Name: Bhavya Samhitha Mallineni
USC ID: 6580252371
USC Email: mallinen@usc.edu
Submission Date: 05/01/2024

REPORT
HOMEWORK 6

Problem 1: Origin of Green Learning

I. Abstract and Motivation :

Green Learning in artificial intelligence and machine learning emphasizes environmentally sustainable practices, aiming to reduce the ecological footprint of AI technologies by lowering energy usage and carbon emissions during training and deployment. Traditional deep learning models, which often rely heavily on backpropagation, consume significant resources and energy. Green Learning champions resource-efficient algorithmic innovations that maintain performance standards Feedforward-designed Convolutional Neural Networks, which forgo backpropagation in favor of feedforward methods like the Subspace Approximation with Augmented Kernels transform. This approach speeds up the training process and cuts down on energy consumption by learning features directly and efficiently. Embracing Green Learning is vital for the sustainability of AI, promoting the development of efficient algorithms and architectures that support environmental conservation and align with global sustainability objectives.

II. Approach and Procedure and Experimental Results

For this problem, the task was to read the papers. The approach is to read, and the answers are in the Discussion section

III. Discussion

(a) Feedforward-designed Convolutional Neural Networks(FF-CNNs)

1. Saab Flow Diagram:

The Saab transform is a feature extraction method used to enhance the performance of machine learning algorithms, such as CNNs. It is based on the

Saak transform but improves upon it by increasing computational efficiency and simplifying data handling. Saab begins by extracting small patches from an image. These patches are then transformed into one-dimensional vectors using linear stretching. The 1D vectors are normalized by removing the average to standardize the features and highlight the distinctive characteristics of each patch.Next, the transform separates the DC and AC components of the vectors. PCA is applied to the AC components to isolate crucial features that accurately represent the data, referred to as AC anchor vectors. These are combined with the DC components to construct a comprehensive kernel for each layer, improving feature representation by incorporating a bias term. This addition helps mitigate the sign confusion typically observed in CNNs due to asymmetric activation functions.

During the testing phase, the Saab transform follows a similar procedure. It normalizes test data using the training mean and projects it onto the established kernels, adjusting with the largest norm or bias from the training to ensure feature detection consistency and accuracy. The features are then reshaped to their original spatial dimensions, preparing them for subsequent analysis or classification. This systematic approach renders the Saab transform an effective tool for robust and clear feature extraction from complex datasets.

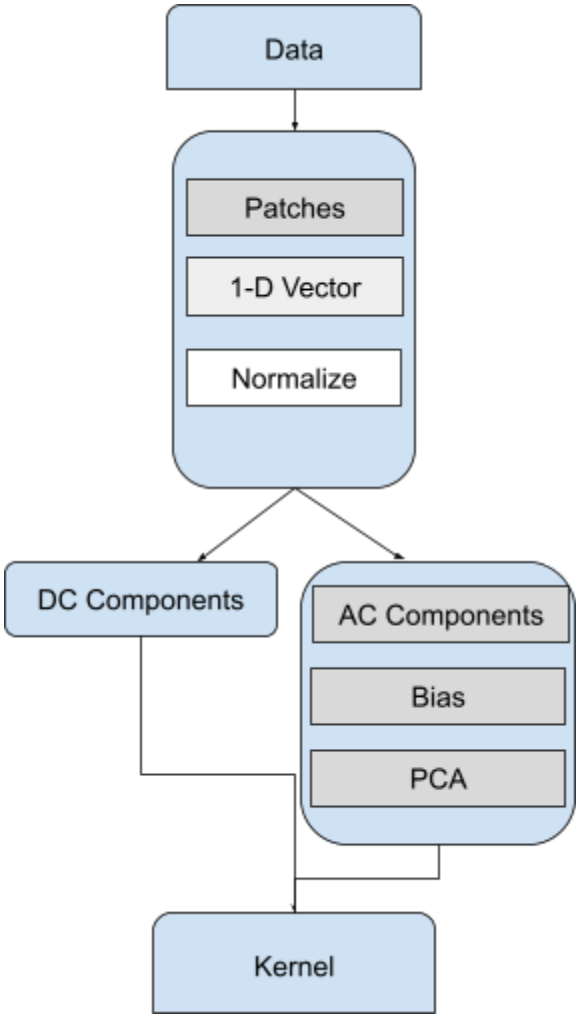


Figure1: Flowchart Saab

2. **FF-CNN and BP-CNN:**

   -> Similarities: Both have the same purpose and are used for classification tasks. They both are data-driven techniques. Each uses methods to process input through layers to extract meaningful features for classification. Both types of networks use advanced methods for classifying data.

   -> Differences: The major difference is that for learning, FF-CNN does not use backpropagation for error correction, whereas BP-CNN relies on backpropagation. FF-CNN also uses unsupervised learning to develop kernels, whereas BP-CNN uses supervised learning, continuously refining feature detection through training. Additionally, FF-CNN is faster because it's one-pass and needs fewer parameters for training, but BP-CNN is more computationally intensive due to the iterative nature of learning and extensive parameter tuning.

