Project 3

Prameth Gaddale (pqg5273@psu.edu)

March 17, 2023

Abstract

In this project, deep neural networks have been used for solving the classification problem. Fully connected multi-layer perceptron models have been implemented for Taiji Keypose Classification using the PSU-TMM100 dataset with different number of features in both variants. Additionally, Wallpaper dataset has been used for image classification using the convolution neural networks.

Contents

1 Part 1

In Part-1 of the project, Taiji keypose classification task with the use of Multi-layer perceptron model for classification.

1.1 Introduction

A Multi-layer perceptron (MLP) is a type of artificial neural network that is most commonly used for classification tasks which involve higher levels of complexity in terms of large classification based datasets. Traditionally, many inefficient algorithms such as K-Nearest Neighbors and Fisher Classification algorithm were used for classification tasks. Here are the steps used for building a classification model with a standard multi layer perceptron:

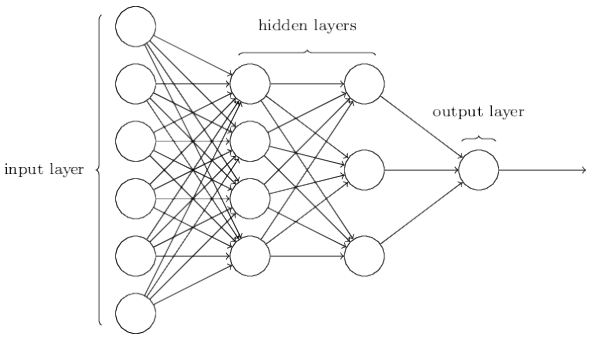


Figure 1: A Schematic of a MLP

- **Data preparation step:** The first step in this task of classification is to prepare the data for training the MLP. This involves dividing the data into training, validation, and test sets(ideally, but even train and test sets in two split strategy would work well), as well as normalizing the input data.

- **Building the MLP model:** The MLP model is built by defining the number of input neurons, hidden layers, and output neurons. The number of hidden layers and nodes in each layer are most important hyper-parameters that need to be chosen based on the complexity of the problem and the amount of data available.

- **Training the model:** The MLP model is trained using the training data fed into the PyTorch dataloader. During training, the weights and biases of the model are adjusted to minimize the error between the predicted and actual output. The training process involves forward propagation of input through the layers, followed by backward propagation of errors to adjust the weights.
- **Model evaluation:** The trained MLP model is evaluated using the validation set to check its performance. If the performance is not satisfactory, the hyper-parameters can be adjusted and later the model could be retrained with other set of hyperparameters tailored specifically for the algorithm.
- Testing the model: Finally, the performance of the MLP model is tested on the test set. The test set is used to evaluate the performance of the model on unseen data.
- Overall, the MLP model can be a powerful tool for classification tasks, as it can learn complex patterns in the data and make accurate predictions on new data. However, it is important to carefully choose the hyper-parameters and train the model on a large and representative dataset to achieve the best performance.

1.2 Data

- The given dataset is PSU-TMM100 (Taiji) based on human-sequence forms, rendered using 3D motion capture devices from various crucial body parts and foot pressure sensors.
- The 3D body joints from MoCAP captures 17 joints and the foot pressure data consists 1910 elements as shown in 2. The total number of features come out to be 1961 for each observation.

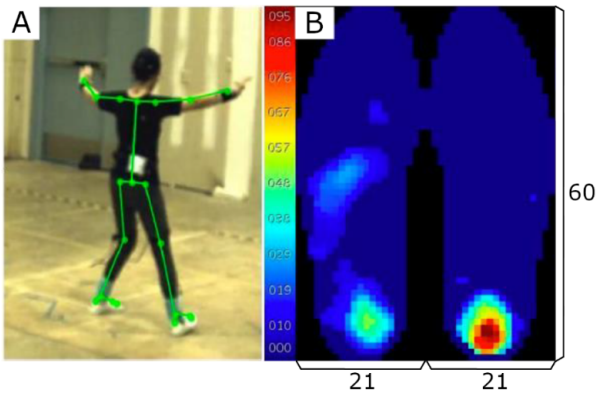


Figure 2: Taiji Dataset

Data Splitting Strategy

The training process involves the *Leave One Subject Out* strategy during classification to enable testing with pseudo-unseen data and resembles the prediction based off in a real scene setting.

1.3 Approach

- In an MLP, the input data is first fed into the input layer, which passes it through the network to produce an output. The intermediate layers (hidden layers) process the input data by applying a nonlinear transformation to the weighted sum of the inputs, while the output layer produces the final prediction or output.

- The MLP approach can be used for a variety of supervised learning tasks, including classification and regression. The network is trained using backpropagation, where the weights are adjusted during the training process to minimize the error between the predicted output and the actual output.

- One of the benefits of MLP is that it can learn complex nonlinear relationships between the input and output data. However, it can also be prone to overfitting if the number of hidden layers and neurons are too high or the training data is insufficient.

