saab transform. The pixelhop unit is unsupervised learning, i.e. labels are not needed in this layer. This step is like the convolutional layers in deep learning. The output of the first pixelhop unit has spatial pooling (max pooling) done on it and then this max pooled output goes in the second pixelhop unit and so on till I th pixelhop unit. Not all of the training images are fed to Module 1. Only some of the training images are fed to module 1. They are fitted and then the model is saved.

From one hop to the next, the input image is scanned in a close up look and also on how it looks from far. So, the object features can be extracted with more knowledge on how the object looks.

Module 2:

The output of max pooling from each pixelhop unit is fed to module 2 where in the first step, aggregation takes place. The aggregation is done by taking the maximum, mean and minimum values of the output. The dimension is reduced further by doing this. Module 2 has supervised learning since it uses labels in the LAG unit. LAG unit is Label Assisted Regression. The lag unit uses regression to extract the features and make the feature subspace into a feature vector of dimension M. We give all the training data in Module 2 to get best results.

Therefore, we get a I feature vectors from the LAG units which is then fed to the classifier.

Module 3:

Module 3 comprises of the classifier. We use Random Forest classifier or any other multi-class classifier like SVM etc. The M-dimensional feature vectors are concatenated to form a M x I matrix and then fed to a classifier which then trains on this concatenated matrix by properly tuning the parameters. We get the predicted object class as the output.

(3) The Pixelhop method uses Saab transform and Pixelhop++ uses channel-wise Saab transform. In both these methods, the transforms are used in module 1, where unsupervised learning takes place. It has spatial and spectral components. So its dimensions are Si x Si x Ki.

In the normal saab transform, the spatial and spectral components are changed from Si x Si x Ki to Si+1 x Si+1 x Ki+1. So, in this saab transform the input image is taken as the spatial and spectral components together.

In the channel-wise saab transform this input is taken as Ki images of dimension Si x Si. After the transform this is converted into Ki+1 images with dimension Si+1 x Si+1.

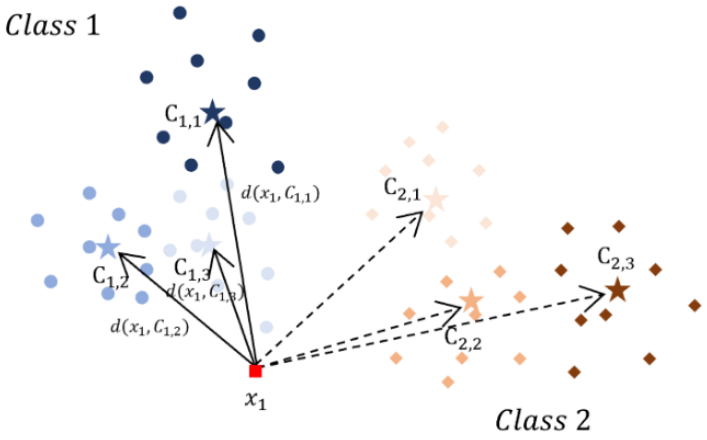
The same kernel and bias design method is used in saab as well as channel-wise saab.

If Si=5, then the input dimension for Pixelhop is 5x5xKi = 25xKi but for channel wise saab transform the input dimension is 5x5=25.

(4) LAG

The full form of LAG is Label Assisted reGression. This is done in Module 2 and it is the supervised learning part in Pixelhop and Pixelhop++. It uses label. It transforms features from one space to another. It considers intra- and inter-class variation. It has 4 steps.