-> Advantages and Limitations of FF-CNN: FF-CNNs, by avoiding backpropagation, streamline training, reducing computational demands and enhancing speed. This design promotes a clearer understanding of data transformations through layers,Unlike traditional BP-CNNs which rely on iterative error correction to refine their models, FF-CNNs leverage a direct, feedforward approach that simplifies the learning process. This not only speeds up model training but also reduces the complexity involved in tuning and adjusting weights. The major difference lies in the learning approach—FF-CNNs do not utilize backpropagation for error correction, leading to potentially quicker deployments and easier scaling. However, this may also limit their ability to adjust to complex patterns as effectively as BP-CNNs, which can fine-tune their parameters through backpropagation, offering superior performance on more complex datasets.

(b) Understanding PixelHop and PixelHop++

1. SSL Methodology:-

SSL method is noted for its computational efficiency and adaptability. Unlike traditional deep learning approaches that rely on supervised parametric backpropagation for feature extraction, SSL utilizes unsupervised techniques like K-means, enhancing computational speed and reducing dependence on large training datasets. It employs methods such as the Saab or Saak transforms for effective dimension reduction, which not only refines the feature set for better generalization but also enables a more robust feature extraction across multiple scales through pooling. This multi-scale approach increases the receptive field and introduces some level of invariance. SSL's classification phase incorporates label assist regression to further reduce dimensionality, utilizing pseudo labels that outnumber actual class labels, facilitating refined subclass distinctions.

Comparing SSL with DL:-

DL and SSL both aim to uncover patterns in data, especially image data, through progressively larger contextual frameworks. However, they diverge significantly in methodology and efficiency. DL relies heavily on supervised learning, necessitating extensive data, iterative training, and substantial computational resources. It employs fixed-size, parametric kernels that cannot be adjusted after training, making the system prone to issues like gradient explosion and sign confusion during the backpropagation phase. In contrast, SSL utilizes an unsupervised approach for feature extraction, making it inherently more flexible and computationally efficient. SSL's feature extraction is

non-parametric, avoiding the stability issues common in DL, and allows for feature reduction to retain only the most useful features. Additionally, SSL applies techniques like Label Assist Gradient for effective dimension reduction in the classification phase, enabling the use of diverse classifiers like SVM and Xgboost, which are not typically used in DL. This leads to better generalization, less reliance on large datasets, and greater interpretability, positioning SSL as a robust alternative to traditional deep learning approaches in specific scenarios.

2. SSL Framework:-

Module 1 :- it uses Pixelhop or Pixelhop++ with a pooling layer,leveraging the Saab transform to project features from previous patches to a lower dimension. This cascading setup is designed to enlarge the receptive field, allowing for multi-scale feature extraction and the elimination of redundancy. It also facilitates the use of neighbor connections to comprehensively analyze input data.

Module 2 :- it focuses on unsupervised dimension reduction and aggressive feature selection. It utilizes statistical values and cross entropy criteria to highlight discriminant features for easier classification. This module also integrates Label Assisted Gradient techniques to assign pseudo labels, preparing features for the final classification stage.

Module 3 :- here, features processed by earlier modules are concatenated and fed into a final classifier, such as Random Forest, Xgboost, or SVM. This stage is responsible for the ultimate classification, leveraging supervised dimension reduction through label-assisted regression to effectively distinguish between different classes based on the attributes of near and far neighbors in the dataset.

3. Neighborhood COnstruction and Subspace Approximation :-

PixelHop and PixelHop++ modules employ neighborhood construction and subspace approximation techniques to extract features from images. Both methods start by creating local patches around pixels, capturing both immediate and broader neighborhood details. In subspace approximation, PixelHop uses the basic Saab transform, which involves subtracting the mean to focus on principal components, thereby enhancing feature extraction and introducing a bias term for non-negativity. PixelHop++ enhances this process by introducing the channel-wise Saab transform, which independently processes each color channel, allowing for better capture of channel-specific information. This adaptation enables PixelHop++ to effectively handle diverse visual data,

enhancing discrimination and performance in complex recognition tasks, especially in detailed color analysis within images.

Problem 2:

IV. Abstract and Motivation :

PixelHop and PixelHop++ are machine learning architectures that employ Successive Subspace Learning (SSL) techniques for effective feature extraction in image processing tasks. PixelHop utilizes the Saab transform to perform unsupervised feature extraction, leveraging a cascade of transformations to analyze and compress image data through spatial and spectral reduction. This process enhances the efficiency and accuracy of subsequent classification tasks. PixelHop++ builds on this by introducing channel-wise Saab transforms, which process image channels separately, improving the model's ability to handle complex features like color variations. Both models are designed to efficiently handle large datasets, with PixelHop++ offering enhanced feature discrimination and generalization capabilities compared to its predecessor. can you paraphrase this into 50 words

V. Approach and Procedure:

Here, our goal is to implement the previously discussed PixelHop and PixelHop++ modules on the MNIST and Fashion-MNIST datasets using the provided GitHub references. The provided code includes implementations for the Saab transform, channel-wise Saab transform, and PixelHop. My main responsibility was to create the "shrink" function, define the parameters, and train the model.