1.4 Baseline Model

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                [-1, 1024]       2,009,088
            Linear-2                [-1, 1024]       1,049,600
            Linear-3                  [-1, 46]          47,150
================================================================
Total params: 3,105,838
Trainable params: 3,105,838
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.02
Params size (MB): 11.85
Estimated Total Size (MB): 11.87
----------------------------------------------------------------
```

Figure 3: Baseline Model Configuration

- The baseline model used for classification was a simple 3 layer neural network consisting of a single hidden layer of 1024 neurons.

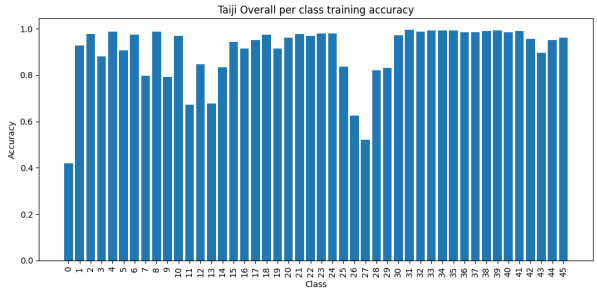
- There was no activation function used in the baseline model for training. In fact, the activation function is the main part of introducing the necessary non-linearity inside the machine learning model.

- If there is no activation function present in the model, the configuration would really correspond to that of a linear classification mode with matrices just got multiplied with each other.

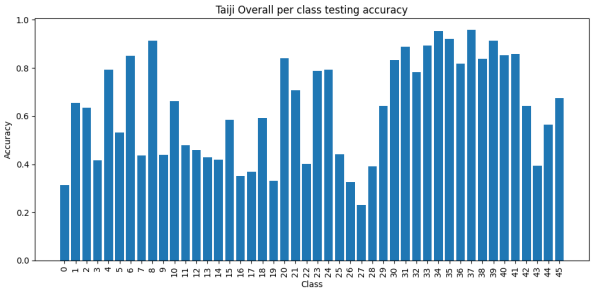
1.4.1 Full Configuration

Observations

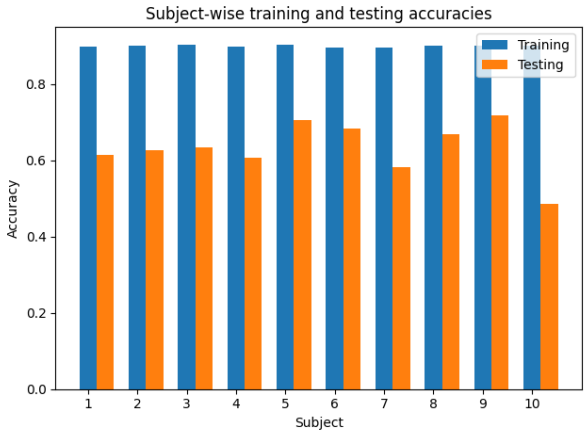
- The perclass training accuracy took a hit at certain areas due to linear nature of the model, and the complexity of the Taiji dataset was not properly modelled.
- Its more evident when we consider the per class test accuracy plot. There are various classes present which have the accuracy less than the reasonable measure of 0.4. In addition, the subject-wise train and test accuracy was also not really good with a huge gap between the train and test accuracy.
- Model training curve suggests that the model was trained completely to its potential, as the curve was almost straightening out.
- Coming to the confusion matrices, the test confusion matrix shows many errors suggesting the need for a better model in terms of complexity.



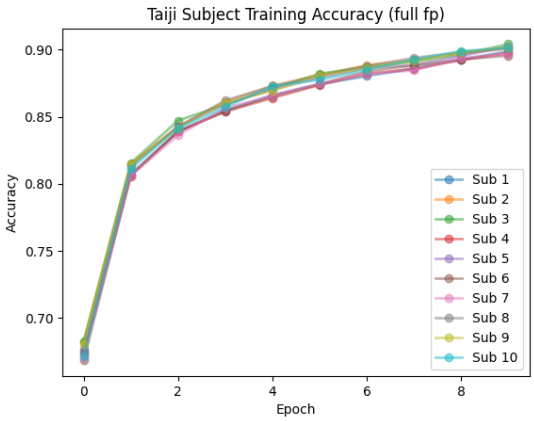
(a) Per-Class Train Accuracy



(b) Per-Class Test Accuracy



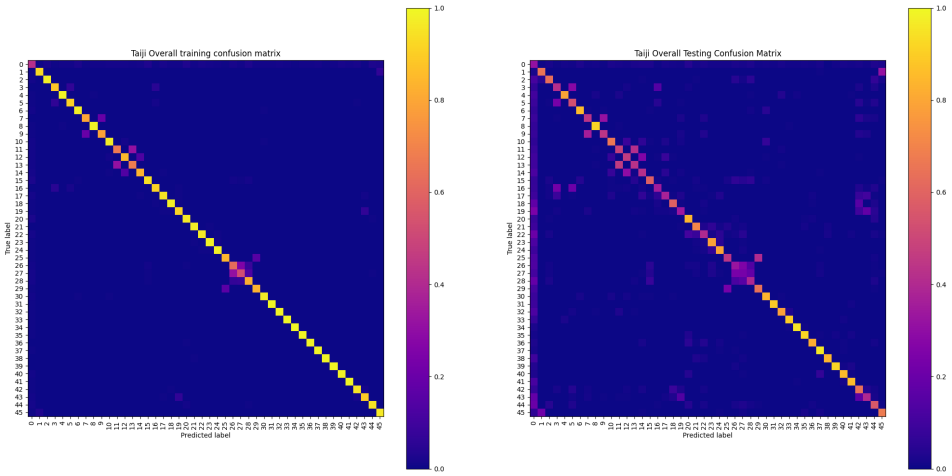
(c) Subject-Wise Train and Test Accuracy



(d) Model Training Curve

Figure 4: Baseline Model: Full Configuration

Confusion Matrices



(a) Train Confusion Matrix (b) Test Confusion Matrix

Figure 5: Baseline Model: Full Configuration Confusion Matrices

1.4.2 lod4 Configuration

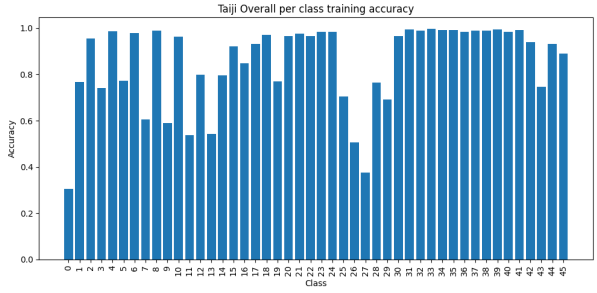
Observations

- The per-class training accuracy got a bit better compared to the full dataset configuration as the dataset compression allowed the performance of certain classes to be exemplified.

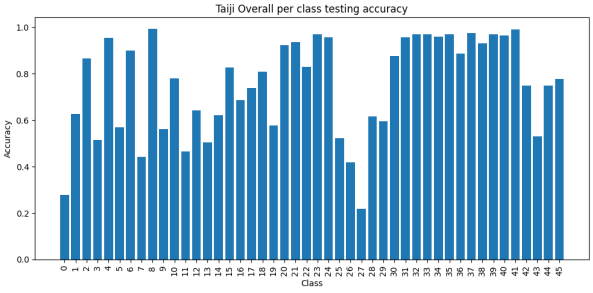
- However, there is a very poor performance in classes in range of 24-26 in the per-class test accuracy plot suggesting those features didn't really correspond well in the training set.

- Similarly, in this case Model training curve suggests that the model was trained completely for the complete 10 epochs.

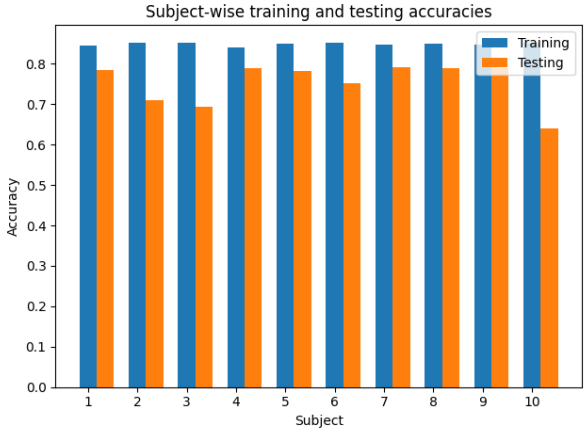
- Confusion matrices suggest a clear improvement in the upper section of the diagonal when compared visually with the performance achieved in the full configuration. The test confusion matrix still shows many errors in the lower part suggesting the need for a better model.



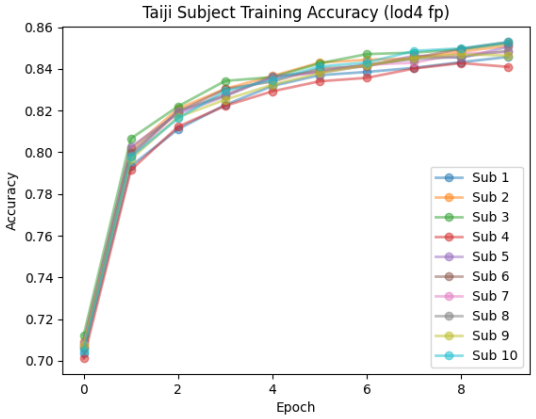
(a) Per-Class Train Accuracy



(b) Per-Class Test Accuracy



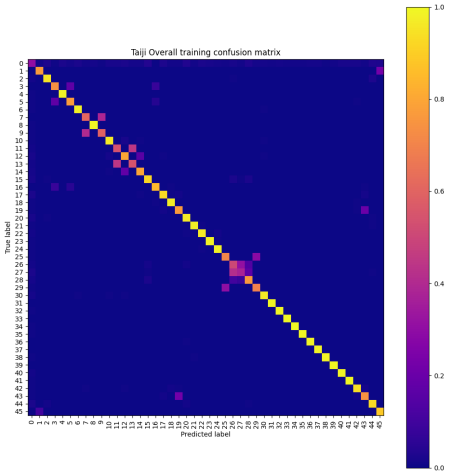
(c) Subject-Wise Train and Test Accuracy



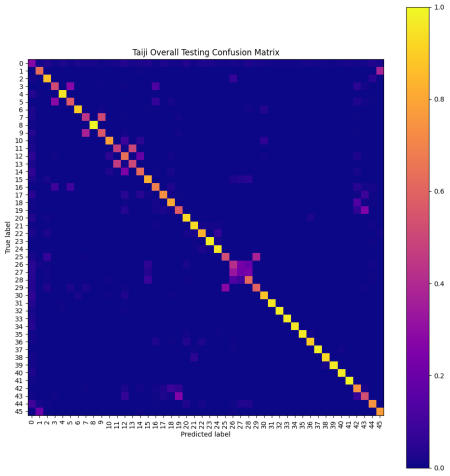
(d) Model Training Curve

Figure 6: Baseline Model: lod4 Configuration

Confusion Matrices



(a) Train Confusion Matrix



(b) Test Confusion Matrix

Figure 7: Baseline Model: lod4 Configuration Confusion Matrices

1.5 Improved Model

Layer (type)	Output Shape	Param #
Linear-1	[-1, 1024]	2,009,088
BatchNorm1d-2	[-1, 1024]	0
Dropout-3	[-1, 1024]	0
Linear-4	[-1, 1024]	1,049,600
BatchNorm1d-5	[-1, 1024]	0
Linear-6	[-1, 46]	47,150

Total params: 3,105,838
Trainable params: 3,105,838
Non-trainable params: 0

Input size (MB): 0.01
Forward/backward pass size (MB): 0.04
Params size (MB): 11.85
Estimated Total Size (MB): 11.89

(a) Improved Model: Full

Layer (type)	Output Shape	Param #
Linear-1	[-1, 512]	34,816
BatchNorm1d-2	[-1, 512]	0
Dropout-3	[-1, 512]	0
Linear-4	[-1, 256]	131,328
BatchNorm1d-5	[-1, 256]	0
Linear-6	[-1, 46]	11,822

Total params: 177,966
Trainable params: 177,966
Non-trainable params: 0

Input size (MB): 0.00
Forward/backward pass size (MB): 0.02
Params size (MB): 0.68
Estimated Total Size (MB): 0.70

(b) Improved Model: lod4

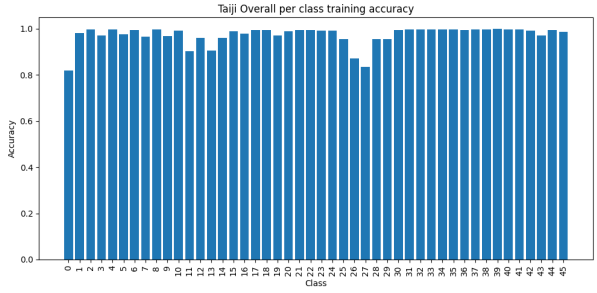
Figure 8: Improved Models

- Two different model configuration have been built for the classification task.
- They both differ in the slighter ways accommodating the right amount of hidden layer units used in the 3-layer neural network configuration.
- The model also consist of batch normalization and dropout layers, which are covered more in depth in the Part-2. In a nutshell, they both tend to curb the over-fitting issue faced by our model.

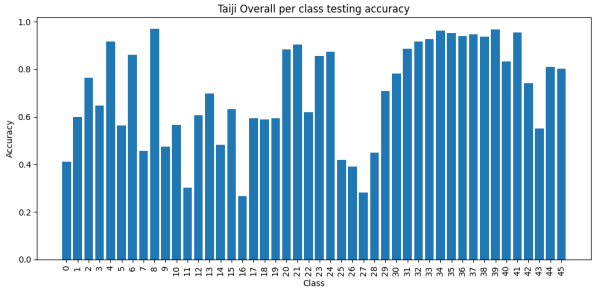
1.5.1 Full Configuration

Observations

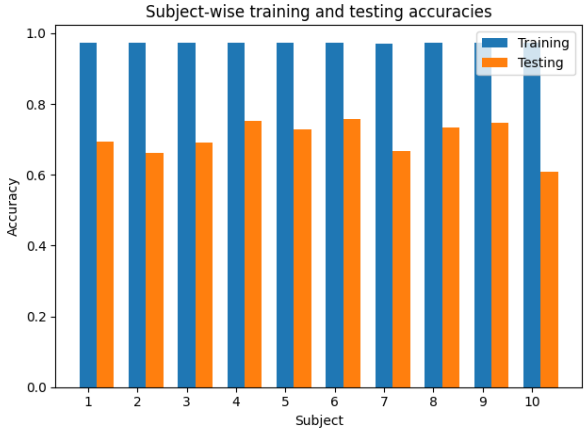
- The training per-class accuracy is really impressive compared to the baseline model, suggesting good amount of features have been trained by the model.
- Also, having the batch normalization layers with the ReLU activation function induced the much wanted model complexity.
- However, the same thing couldn't be said to the test per-class accuracy. The test accuracy seems to be same(low) for certain classes, but there is serious improvement in the class ranges 30-40.
- Lower gap in the subject-wise train and test accuracy suggests a lower over-fitting response.
- Model training curve suggests the model was trained for 25 epochs with the curve straightening out.