In the first step, samples of the same class are clustered to create object-oriented subspaces. There are multiple seeds in each class to account for intra-class variability. This represents the diversity in each class. The clusters and seeds can be seen in the image below:



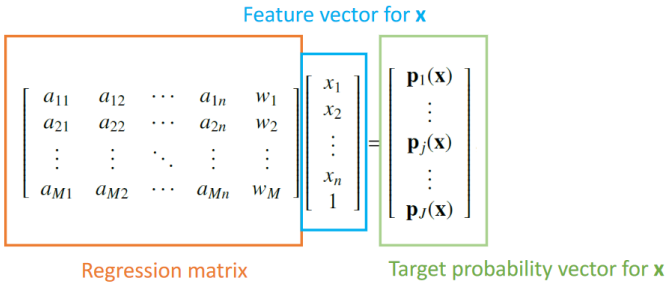
In step 2, a target probability vector is constructed. This is based on the Euclidean distance between the data points and the seeds of each class. An example is shown here, if there are 2 classes with 3 seeds in each class then the target probability vector looks like this:

$$\begin{cases} \text{Prob}(\mathbf{x}_j, \mathbf{c}_{j',l}) = 0 \quad \text{if } j \neq j', \\ \text{Prob}(\mathbf{x}_j, \mathbf{c}_{j,l}) = \dfrac{\exp(-\alpha d(\mathbf{x}_j, \mathbf{c}_{j,l}))}{\sum_{l=1}^{L} \exp(-\alpha d(\mathbf{x}_j, \mathbf{c}_{j,l}))}, \quad \text{if } j = j' \end{cases}$$

If x1 is from class #1, then:

$$\text{Target}(x_1) = \begin{bmatrix} P_1(x_1) \\ P_2(x_1) \end{bmatrix} = \begin{bmatrix} Prob(x_1, c_{1,1}) \\ Prob(x_1, c_{1,2}) \\ Prob(x_1, c_{1,3}) \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In step 3, the regression matrix is calculated by mapping the feature vector to the target probability vector.



In step 4, we use the regression matrix on a test dataset to predict the probability vector.

The advantage of LAG is that we get this probability vector which is like the transformed feature, which has a higher discriminant power than the original features of the test data. Because of this the LAG unit enables the model to extract better features and hence classify the data with higher accuracy.

Problem 2: (a)

I trained module 1 on 10000 images. Each class has 1000 images. All 50000 training images are used for modules 2 and 3. I have implemented this homework using Google Colab.

I used Ns as 50%.

(1)

Training time:

The training time for Module 1 is 84.1647 seconds.
The training time for Module 2 is 979.7584 seconds.
The training time for Module 3 is 194.3695 seconds.
The total training time is 1258.2926 seconds, which is equal to 20.9715 minutes.

Training accuracy:

The training accuracy for 50,000 training images is **93.484 %**

Model size:

Part A

Hop 1: 14x14x40 = 7840
Hop 2: 5x5x281 = 7025
Hop 3: 1x1x539 = 539

Part B

$M\times(n+1)$
Hop 1: 50 x ((14x14x40)+1) = 3,92,050
Hop 2: 50 x ((5x5x281)+1) = 3,51,300
Hop 3: 50 x ((1x1x539)+1) = 27,000

Part C (Classifier)

(number of hops)x(M)x(number of classes)