VI. Experimental Results:

A. Building PixelHop++ Model

1) And 2) For TH1 = 0.005 and TH2=0.001

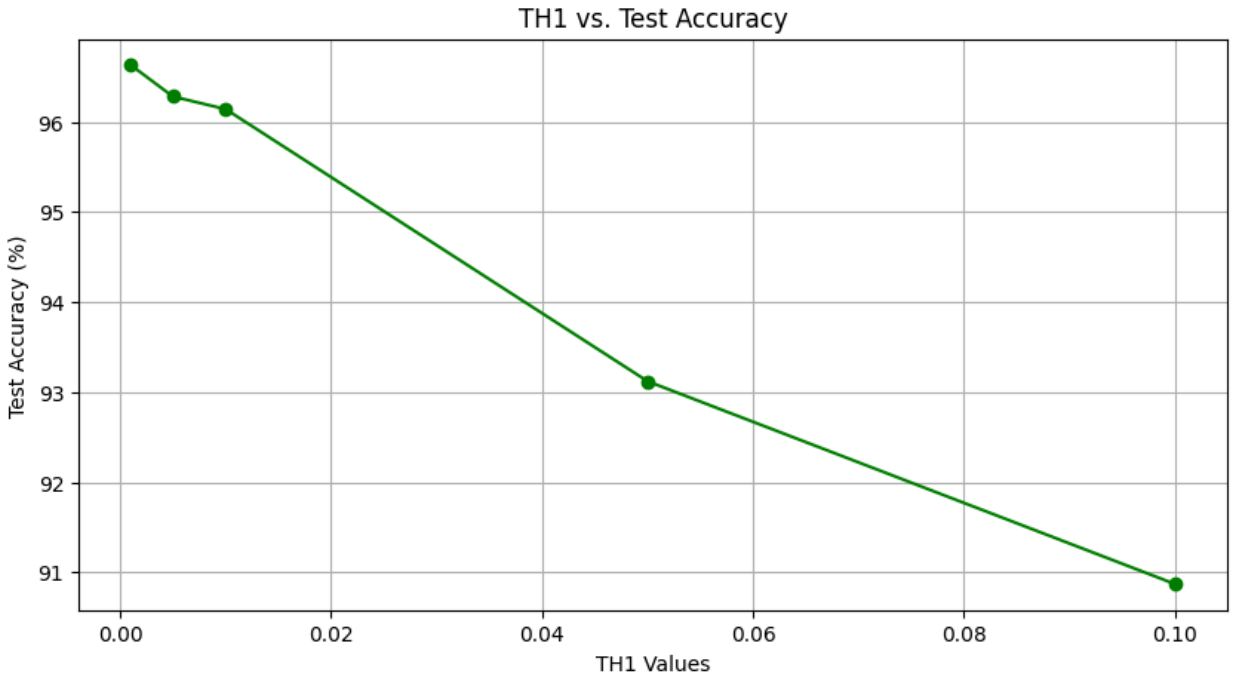
	Train Accuracy	Test Accuracy	K1	K2	K3	ModelSize	Runtime (secs)
MNIST	98.38	96.28	24	105	129	6450	336.37
Fashion MNIST	89.96	85.03	23	59	73	3875	248.95

Table 1

3) for TH2 = 0.001 and different TH1 values
MNIST:

TH1	Train Accuracy	Test Accuracy	K1	K2	K3	Model Size	Runtime (secs)
0.005	98.38	96.28	24	105	129	6450	336.37
0.001	98.56	96.63	24	105	132	6526	786.41
0.01	98.28	96.14	24	102	112	5950	229.45
0.05	95.81	93.12	24	61	32	2925	89.51
0.1	93.52	90.87	24	47	22	2325	72.34