- The train confusion matrix has gotten better. Also in the case for its test-counterpart, we can see a clear improvement in the lower upper section of the diagonal element classes being correctly classified.



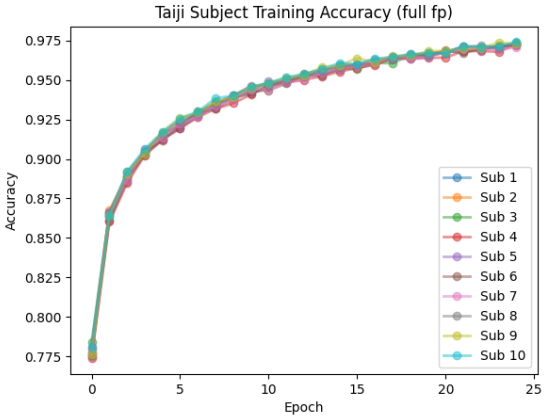
(a) Per-Class Train Accuracy



(b) Per-Class Test Accuracy



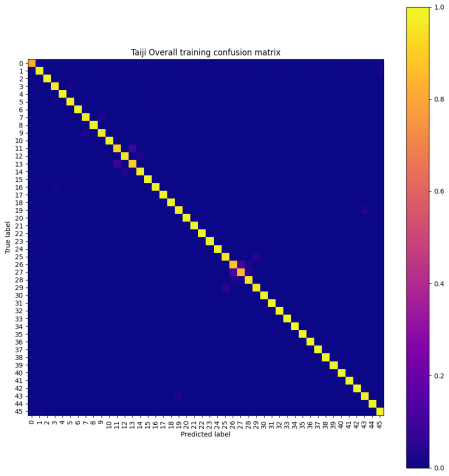
(c) Subject-Wise Train and Test Accuracy



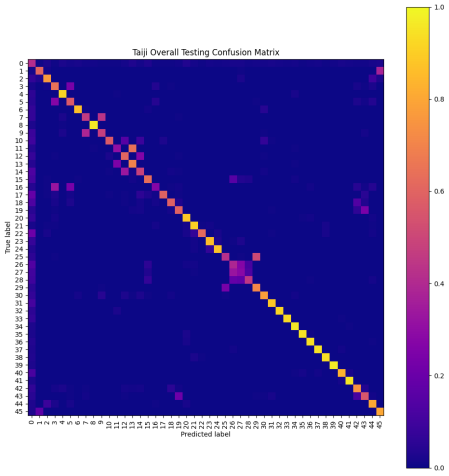
(d) Model Training Curve

Figure 9: Baseline Model: lod4 Configuration

Confusion Matrices



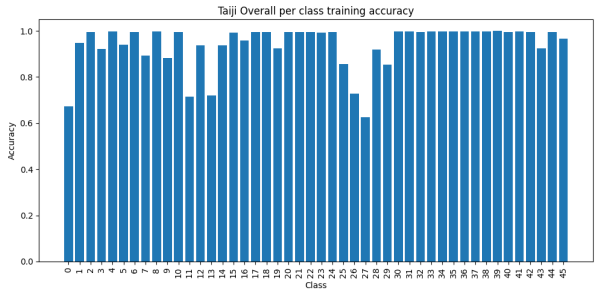
(a) Train Confusion Matrix



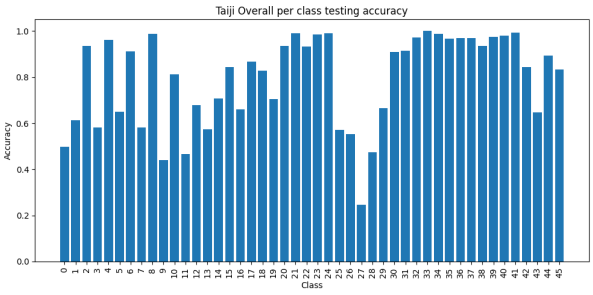
(b) Test Confusion Matrix

Figure 10: Improved Model: Full Configuration Confusion Matrices

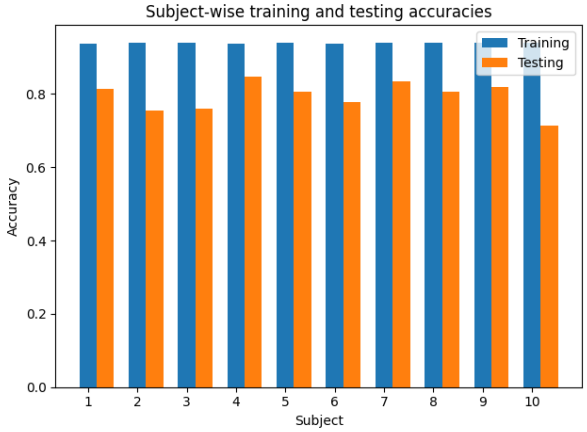
1.5.2 lod4 Configuration



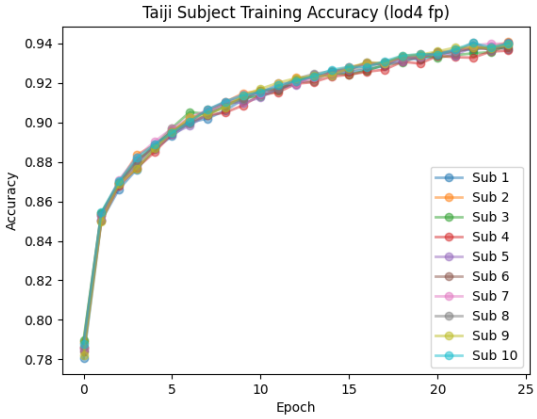
(a) Per-Class Train Accuracy



(b) Per-Class Test Accuracy



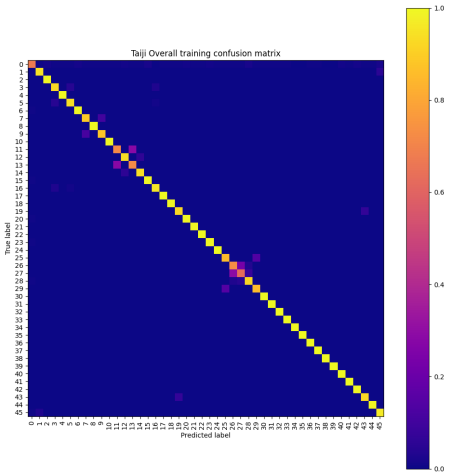
(c) Subject-Wise Train and Test Accuracy



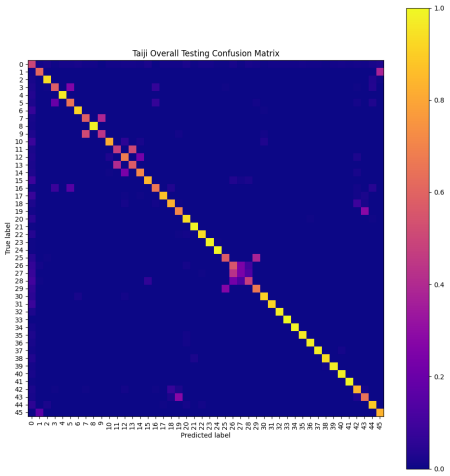
(d) Model Training Curve

Figure 11: Baseline Model: lod4 Configuration

Confusion Matrices



(a) Train Confusion Matrix



(b) Test Confusion Matrix

Figure 12: Improved Model: lod4 Configuration Confusion Matrices

Observations

- The improved model has lot more improvement in terms of sustained performance in the lod4 variant of the featured dataset compared to the efficiency obtained when the model was trained using the full configuration.
- The train and test per-class accuracy have gotten lot better compared to the previous baseline results indicating the robust performance and need for relevant features.
- The over-fitting has reduced quite a lot too, which is evident from the subject-wise train and test accuracy plot.
- Model training curve suggests all the subjects were trained around the same level with good training accuracy over 90
- Confusion matrices have gotten a lot denser around the diagonal elements suggesting an improvement in the classification rates.

1.6 Analysis

Model	Feature Variant	Train Accuracy	Test Accuracy	Standard Deviation Train	Standard Deviation Test
Baseline	full	0.90008	0.63266	0.00259	0.06459
Improved	full	0.97273	0.70394	0.0009	0.04586
Baseline	lod4	0.84899	0.75189	0.00361	0.05008
Improved	lod4	0.93901	0.79305	0.0014	0.03871

Table 1: Taiji Dataset Model Analysis

- The baseline model was found to be not very robust to the features equipped by the Taiji keypose dataset used for the classification tasks.
- The improved model consisted of batch normalization layers, dropout and ReLU activation function which induced the necessary model complexity.
- Looking at the Table 1, there is clear improvement in every single category being quantified.
- Especially, there is improvement observed in the standard deviation over various subjects in the training and test accuracies.
- Suggesting that the model is more robust and not very sensitive to small changes in the dataset variation through the LOSO strategy.

2 Part 2

2.1 Introduction

- Image classification is a process of categorizing/labelling/characterizing a given image based on the content present. It is one of the most important computer vision tasks that involves analyzing an image and assigning it to a specific category from the given class distribution. The goal of image classification is to automate the process of recognizing and detect the presence objects or patterns within an image, and to accurately and quickly assign them to a specific category [1].

- To perform image classification, a computer vision algorithm is trained on a dataset of labeled images. The algorithm uses various machine learning or representation learning techniques to learn the patterns and features that distinguish one category from another. Once the algorithm is trained on sufficient data, it can be used to classify new, unlabeled images called simply the Test set.

- Image classification has numerous applications, including facial recognition, object detection, and medical imaging. It is used in a variety of fields, including healthcare, security, and manufacturing, to improve efficiency and accuracy in various tasks.

2.2 Data

Wallpaper dataset consists of images containing the 17 Wallpaper Group. The wallpaper groups are mentioned as [2]:

- A wallpaper is a mathematical object covering a whole Euclidean plane by repeating a recurring theme indefinitely, in manner that certain a isometry being followed is kept unchanged.

- To a given wallpaper there usually is a corresponding group of congruent transformations, with the functional composition as the group operation.

- Thus, a wallpaper group (or plane symmetry group or plane crystallographic group) in general terms is a mathematical classification of a two-dimensional repetitive pattern, based on the symmetries in the pattern.

In general creating a symmetry based dataset really entails to:

- The process of classifying images based on their degree of symmetry over a speific direction or axis being specified.

- The symmetry dataset consists of images that have been artificially generated to exhibit different levels of symmetry, ranging from highly symmetric to completely asymmetric categories.

- To classify images in the symmetry dataset, a computer vision algorithm is generally trained using a large set of labeled images.

- The algorithm then tries to learn to detect and analyze these different types of symmetry encoded in the images, such as bilateral, radial, or translational symmetry.

- The algorithm is then tested on a set of validation images to determine its accuracy in classifying images based on their symmetry.

- The classification of symmetry in images has various applications, including image recognition, robotics, and manufacturing.

- For example, in robotics, it is important to recognize the symmetrical features of an object in order to perform accurate grasping and manipulation.

- In manufacturing, symmetry classification is used to ensure the consistency and quality of products during production.