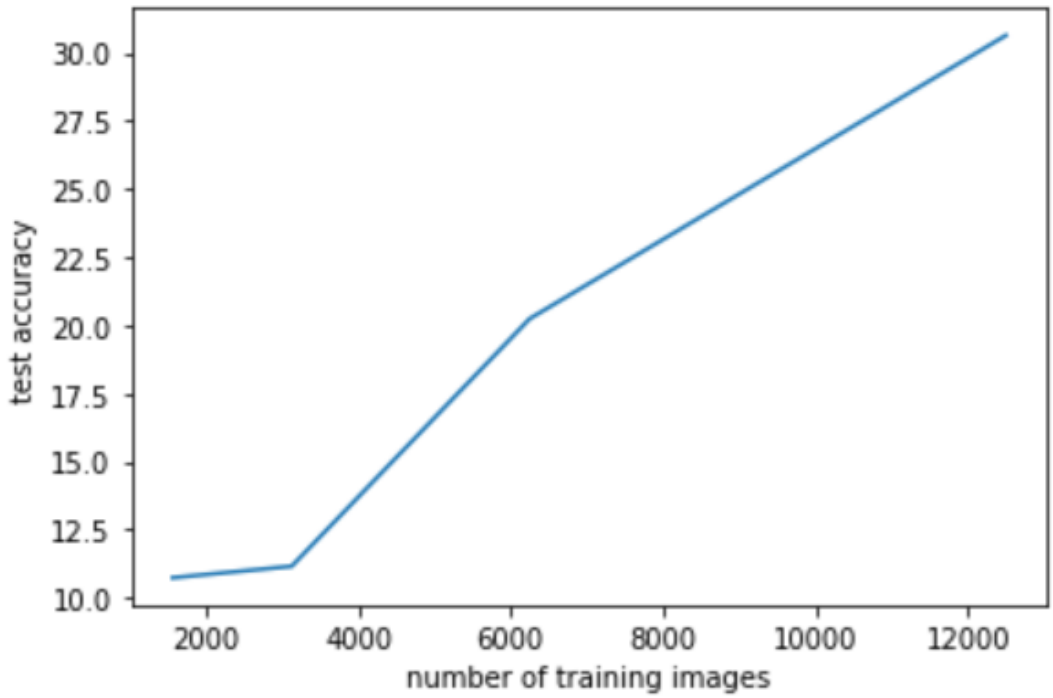
3x50x10 = 1500

(2) I tested the model on 10,000 test images.

The testing accuracy is 43.06 %

(3) Test accuracy for reduced labeled training data in modules 2 and 3

Number of training images	**Training Accuracy**	**Test Accuracy**
(1/4)*50000 = 12500	97.936	30.63
(1/8)*50000 = 6250	99.552	20.24
(1/16)*50000 = 3125	100	11.16
(1/32)*50000 = 1562	100	10.74



The plot shows the test accuracy vs number of training images.

As the number of training images gets lesser, the test accuracy drops.

12500 training images reduce the test accuracy to 30.63%. 37500 less training images lead to an accuracy drop of about 12 %.

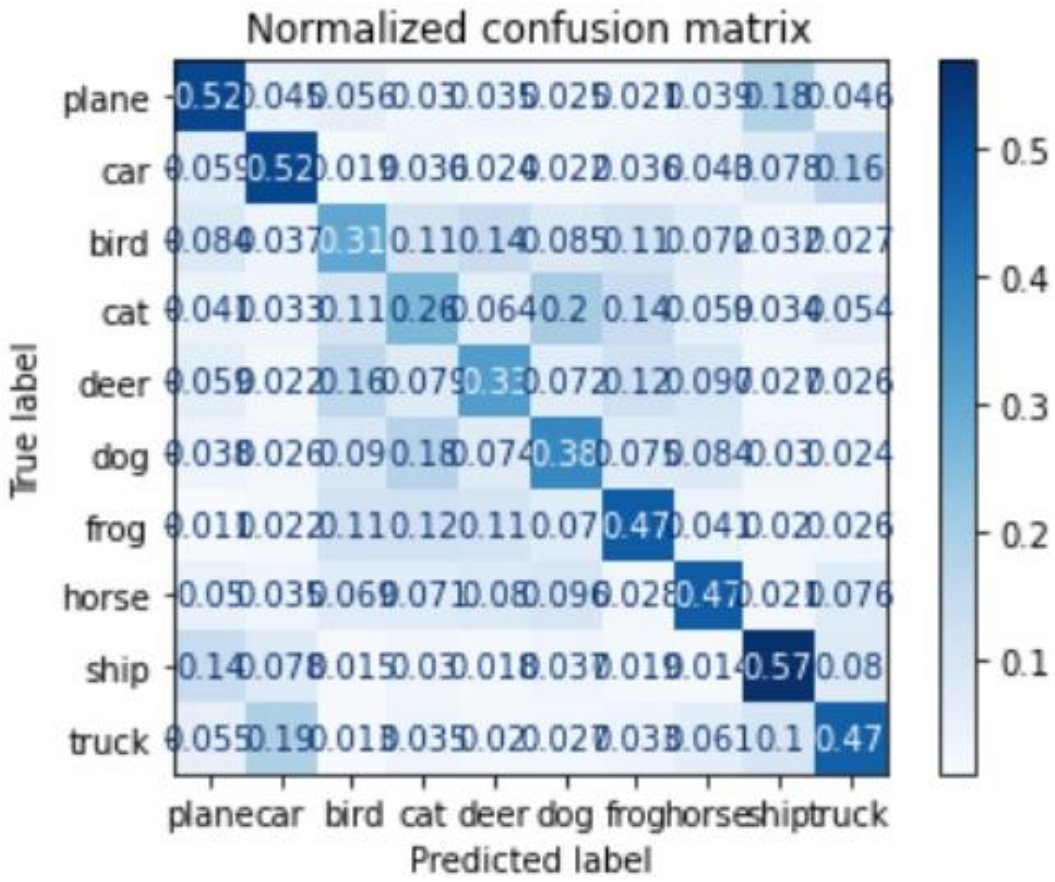
As we further reduce the training images to 6125, the accuracy drops to 20.24%. Here, the number of training images are lesser than the testing images. So such a low test accuracy value is expected.