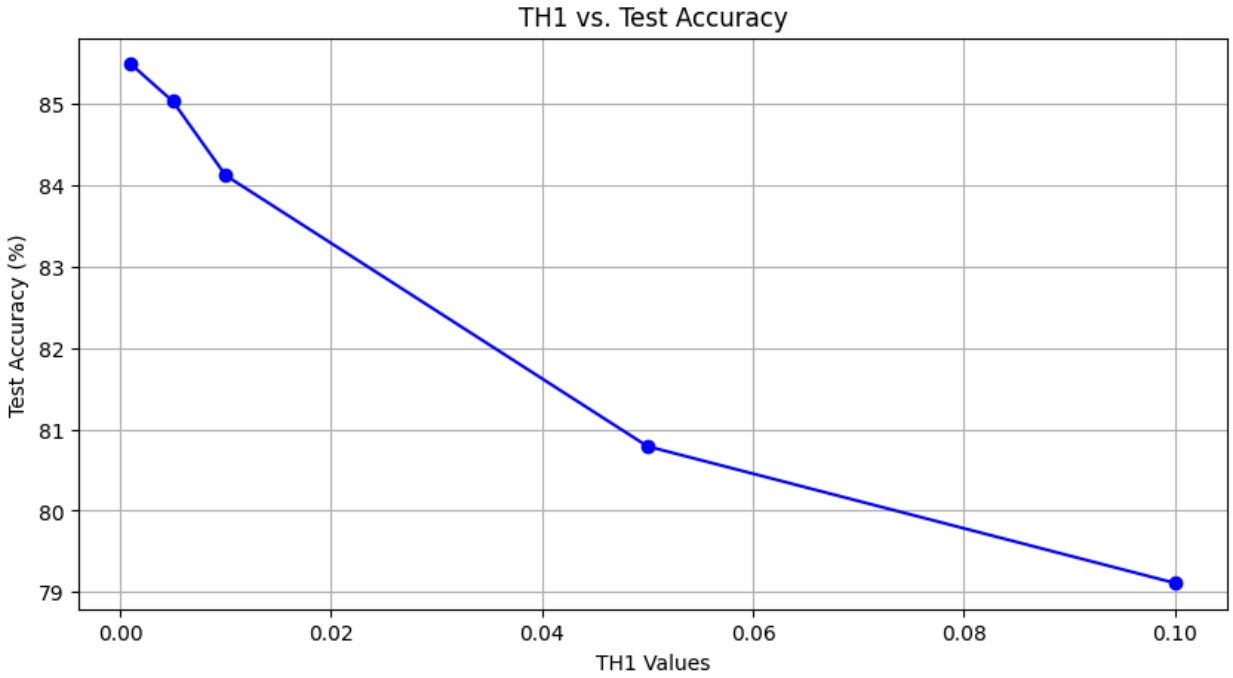
Table 2



Fashion MNIST:

TH1	Train Accuracy	Test Accuracy	K1	K2	K3	Model Size	Runtime (secs)
0.005	89.96	85.03	23	59	73	3875	248.95
0.001	90.39	85.49	23	58	82	4075	545.29
0.01	88.82	84.12	23	51	56	3250	171.60
0.05	85.29	80.79	23	28	23	1850	78.33
0.1	83.52	79.11	23	20	16	1475	62.10

Table 3



B. Pixel Hop ++ Vs Pixel hop

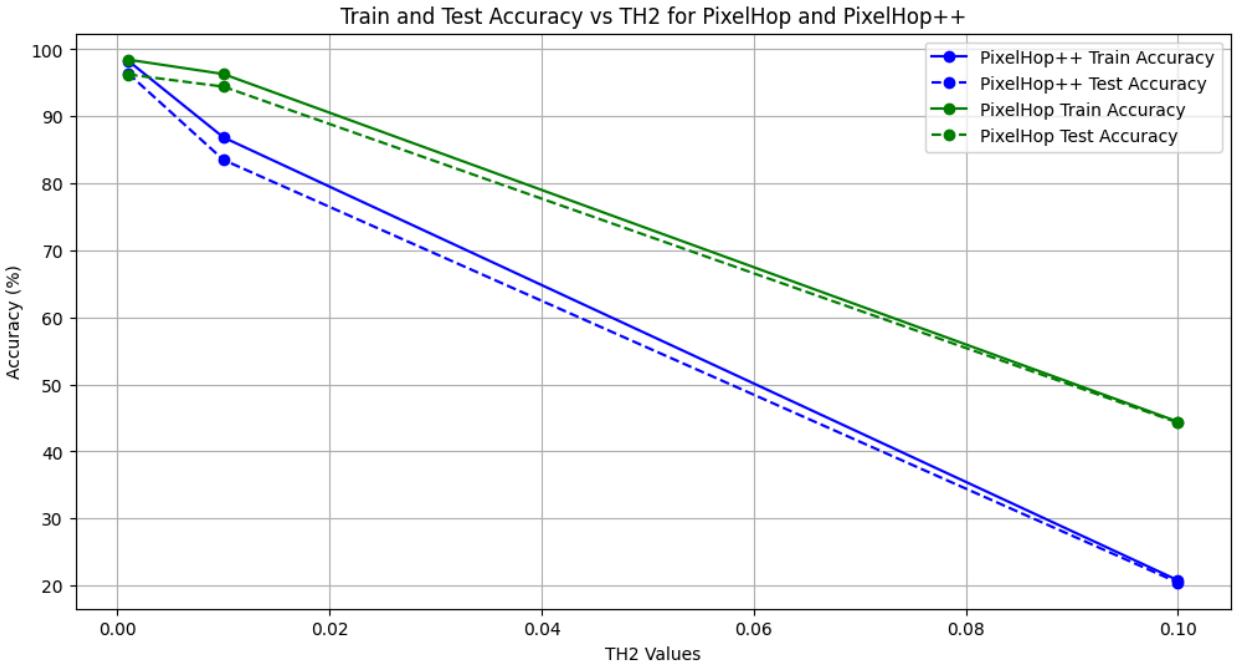
For a Constant TH1 and variable TH2 comparing PixelHop and PixelHop++