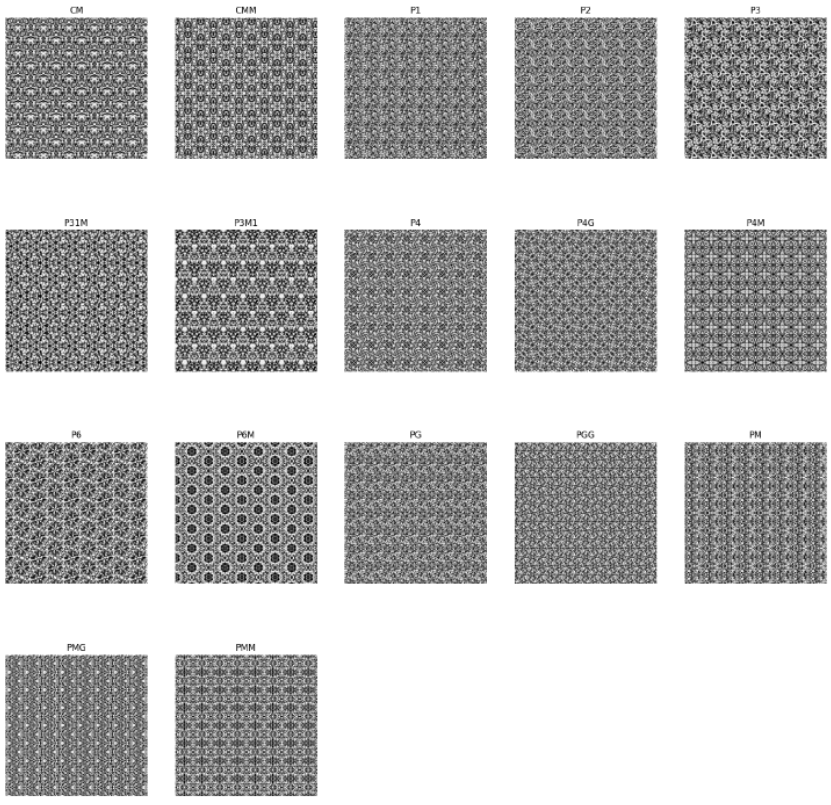


Figure 13: Wallpaper Dataset

2.3 Approach

The approach in this part of the project is to be having to deploy convolutional neural networks for image classification on the Wallpaper dataset. The high-level detailed explanation of the approach taken is mentioned as follows:

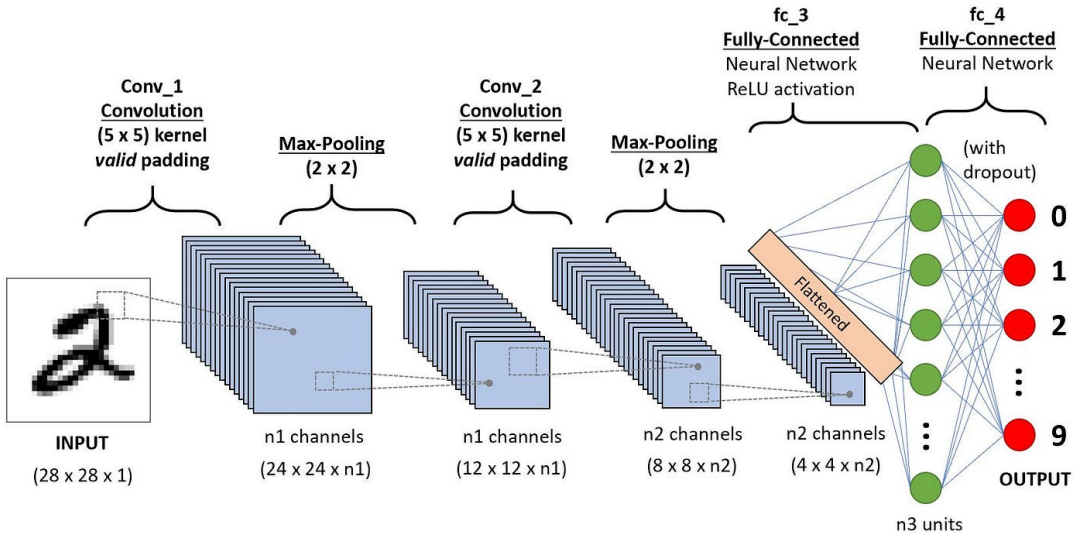
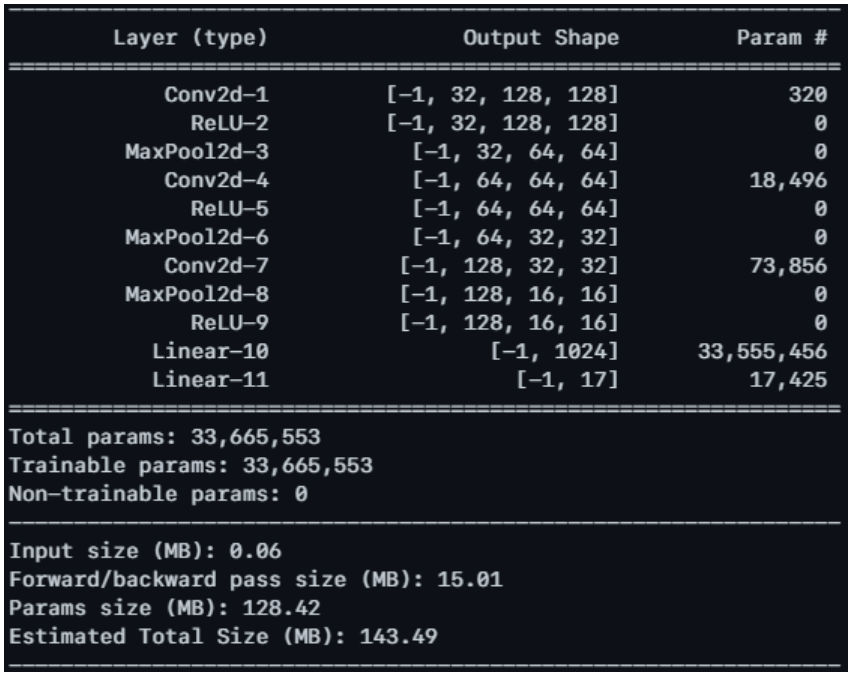


Figure 14: A Typical Model Architecture for Image Classification based on CNNs

- The basic architecture of a CNN (as shown in 14) consists of a series of convolutional layers, followed by one or more fully connected layers(also termed as Linear layers in PyTorch). The convolutional layers use filters/matrix-sized weights to extract features from the input image, and each filter produces a feature map that highlights a specific pattern in the image.
- Additionally, the fully connected layers then employ these learnt feature maps to classify the image into one of several possible categories.
- To train a CNN for image classification, a large dataset of correctly labeled images is required. The CNN is first initialized with random weights, and the training process involves iteratively adjusting these weights in order to minimize a certain loss function(such as Mean Squared Error or Categorical Cross Entropy) that measures the difference between the predicted class and the true class of each given training image to the model initialized.
- The weights are then adjusted using an optimization algorithm such as stochastic gradient descent or Adam(a more modern variant), which updates the weights based on the gradient of the loss function with respect to the weights.
- Once the CNN is trained, it can be used to classify new, unlabeled images which is also termed as the test set. The input image is fed into the CNN, which applies the learned filters to extract features from the image.
- The fully connected layers then use these features to compute a probability distribution over the possible categories, and the category with the highest probability is assigned as the predicted class for the fed image.

2.4 Baseline Model

The baseline model used for analysis is given as follows:



Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 128, 128]	320
ReLU-2	[-1, 32, 128, 128]	0
MaxPool2d-3	[-1, 32, 64, 64]	0
Conv2d-4	[-1, 64, 64, 64]	18,496
ReLU-5	[-1, 64, 64, 64]	0
MaxPool2d-6	[-1, 64, 32, 32]	0
Conv2d-7	[-1, 128, 32, 32]	73,856
MaxPool2d-8	[-1, 128, 16, 16]	0
ReLU-9	[-1, 128, 16, 16]	0
Linear-10	[-1, 1024]	33,555,456
Linear-11	[-1, 17]	17,425

Total params: 33,665,553
Trainable params: 33,665,553
Non-trainable params: 0

Input size (MB): 0.06
Forward/backward pass size (MB): 15.01
Params size (MB): 128.42
Estimated Total Size (MB): 143.49

Figure 15: Baseline Model Configuration: Wallpaper Dataset

- The model consists of 3 convolutions layers which are preserving the image size through the use of same-zero-padding strategy. The use of ReLU non-linearity with MaxPool layers in between the each convolution layer.

- The 2 fully connected layers are used in the end of the convolution layers to get the classification results through the use of softmax layer after the full connected layers.

2.4.1 Test Set

Observations

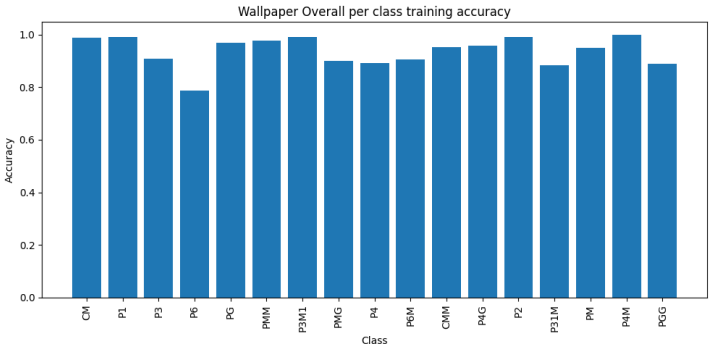
- The per-class train accuracy is reasonable except on one of the classes: P6 where it has dropped lower than 80%. Except that certain class, all others have performed good enough to get a good measure of accuracy on the train set. The test per-class accuracy is also on the average end, with certain classes dropping to as low as 70%.

- The model training curve suggests good performance in terms of training accuracy, and has been straightened out at the end suggesting good enough training level.

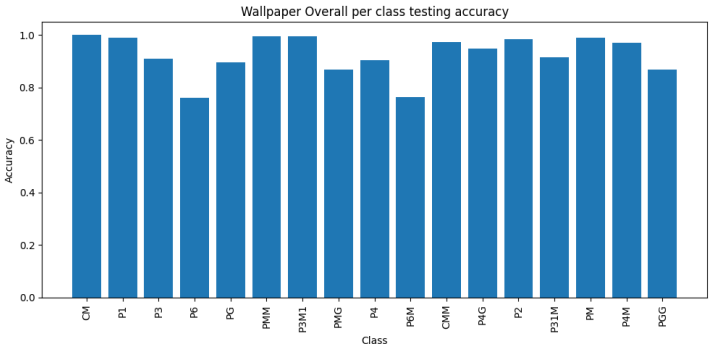
- The confusion matrices of both train and test sets are good enough, with the test matrix having certain mis-classifications for certain classes. However, on the overall side the baseline model without any serious implementation has performed well.

- Layer-1 of the model has been visualized. Each image from one of the classes has been fed into the model and the feature map of the layer-1 activation has been found.

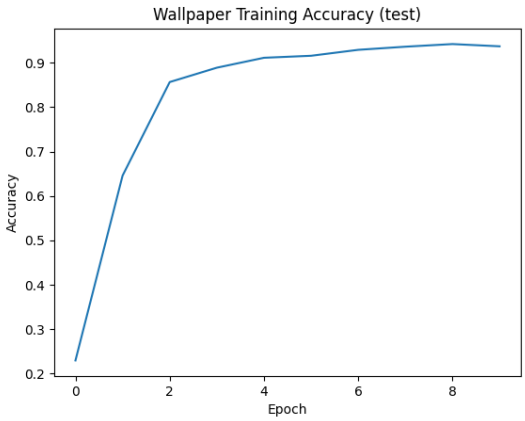
- The TSNE visualization show good performance on the training set as most of the classes got spread out in different directions, also a reasonable performance on the test set has been achieved, however there are a few overlaps being observed in certain categories.



(a) Per-Class Train Accuracy



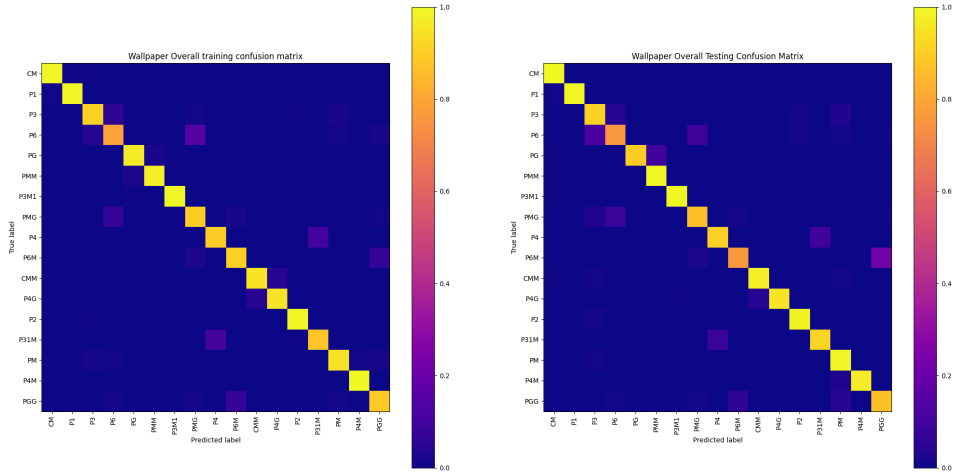
(b) Per-Class Test Accuracy



(c) Model Test Accuracy vs Epochs

Figure 16: Baseline Model: Wallpaper(test)

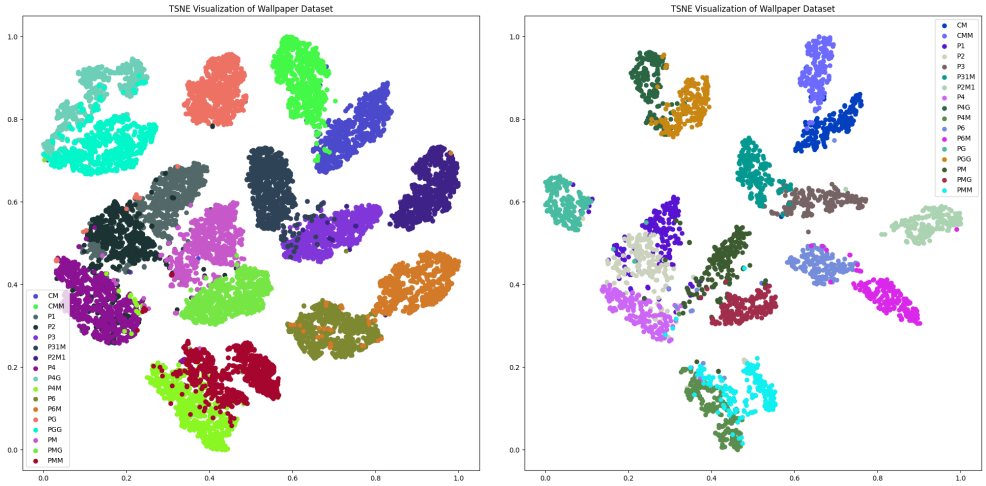
Confusion Matrices



(a) Train Confusion Matrix (b) Test Confusion Matrix

Figure 17: Baseline Model Confusion Matrices: Wallpaper(test)

T-SNE Visualization



(a) TSNE: Train Set (b) TSNE: Test Set

Figure 18: Baseline Model TSNE Visualization: Wallpaper(test)

Layer 1 Visualization

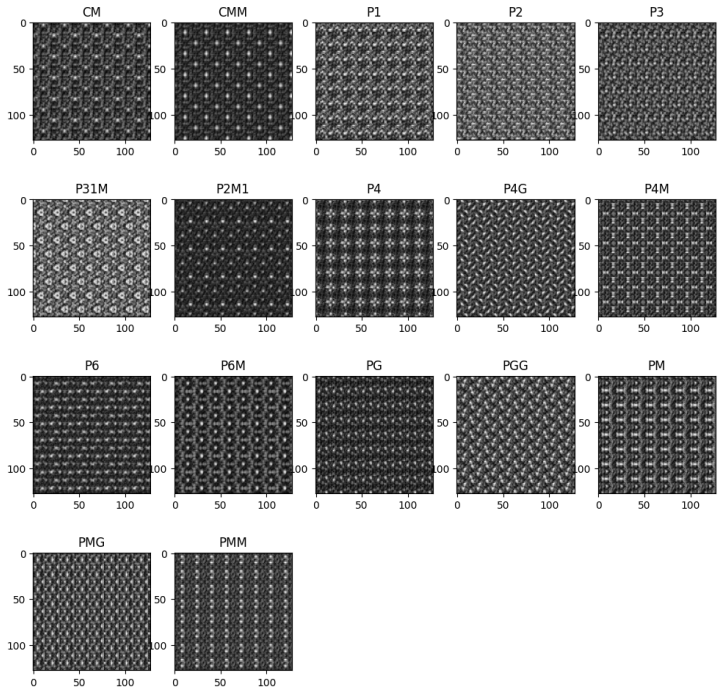


Figure 19: Baseline Model Layer 1 Visualization: test

2.4.2 Test Challenge Set

Observations

- The per-class train accuracy is similar to the one observed in the last case. Except that, certain class, good performance is observed on the training set. The test per-class accuracy is not good, as the challenge variant of the test set is not that similar to the normal test set.

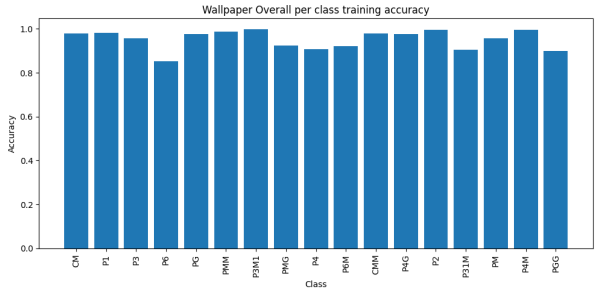
- There are only certian classes on which the performance of the model is good. Classes P3, P31M have gotten good test accuracy compared to the rest of them.

- The model training curve suggests good performance in terms of training accuracy as expected, and has been straightened out at the end suggesting good enough training level.

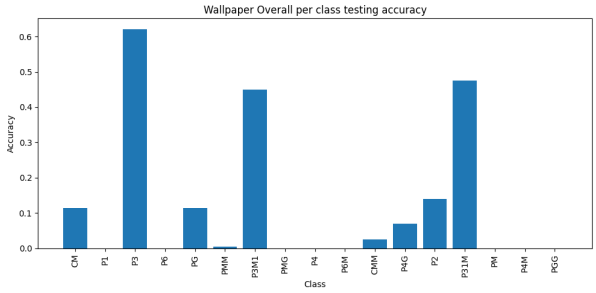