The accuracy drops to 11.16% when training images are 3125. This is a very bad accuracy. When the number of training images drop to 1562, the test accuracy is 10.74%, which is almost the same as the previous case. Here, the number of training images are almost $1/10^{th}$ of the number of test images. So a very poor test accuracy value is expected.

Problem 2: (b)

(1)

The confusion matrix is computed as a heat map. The values in the heat map can be seen clearly in the matrix below.




```
Normalized confusion matrix
[[0.52 0.04 0.06 0.03 0.04 0.03 0.02 0.04 0.18 0.05]
 [0.06 0.52 0.02 0.04 0.02 0.02 0.04 0.04 0.08 0.16]
 [0.08 0.04 0.31 0.11 0.14 0.09 0.11 0.07 0.03 0.03]
 [0.04 0.03 0.11 0.26 0.06 0.2  0.14 0.06 0.03 0.05]
 [0.06 0.02 0.16 0.08 0.33 0.07 0.12 0.1  0.03 0.03]
 [0.04 0.03 0.09 0.18 0.07 0.38 0.07 0.08 0.03 0.02]
 [0.01 0.02 0.11 0.12 0.11 0.07 0.47 0.04 0.02 0.03]
 [0.05 0.04 0.07 0.07 0.08 0.1  0.03 0.47 0.02 0.08]
 [0.14 0.08 0.01 0.03 0.02 0.04 0.02 0.01 0.57 0.08]
 [0.06 0.19 0.01 0.04 0.02 0.03 0.03 0.06 0.1  0.47]]
```

The ‘ship’ class has the highest accuracy and hence it has the lowest error rate.

The ‘cat’ object class is the most difficult since it has the lowest accuracy and highest error rate.

(2)

1. Plane and ship are confused with each other because both these images have a streamlined shape and have the background blue.

Plane Ship


This is an image of a plane which can be mistaken for a ship by the model. 18% of images of plan are classified as ship and 14% of ship images are classified as plane.

A bird and a plane can also be mistaken. For example, the image of the plane below could be classified as a bird. This happens because both fly in the sky and the wings of a plane could be mistaken for the wings of a bird. An example of a plane image that looks like a bird:

2. Cats and dogs can be mistaken for each other because many of the CIFAR 10 images are not that clear and show only the face or only the body of the animals, which makes it very difficult to classify them correctly. This also happens because both classes have fur and similar colors.

20% of cat images are classified as dog images and 18% of dog images are classified as cat images.

Dogs


Cats


3. Car and truck class is also a confusing class because both are vehicles with wheels and have roads in the background. The examples are shown below:

Cars


Trucks


16% of the car images are classified as truck and 19% of truck images are classified as car images.

(3)

The following things can be done to improve the accuracy of the difficult classes.

I can modify the hyperparameters in the LAG units and feature selection to improve accuracy of the model.

I can change the energy thresholds or the window size to improve accuracy.

I can increase the number of hops to get a more near-to-far view of the image which will lead to better feature selection.

I can implement different types of pooling and then combine their results. For eg. I will do mean pooling, max pooling and min pooling. This will lead to better pooling and hence better feature selection.

I could also try using a different classifier in module 3 or change the hyperparameters more optimally in the random forest classifier to improve test accuracy of classification.