MNIST:

	TH2	Train Accuracy	Test Accuracy	K1	K2	K3	ModelSize	Runtime (secs)
PixelHop++	0.001	98.32	96.45	24	105	126	6375	415.54
PixelHop	0.001	98.47	96.26	24	154	66	346100	266.84

PixelHop++	0.01	86.87	83.51	10	19	10	975	261.65
PixelHop	0.01	96.32	94.44	10	17	17	11725	167.03
PixelHop++	0.1	20.78	20.45	3	2	1	150	100.00
PixelHop	0.1	44.45	44.31	3	4	3	675	75.90

Table 4

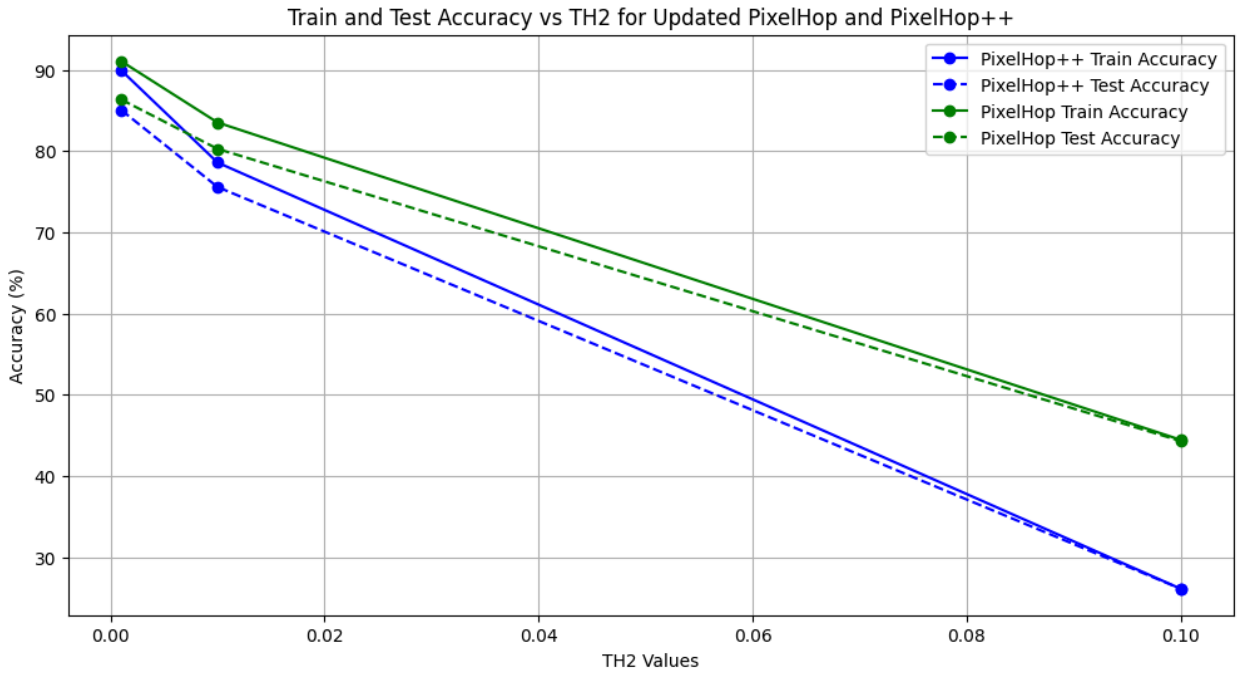


Fashion MNIST:

	TH2	**Train Accuracy**	**Test Accuracy**	**K1**	**K2**	**K3**	**ModelSize**	**Runtime (secs)**
PixelHop++	0.001	89.92	85.04	23	56	73	3800	320.36
PixelHop	0.001	91.05	86.33	23	44	43	73175	217.18
PixelHop++	0.01	78.58	75.57	8	11	12	775	198.71
PixelHop	0.01	83.51	80.26	8	10	11	4950	132.18
PixelHop++	0.1	2662	26.13	2	2	1	125	90.46