- The confusion matrices of both train is good, but the one on the test set is horrible. There is no correspondence with the diagonal elements at all. The baseline model needs serious improvements in the case of model generalizability.

- Layer-1 of the model has been visualized from the test loader. Each image from one of the classes has been fed into the model and the feature map of the layer-1 activation has been found. Its evident that model is looking out for symmetric based projections.

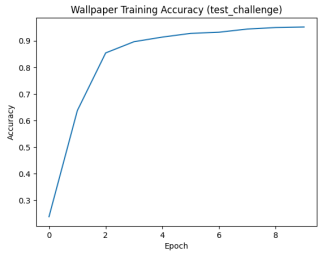
- The TSNE visualization show good performance on the training set as most of the classes got spread out in different directions. But as expected, the test set TSNE has elements all over the place.



(a) Per-Class Train Accuracy



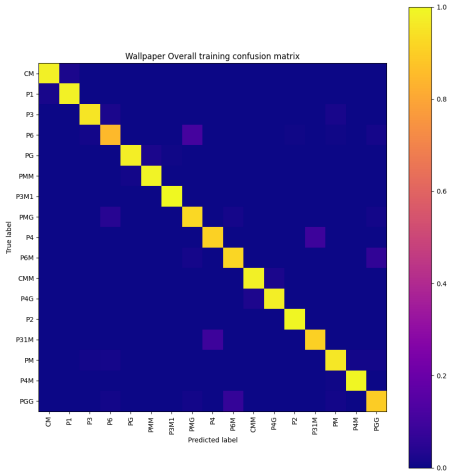
(b) Per-Class Test Accuracy



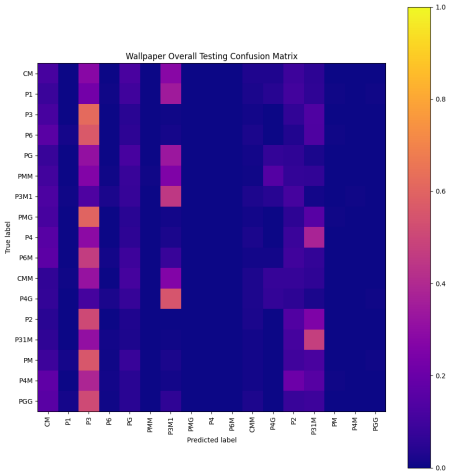
(c) Model Test Accuracy vs Epochs

Figure 20: Baseline Model: Wallpaper(test_challenge)

Confusion Matrices



(a) Train Confusion Matrix



(b) Test Confusion Matrix

Figure 21: Baseline Model Confusion Matrices: Wallpaper(test_challenge)

Layer 1 Visualization

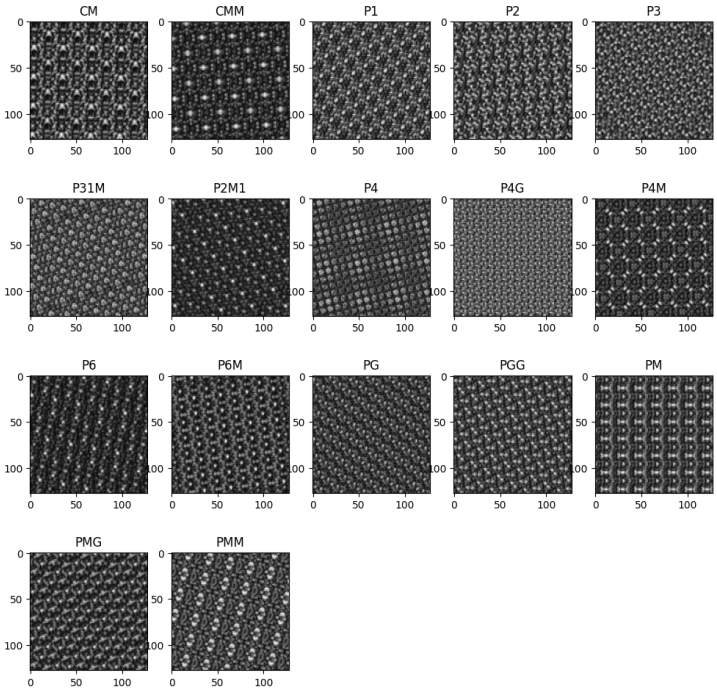
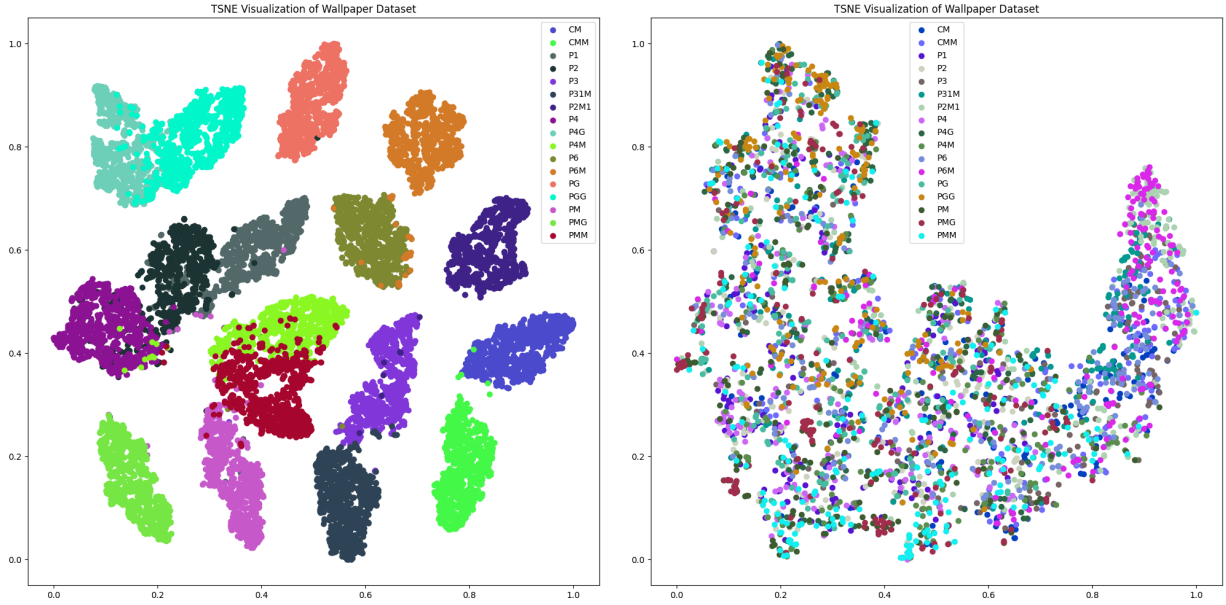


Figure 22: Baseline Model Layer 1 Visualization: test_challenge

T-SNE Visualization



(a) TSNE: Train Set (b) TSNE: Test Set

Figure 23: Baseline Model TSNE Visualization: Wallpaper(test_challenge)

2.5 Improved Model

In addition to the convolutional neural networks used in the baseline model, the improved model implementation used batch normalization and dropout which are considered crucial layer for general image classification tasks. Batch normalization and dropout are two regularization techniques commonly used in deep learning models for image classification, including convolutional neural networks (CNNs).

2.5.1 Batch Normalization and Dropout

- Batch normalization is a technique that normalizes the inputs to each layer in a CNN. It works by standardizing the mean and variance of the input to each layer, which can help improve the stability and efficiency of the model during training.

- This technique has been shown to improve the accuracy and speed of CNNs in various image classification tasks.

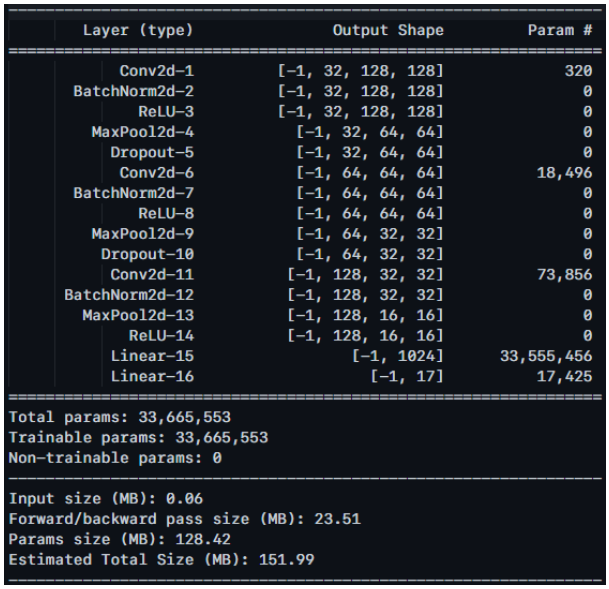
- The basic idea of batch normalization is to normalize the activations of each layer by subtracting the batch mean and dividing by the batch standard deviation. This helps to prevent the model from becoming too sensitive to small changes in the input data, which can lead to overfitting.

- Dropout is another regularization technique that helps prevent overfitting in deep learning models, including CNNs.

- Dropout works by randomly dropping out some of the neurons in the model during training, forcing the model to learn more robust features that are less dependent on specific input neurons. This technique has been shown to improve the accuracy and generalization performance of CNNs in various image classification tasks.

- During training, dropout is applied by randomly dropping out a fraction of the neurons in each layer of the CNN. This fraction is typically set to a small value, such as 0.25, and can be adjusted to control the strength of regularization.

- During inference however, all neurons are used, but their outputs are scaled by the dropout probability to ensure that the model behaves the same way during training and inference.



Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 128, 128]	320
BatchNorm2d-2	[-1, 32, 128, 128]	0
ReLU-3	[-1, 32, 128, 128]	0
MaxPool2d-4	[-1, 32, 64, 64]	0
Dropout-5	[-1, 32, 64, 64]	0
Conv2d-6	[-1, 64, 64, 64]	18,496
BatchNorm2d-7	[-1, 64, 64, 64]	0
ReLU-8	[-1, 64, 64, 64]	0
MaxPool2d-9	[-1, 64, 32, 32]	0
Dropout-10	[-1, 64, 32, 32]	0
Conv2d-11	[-1, 128, 32, 32]	73,856
BatchNorm2d-12	[-1, 128, 32, 32]	0
MaxPool2d-13	[-1, 128, 16, 16]	0
ReLU-14	[-1, 128, 16, 16]	0
Linear-15	[-1, 1024]	33,555,456
Linear-16	[-1, 17]	17,425

Total params: 33,665,553
Trainable params: 33,665,553
Non-trainable params: 0

Input size (MB): 0.06
Forward/backward pass size (MB): 23.51
Params size (MB): 128.42
Estimated Total Size (MB): 151.99

Figure 24: Improved Model Configuration: Wallpaper Dataset

2.5.2 Test Set

Observations

- Compared to the baseline model, this improvements have paid off in the accuracy in both train and test accuracy quantitatively. The per-class train accuracy and test accuracy is much better than the baseline measures.

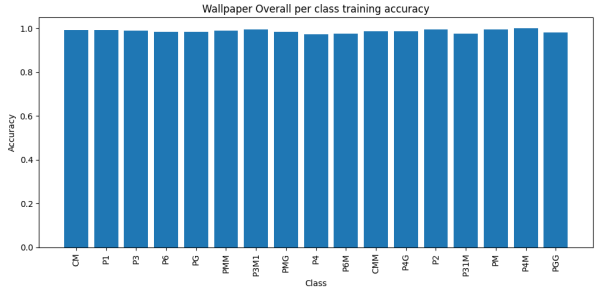
- There is only a certain class: P4G, where the test accuracy has dropped 80%, other than that all the classes have really well.

- The model training curve suggests good performance in terms of training accuracy as expected, and has been straightened out at the end suggesting good enough training level. Also, running the model with more epochs has helped in training the model for better performance.

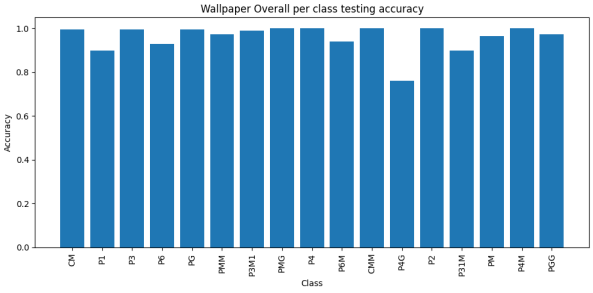
- The performance from the per class plots corresponds, continues to the confusion matrices. The training confusion matrix is clearly very god, with very less elements in the off-diagonal elements.

- Layer-1 of the model has been visualized from the test loader. Each image are distinct enough and has suggested the filters learnt smaller, minor details.

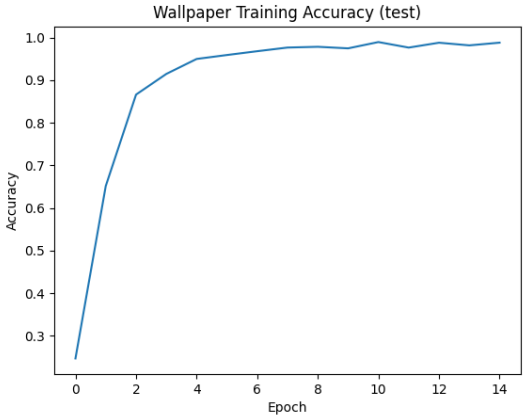
- The TSNE visualization show good performance on both the train and test sets. All the classes have spread out really well. In the test set, there is a overlap at a certain extent but there's good improvement over the baseline model results.



(a) Per-Class Train Accuracy



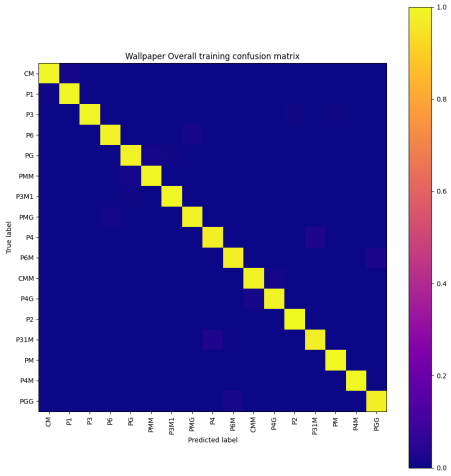
(b) Per-Class Test Accuracy



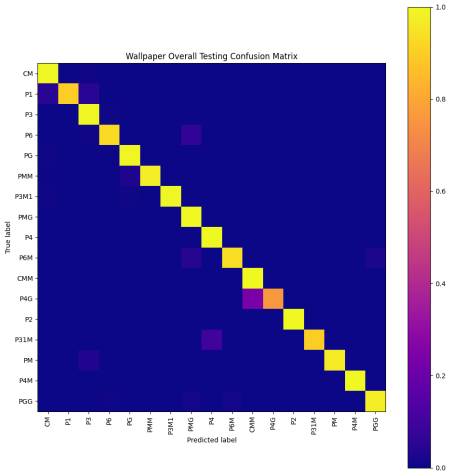
(c) Model Test Accuracy vs Epochs

Figure 25: Improved Model: Wallpaper(test)

Confusion Matrices



(a) Train Confusion Matrix



(b) Test Confusion Matrix

Figure 26: Improved Model Confusion Matrices: Wallpaper(test)