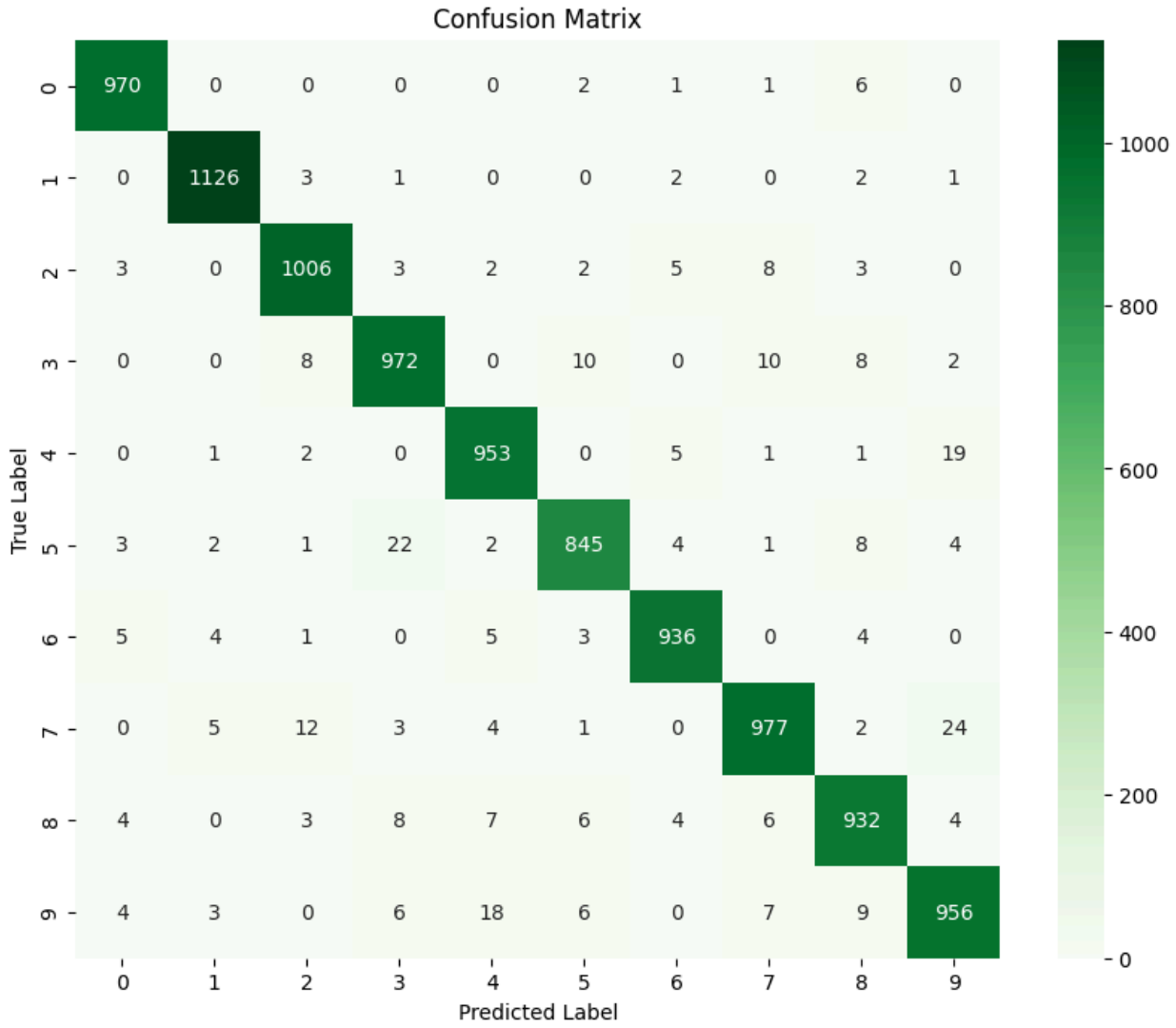
PixelHop	0.1	44.45	44.31	2	3	2	350	69.90

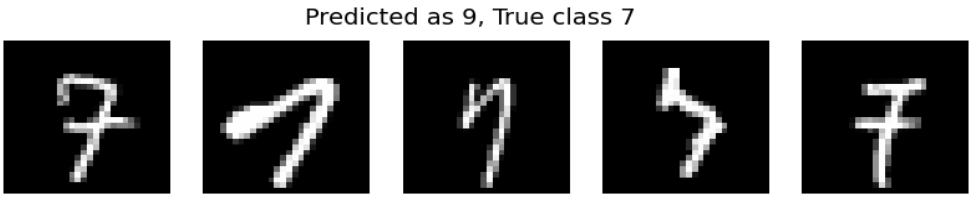
Table 5

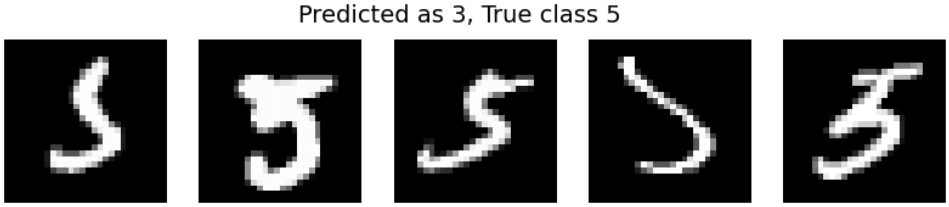


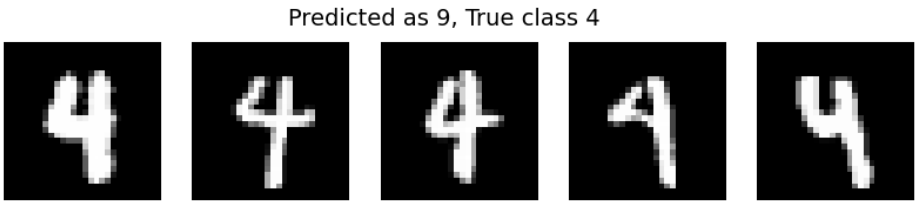
C. Error Analysis:

MNIST:

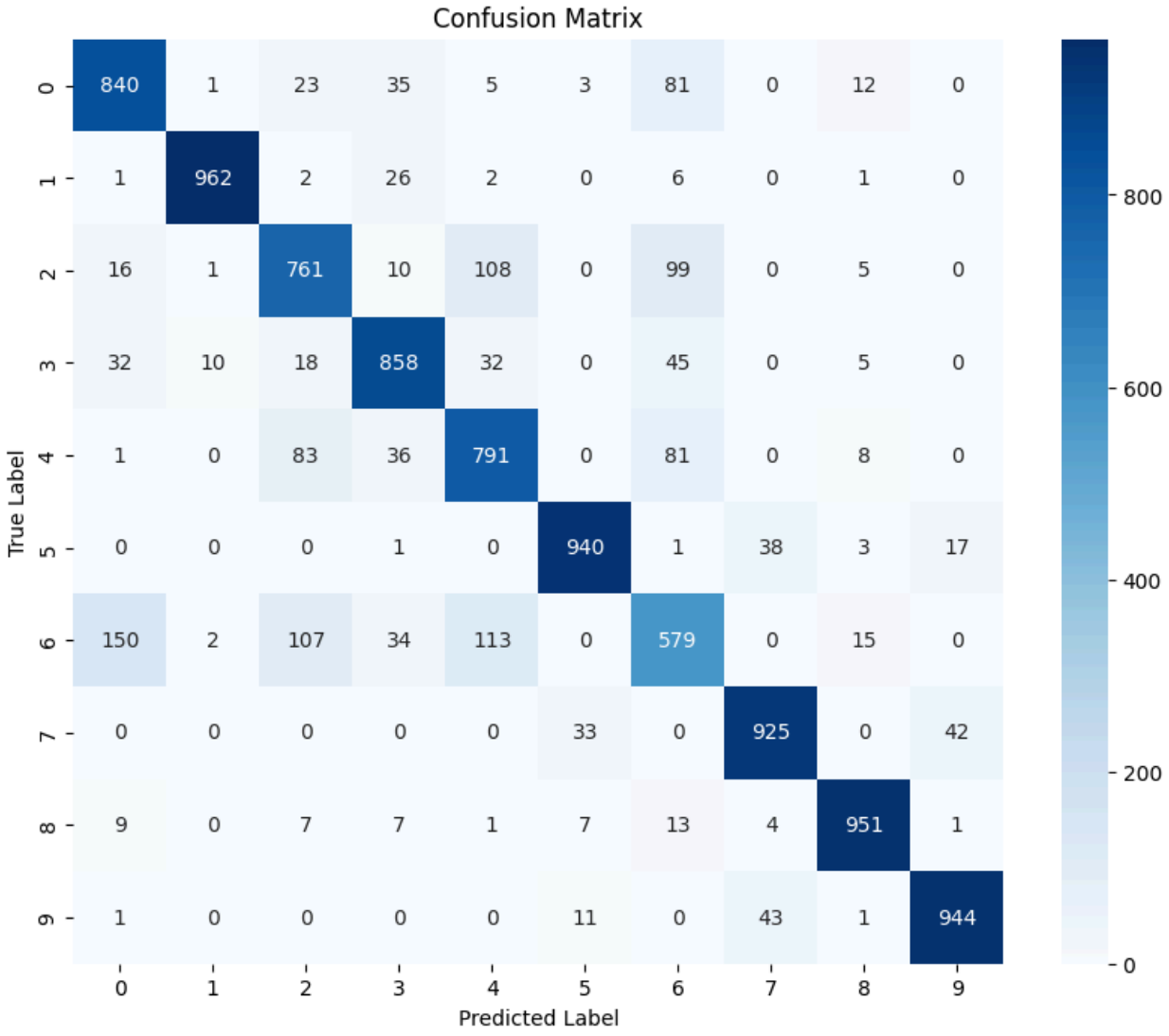

Confusion Matrix
True Label
Predicted Label
0 970 0 0 0 0 2 1 1 6 0
1 0 1126 3 1 0 0 2 0 2 1
2 3 0 1006 3 2 2 5 8 3 0
3 0 0 8 972 0 10 0 10 8 2
4 0 1 2 0 953 0 5 1 1 19
5 3 2 1 22 2 845 4 1 8 4
6 5 4 1 0 5 3 936 0 4 0
7 0 5 12 3 4 1 0 977 2 24
8 4 0 3 8 7 6 4 6 932 4
9 4 3 0 6 18 6 0 7 9 956
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0

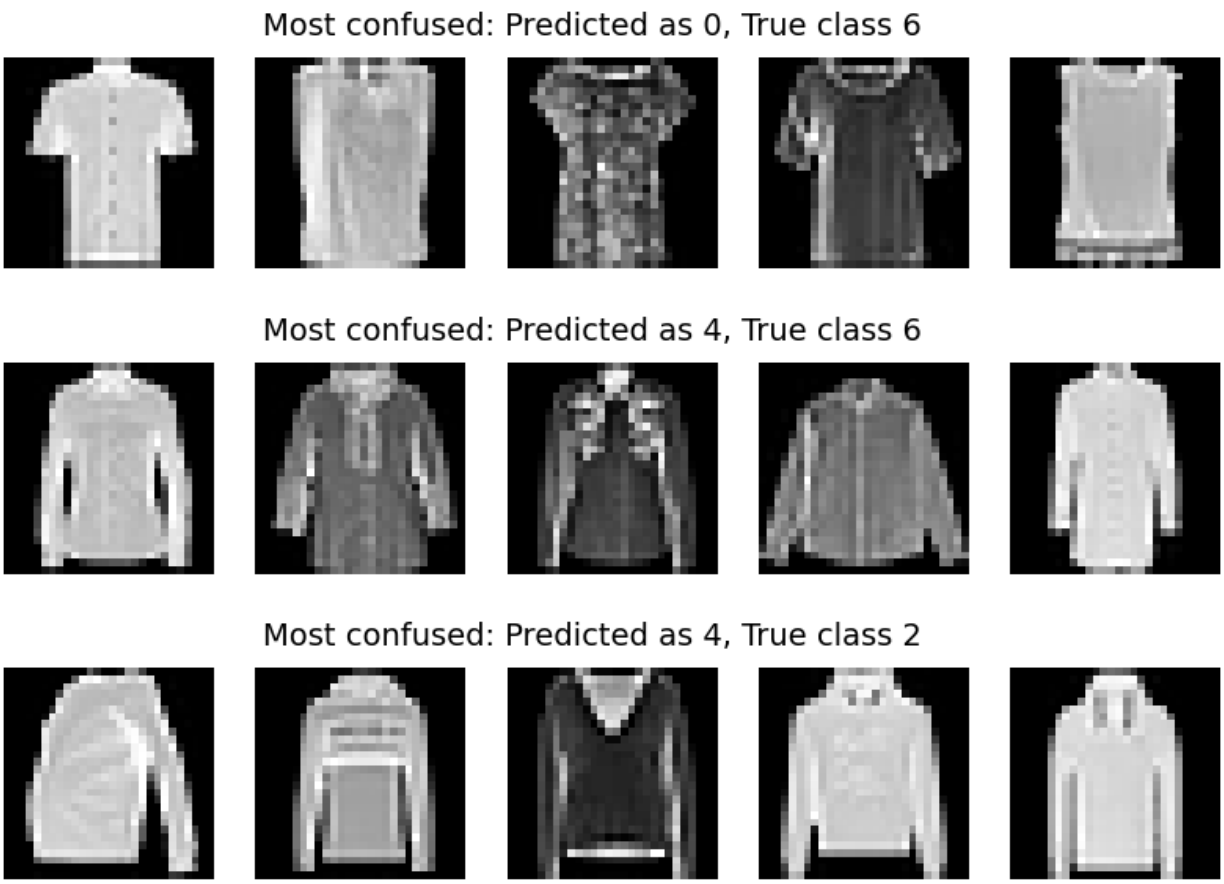

Predicted as 9, True class 7


Predicted as 3, True class 5



Fashion MNIST:





VII. Discussion

A.

1. The training has been performed and the train accuracy runtime and model size are in table 1.
2. Also in table 1
3. The values are in table 2 and 3, the observations from the values show that as the TH1 energy values are increasing, the runtime is decreasing so is the model size, which intern is giving a very low accuracy. It can be understood that lower the TH1 value, the better the performance.

B.

1. The values of the comparison are in table 4 and 5, from the table we can see that the performance of PixelHop++ decreases rapidly with the increase in TH2 value, PixelHop values also decrease but not as drastically as of PixelHop++ and PixelHop gave a better performance on both the datasets

2. The computational time for PixelHop is lower than that of PixelHop++ .and the model size of PixelHop is grater than the PixelHop++. As PixelHop++ uses cwSaab which eliminates useless features and only keeps the ones that are necessary.

C.

1. 2. For MNIST the lowest error rate is for class 1 and the highest error rate was for 5, class 9 has the lowest error rate and class 6 has the highest error rate. The most error classes are in the results above, and we see that these are wrongly classified because they have features which match with other classes.

   3. To improve accuracy of this model we can use Autoencoders and further enhance feature extraction by encoding it into a lower-dimensional space (latent space) and then reconstructing it back to its original form. This process forces the autoencoder to capture the most important features in the data, discarding noise and less useful information.