Layer 1 Visualization

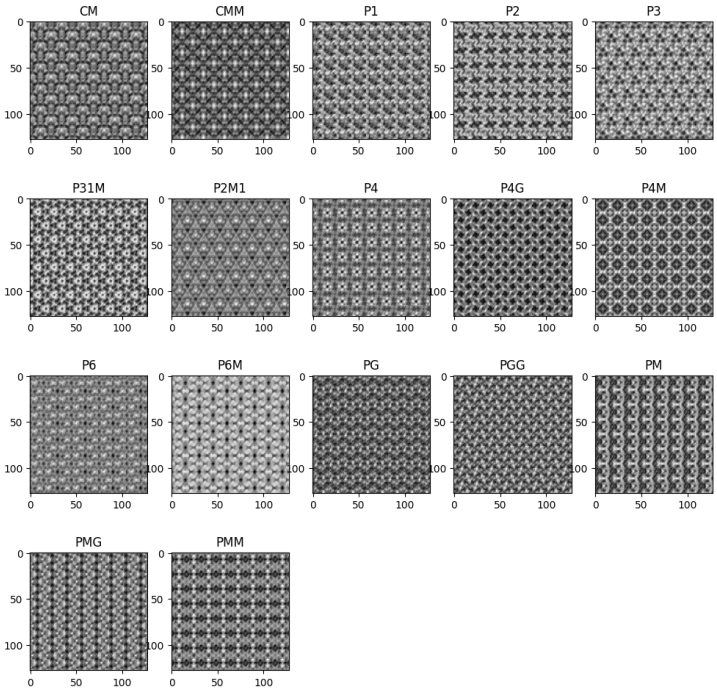
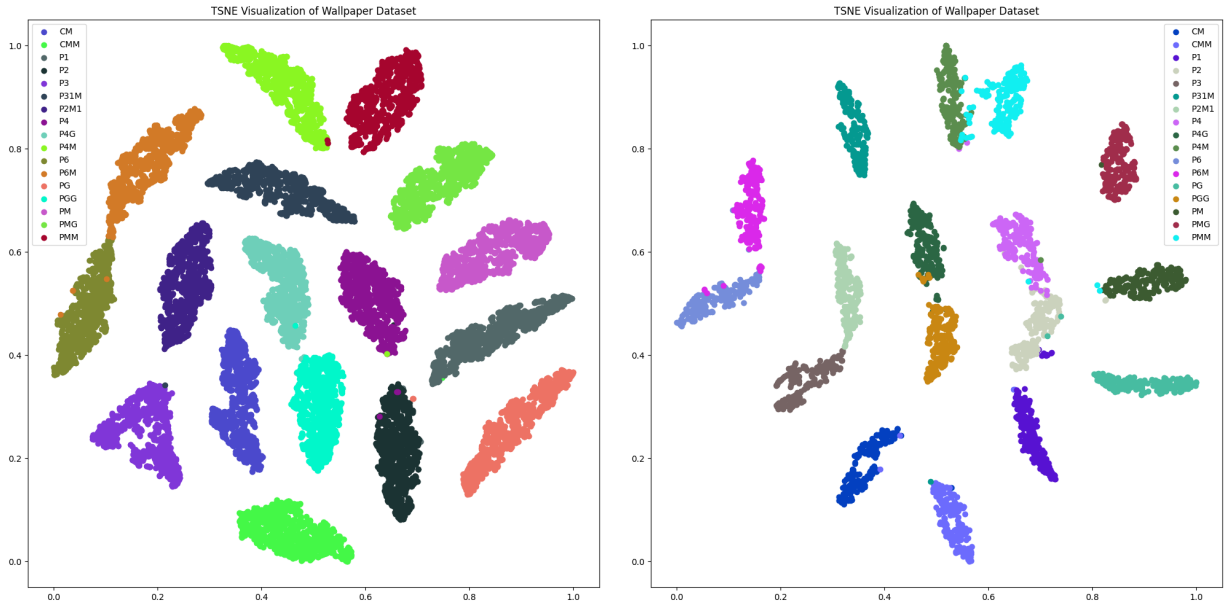


Figure 27: Improved Model Layer 1 Visualization: test

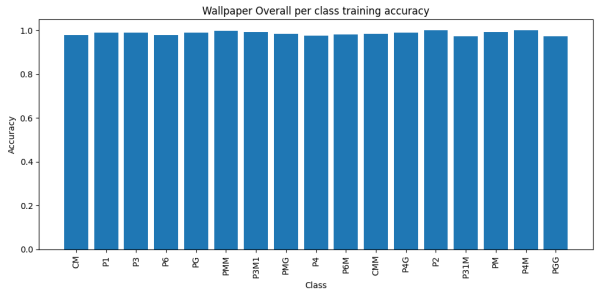
T-SNE Visualization



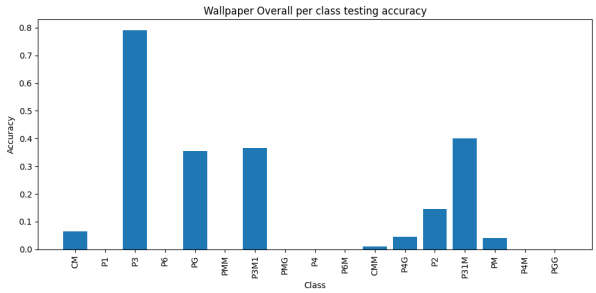
(a) TSNE: Train Set (b) TSNE: Test Set

Figure 28: Improved Model TSNE Visualization: Wallpaper(test)

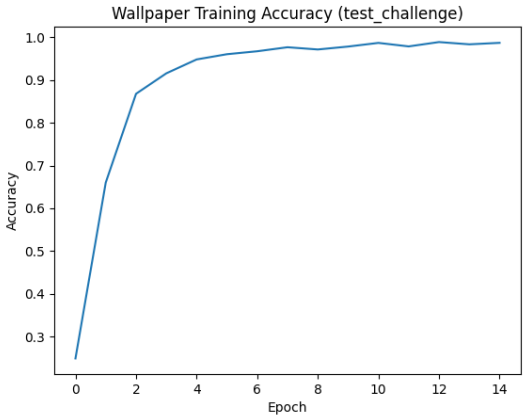
2.5.3 Test Challenge Set



(a) Per-Class Train Accuracy



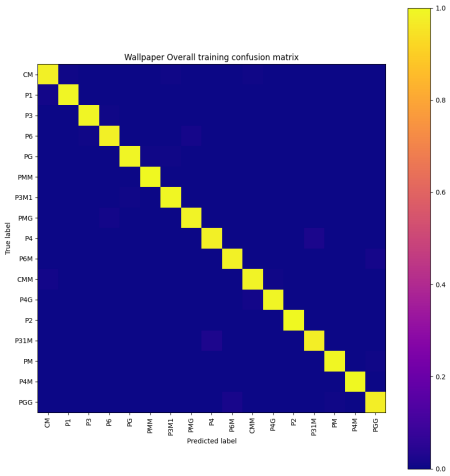
(b) Per-Class Test Accuracy



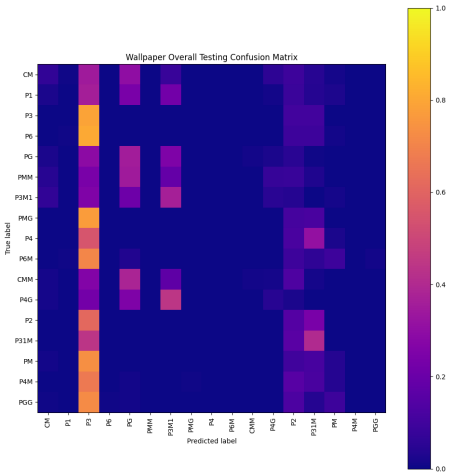
(c) Model Test Accuracy vs Epochs

Figure 29: Improved Model: Wallpaper(test_challenge)

Confusion Matrices



(a) Train Confusion Matrix



(b) Test Confusion Matrix

Figure 30: Improved Model Confusion Matrices: Wallpaper(test_challenge)

Layer 1 Visualization

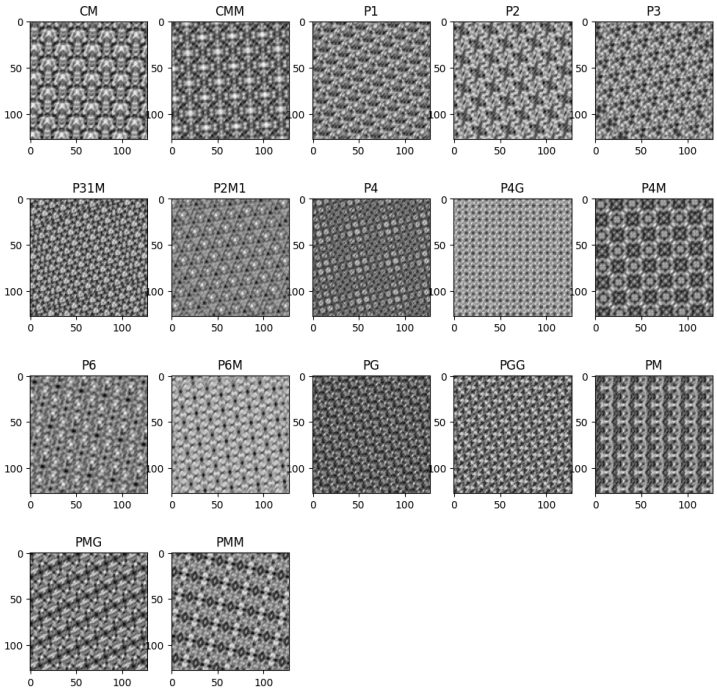
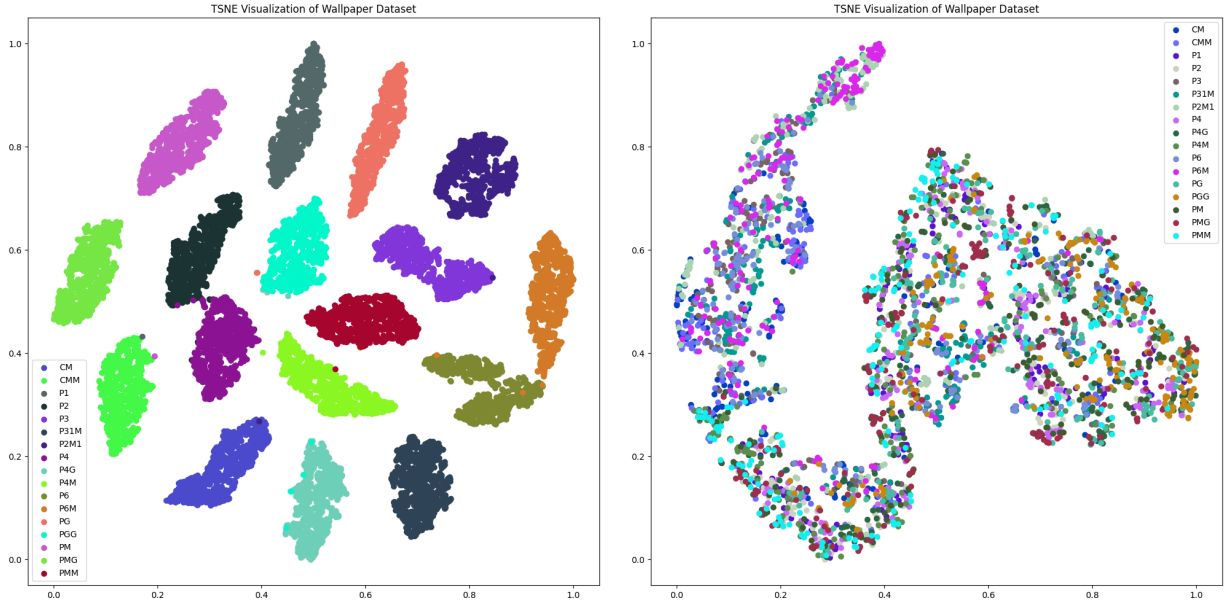


Figure 31: Improved Model Layer 1 Visualization: test_challenge

T-SNE Visualization



(a) TSNE: Train Set (b) TSNE: Test Set

Figure 32: Improved Model TSNE Visualization: Wallpaper(test_challenge)

Observations

- In terms of the test challenge set, the randomness and stark contrast in the data distribution are the main causes in the decrease in the test set performance. The performance on the test set however, is better than the one observed in the case of the baseline model.
- The model training curve suggests good performance in terms of training accuracy as expected, and has been straightened out at the end suggesting good enough training level. Also, running the model with more epochs has helped in training the model for better performance.
- The performance in the per-class test accuracy suggest that each class has been exemplified by training on the improved model compared to the basleine model.
- Layer-1 of the model has been visualized from the test loader. Each image are distinct enough and has suggested the filters learnt smaller, minor details.
- Confusion matrices of the train is very clear in the off-diagonal elements, however the test confusion is better than the previous model but still not that good.
- The TSNE visualization show good performance on both the training set. All the classes have spread out really well. In the test set, there is a overlap at a higher extent but there's good improvement over the baseline model results.

2.6 Analysis

Model	Test Set	Train Accuracy	Test Accuracy	Standard Deviation Train	Standard Deviation Test
Baseline	test	93.712	92.588	0.05456	0.07401
Improved	test	98.794	96.0	0.00736	0.06017
Baseline	test_challenge	95.224	11.853	0.04155	0.19179
Improved	test_challenge	98.706	13.029	0.00834	0.21471

Table 2: Wallpaper Dataset Model Analysis

- Having the model improved, performance in every category has been showing good improvements over the baseline results.
- In summary, both batch normalization and dropout are effective techniques for improving the accuracy and generalization performance of deep learning models, including CNNs, in image classification tasks.
- Batch normalization helps improve the stability and efficiency of the model during training, while dropout helps prevent overfitting by promoting the learning of more robust features.
- Maybe data augmentation would help solve the test challenge set classification.

3 Data Augmentation

- Data augmentation is a technique used in machine learning to increase the size of a training dataset by creating new variations of the existing data.

- The aim of data augmentation is to improve the accuracy and robustness of machine learning models by introducing more diversity in the training data.

- In practice, data augmentation involves applying a set of transformations or modifications to the existing data, such as rotation, scaling, flipping, cropping, adding noise, changing brightness or contrast, and so on.

- These transformations can create new examples that are similar to the original data but with some variations, such as different viewpoints, lighting conditions, or deformations.

- Data augmentation is particularly useful in computer vision tasks, such as image classification or object detection, where the availability of large and diverse datasets is crucial for achieving high accuracy.

- By applying data augmentation techniques, machine learning models can learn to recognize objects and patterns in a more robust and generalized way, even when faced with new or unseen examples.

3.1 Introduction

After performing data augmentation on the Wallpaper dataset, we end up with a total of 51000 images on the training set. The transforms aren't applied on the test sets as they don't hamper the training process.

Type	Transforms Description	Number of Images
Augmentation Strategy 1	Random Crop, Random Horizontal Flip, Random Vertical Flip, Resizing, Grayscale, To Tensor, Normalization	17000
Augmentation Strategy 2	Random Rotation, Resizing, Grayscale, To Tensor, Normalization	17000
Normal	Resizing, Grayscale, To Tensor, Normalization	17000
		Total: 51000

Table 3: Data Augmentation Wallpaper Dataset: Train Set

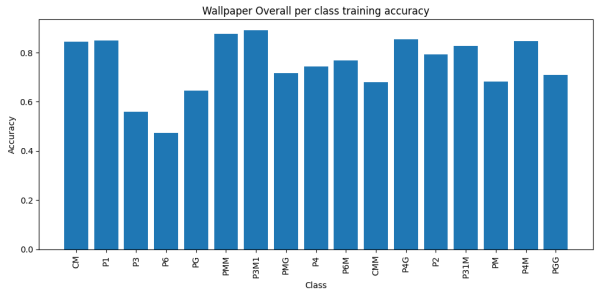
```python
if args.aug_train:
    transform_1 = transforms.Compose([
        transforms.RandomCrop((150, 150)),
        transforms.RandomHorizontalFlip(p = 0.5),
        transforms.RandomVerticalFlip(p = 0.5),
        transforms.Resize((args.img_size, args.img_size)),
        transforms.Grayscale(),
        transforms.ToTensor(),
        transforms.Normalize((0.5, ), (0.5, ))
    ])
    transform_2 = transforms.Compose([
        transforms.RandomRotation(360),
        transforms.Resize((args.img_size, args.img_size)),
        transforms.Grayscale(),
        transforms.ToTensor(),
        transforms.Normalize((0.5, ), (0.5, ))
    ])
transform = transforms.Compose([
    transforms.Resize((args.img_size, args.img_size)),
    transforms.Grayscale(),
    transforms.ToTensor(),
    transforms.Normalize((0.5, ), (0.5, )),
])

# If data augmentation is used, concatenate the augmented and normal datasets to get 34000 training images
if args.aug_train:
    train_dataset_aug_1 = ImageFolder(os.path.join(data_root, 'train'), transform=transform_1)
    train_dataset_aug_2 = ImageFolder(os.path.join(data_root, 'train'), transform=transform_2)
    train_dataset_normal = ImageFolder(os.path.join(data_root, 'train'), transform=transform)
    train_dataset = ConcatDataset([train_dataset_aug_1, train_dataset_aug_2, train_dataset_normal])
else:
    train_dataset = ImageFolder(os.path.join(data_root, 'train'), transform=transform)
```

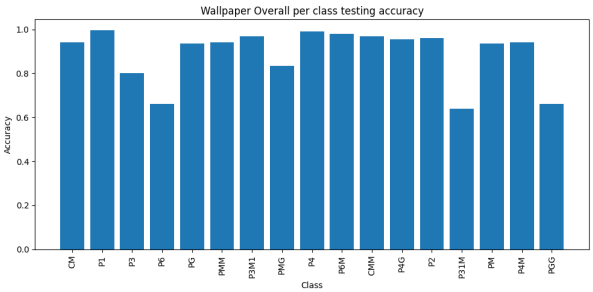
Figure 33: Data Augmentation Implementation

3.2 Baseline Model

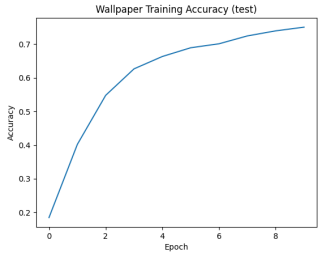
3.2.1 Test Dataset



(a) Per-Class Train Accuracy



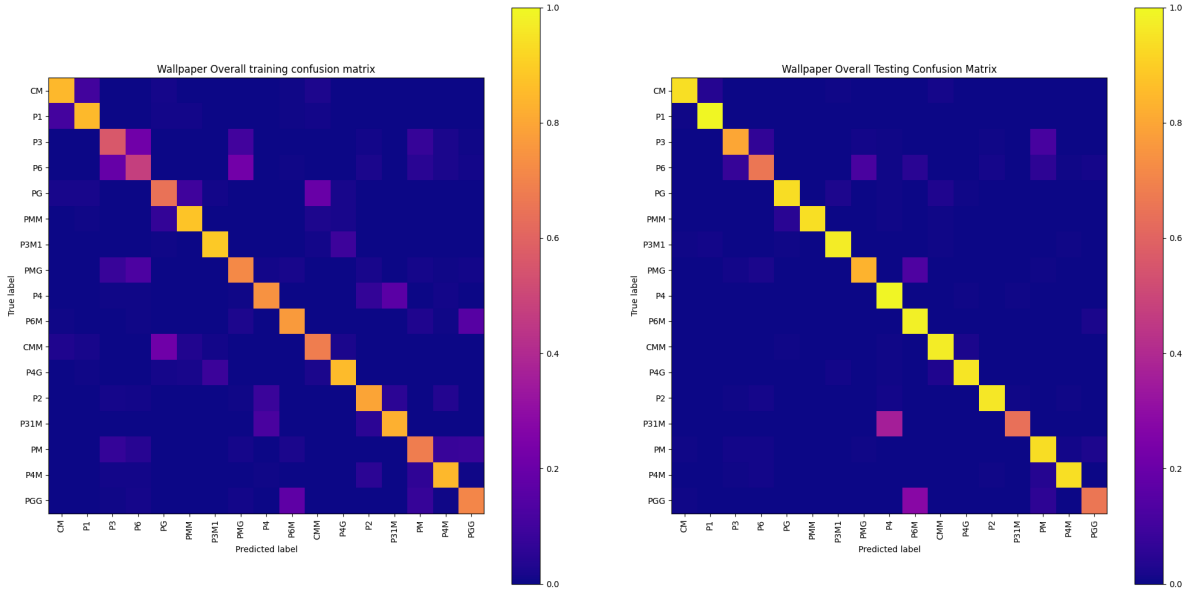
(b) Per-Class Test Accuracy



(c) Model Test Accuracy vs Epochs

Figure 34: Data Augmented Baseline Model: Wallpaper(test)

Confusion Matrices



(a) Train Confusion Matrix (b) Test Confusion Matrix

Figure 35: Data Augmented Baseline Model Confusion Matrices: Wallpaper(test)

Layer 1 Visualization

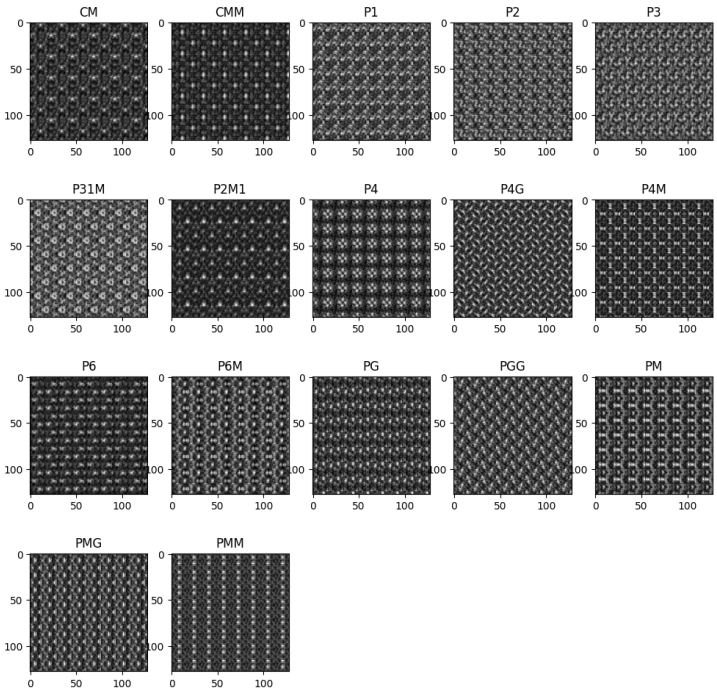
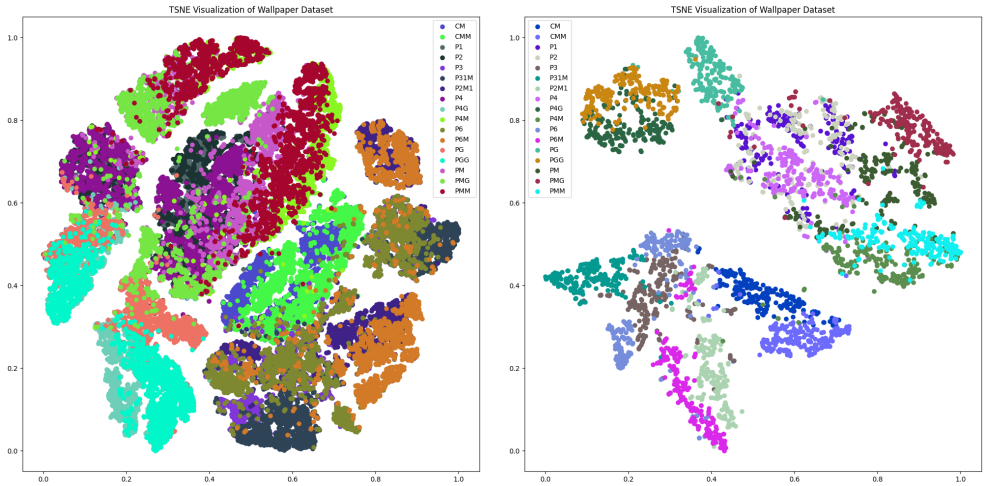


Figure 36: Data Augmented Baseline Model Layer 1 Visualization: test

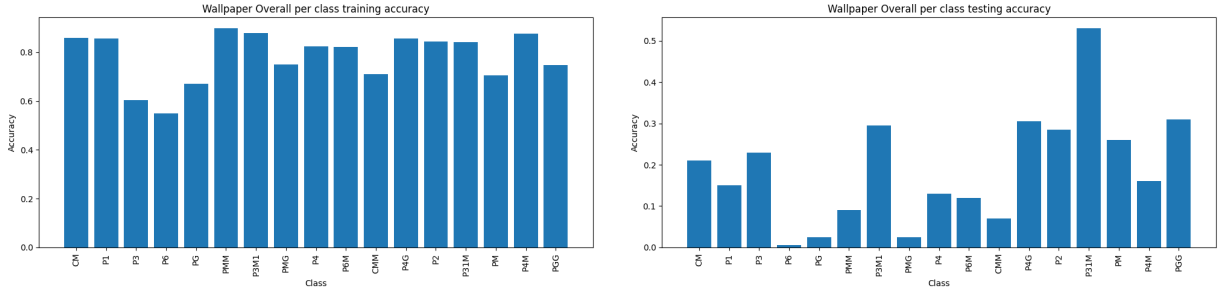
T-SNE Visualization



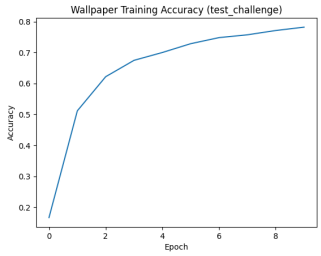
(a) TSNE: Train Set (b) TSNE: Test Set

Figure 37: Data Augmented Baseline Model TSNE Visualization: Wallpaper(test)

3.2.2 Test Challenge Set



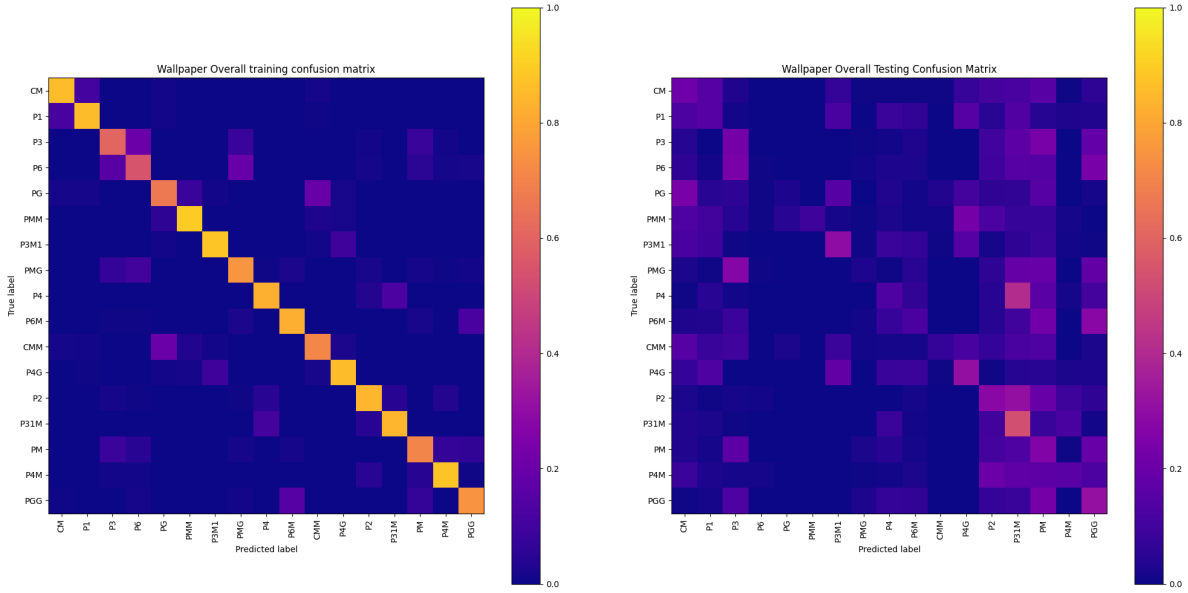
(a) Per-Class Train Accuracy (b) Per-Class Test Accuracy



(c) Model Test Accuracy vs Epochs

Figure 38: Data Augmented Baseline Model: Wallpaper(test_challenge)

Confusion Matrices



(a) Train Confusion Matrix (b) Test Confusion Matrix

Figure 39: Data Augmented Baseline Model Confusion Matrices: Wallpaper(test_challenge)

Layer 1 Visualization

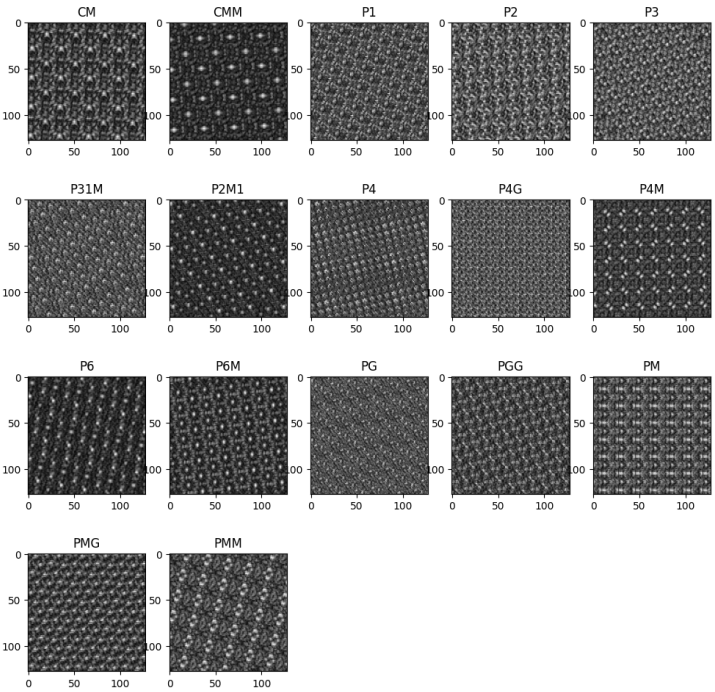
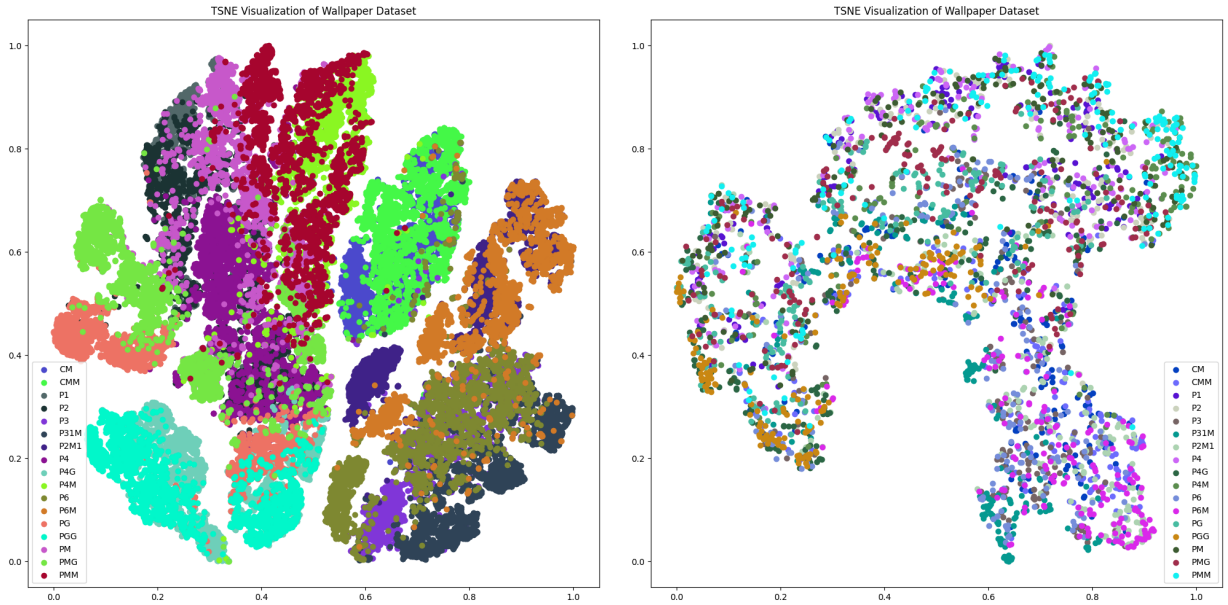


Figure 40: Data Augmented Baseline Model Layer 1 Visualization: test_challenge

T-SNE Visualization



(a) TSNE: Train Set

(b) TSNE: Test Set

Figure 41: Data Augmented Baseline Model TSNE Visualization: Wallpaper(test_challenge)

Observations: test and test_challenge sets

- In contrast with the baseline model trained on the smaller dataset, the performance on the train set has gotten worse as the model is not good enough to generalize over a larger dataset and more set of features.
- However, the performance on the test_challenge set has gotten better compared to the dataset without data augmentation.
- Also, the confusion matrices have shown good results in the case of test_challenge set as the dataset enlargement has sufficed the model to learn better in terms of minor details.
- TSNE visualizations also show a clear improvement in the case of test sets, however the TSNE train visualization has gotten worse.

3.3 Improved Model

- The model used for training the data augmented training set of 51000 images is a modified VGG-16 [3].
- In the 2014 ImageNet Classification Challenge [4], VGG16 achieved a 92.7% classification accuracy and is considered one of the most novel CNN models used for image classification tasks.

- As for as the modifications are concerned, the first CNN layer takes single gray scale input channel, and outputs 32 feature maps instead of 64 suggested in the paper.
- The last layer also, instead of having 1000 neurons to hold the ImageNet dataset, has 17 neurons to output 17 wallpaper categories.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 128, 128]	320
BatchNorm2d-2	[-1, 32, 128, 128]	64
ReLU-3	[-1, 32, 128, 128]	0
Conv2d-4	[-1, 64, 128, 128]	18,496
BatchNorm2d-5	[-1, 64, 128, 128]	128
ReLU-6	[-1, 64, 128, 128]	0
MaxPool2d-7	[-1, 64, 64, 64]	0
Conv2d-8	[-1, 128, 64, 64]	73,856
BatchNorm2d-9	[-1, 128, 64, 64]	256
ReLU-10	[-1, 128, 64, 64]	0
Conv2d-11	[-1, 128, 64, 64]	147,584
BatchNorm2d-12	[-1, 128, 64, 64]	256
ReLU-13	[-1, 128, 64, 64]	0
MaxPool2d-14	[-1, 128, 32, 32]	0
Conv2d-15	[-1, 256, 32, 32]	295,168
BatchNorm2d-16	[-1, 256, 32, 32]	512
ReLU-17	[-1, 256, 32, 32]	0
Conv2d-18	[-1, 256, 32, 32]	590,080
BatchNorm2d-19	[-1, 256, 32, 32]	512
ReLU-20	[-1, 256, 32, 32]	0
Conv2d-21	[-1, 256, 32, 32]	590,080
BatchNorm2d-22	[-1, 256, 32, 32]	512
ReLU-23	[-1, 256, 32, 32]	0
MaxPool2d-24	[-1, 256, 16, 16]	0
Conv2d-25	[-1, 512, 16, 16]	1,180,160
BatchNorm2d-26	[-1, 512, 16, 16]	1,024
ReLU-27	[-1, 512, 16, 16]	0
Conv2d-28	[-1, 512, 16, 16]	2,359,808
BatchNorm2d-29	[-1, 512, 16, 16]	1,024
ReLU-30	[-1, 512, 16, 16]	0
Conv2d-31	[-1, 512, 16, 16]	2,359,808
BatchNorm2d-32	[-1, 512, 16, 16]	1,024
ReLU-33	[-1, 512, 16, 16]	0
MaxPool2d-34	[-1, 512, 8, 8]	0
Conv2d-35	[-1, 512, 8, 8]	2,359,808
BatchNorm2d-36	[-1, 512, 8, 8]	1,024
ReLU-37	[-1, 512, 8, 8]	0
Conv2d-38	[-1, 512, 8, 8]	2,359,808
BatchNorm2d-39	[-1, 512, 8, 8]	1,024
ReLU-40	[-1, 512, 8, 8]	0
Conv2d-41	[-1, 512, 8, 8]	2,359,808
BatchNorm2d-42	[-1, 512, 8, 8]	1,024
ReLU-43	[-1, 512, 8, 8]	0
MaxPool2d-44	[-1, 512, 4, 4]	0
Dropout-45	[-1, 8192]	0
Linear-46	[-1, 4096]	33,558,528
ReLU-47	[-1, 4096]	0
Dropout-48	[-1, 4096]	0
Linear-49	[-1, 4096]	16,781,312
ReLU-50	[-1, 4096]	0
Linear-51	[-1, 17]	69,649

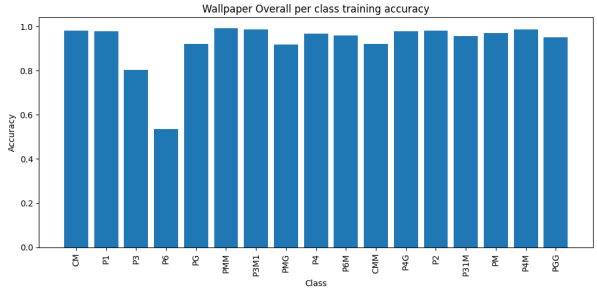
Total params: 65,112,657
Trainable params: 65,112,657
Non-trainable params: 0

Input size (MB): 0.06
Forward/backward pass size (MB): 93.28
Params size (MB): 248.39
Estimated Total Size (MB): 341.73

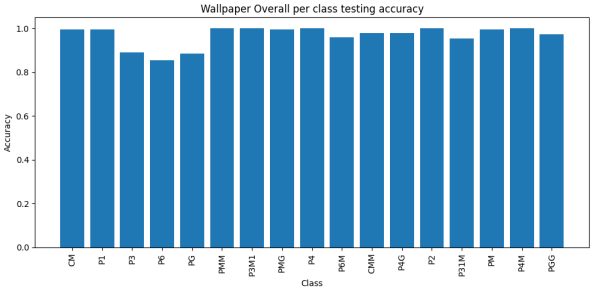
Figure 42: Improved Model for Data Augmentation: Modified VGG16 Model

3.3.1 Results

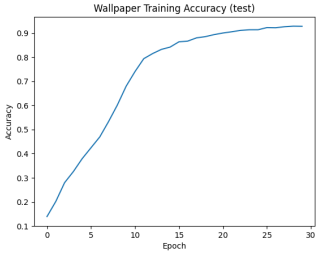
3.3.2 Test Dataset



(a) Per-Class Train Accuracy



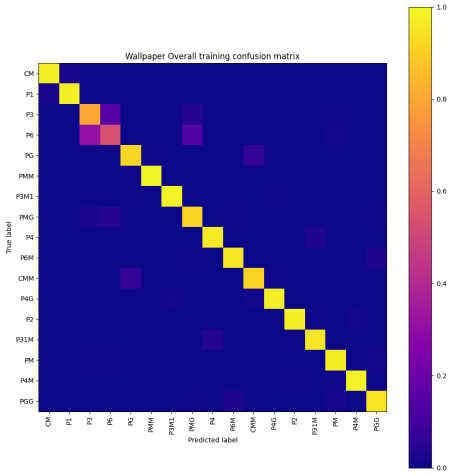
(b) Per-Class Test Accuracy



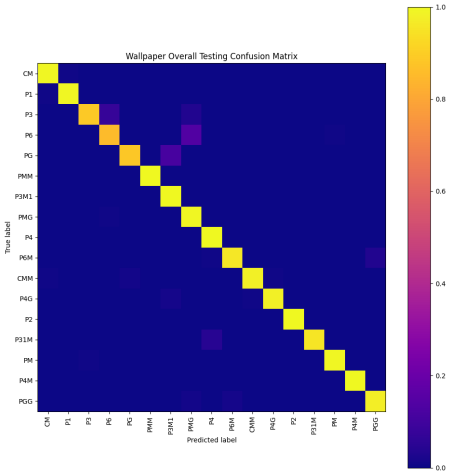
(c) Model Test Accuracy vs Epochs

Figure 43: Data Augmented Improved Model: Wallpaper(test)

Confusion Matrices



(a) Train Confusion Matrix



(b) Test Confusion Matrix

Figure 44: Data Augmented Improved Model Confusion Matrices: Wallpaper(test)

Layer 1 Visualization

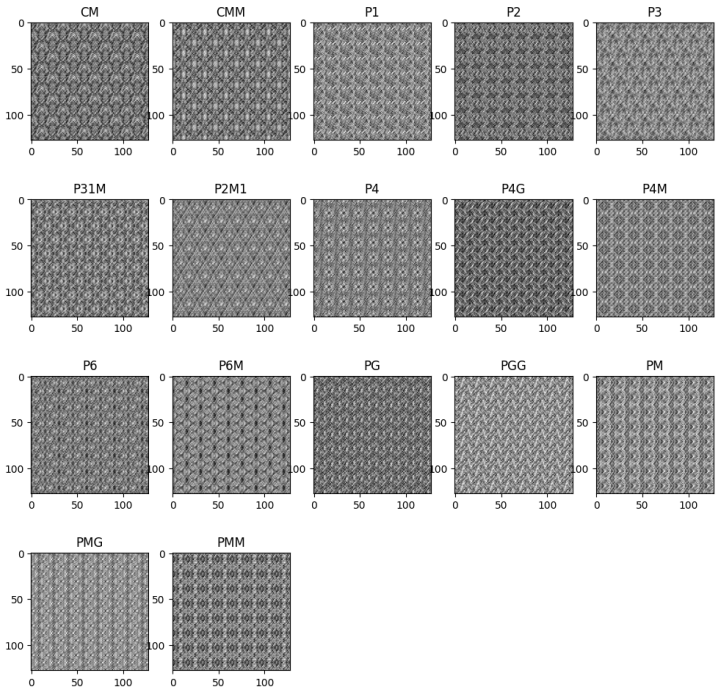
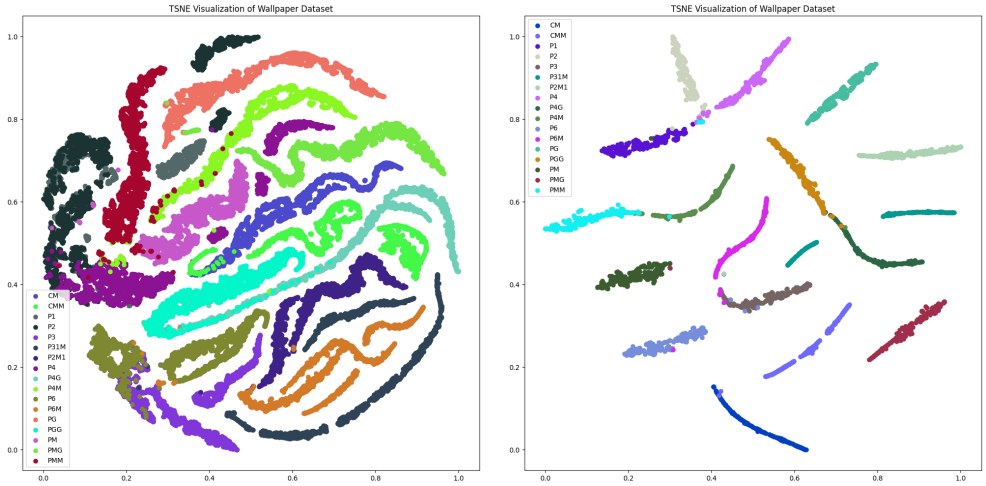


Figure 45: Data Augmented Improved Model Layer 1 Visualization: test

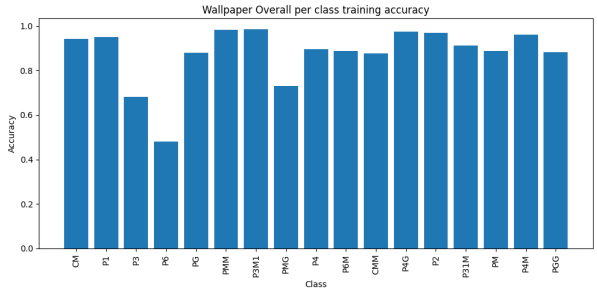
T-SNE Visualization



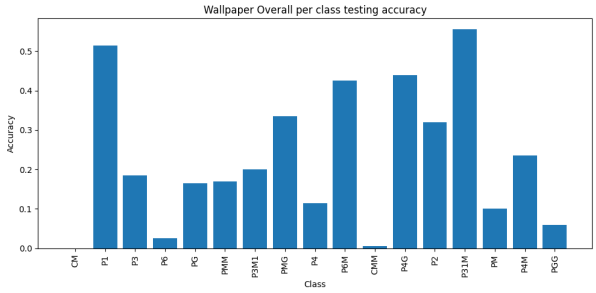
(a) TSNE: Train Set (b) TSNE: Test Set

Figure 46: Data Augmented Improved Model TSNE Visualization: Wallpaper(test)

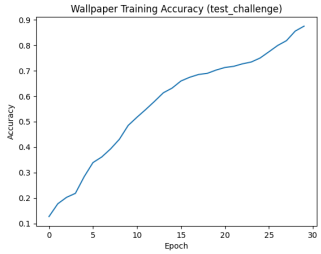
3.3.3 Test Challenge Set



(a) Per-Class Train Accuracy



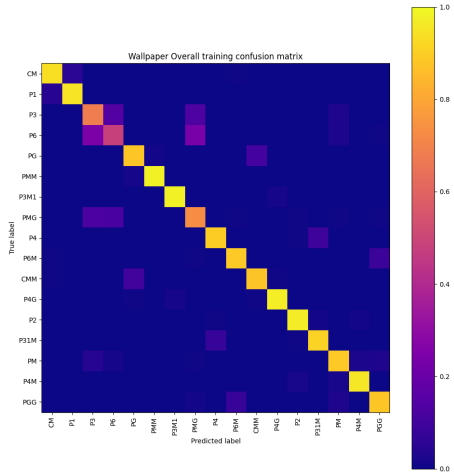
(b) Per-Class Test Accuracy



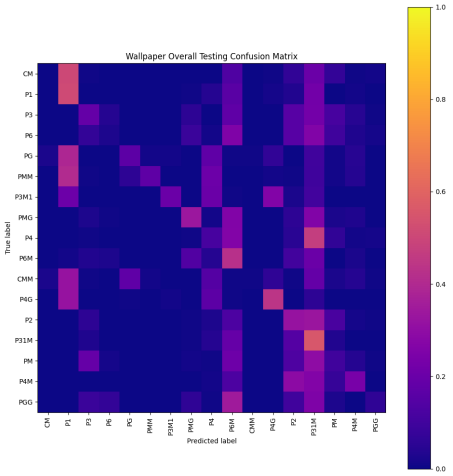
(c) Model Test Accuracy vs Epochs

Figure 47: Data Augmented Improved Model: Wallpaper(test_challenge)

Confusion Matrices



(a) Train Confusion Matrix



(b) Test Confusion Matrix

Figure 48: Data Augmented Improved Model Confusion Matrices: Wallpaper(test_challenge)

Layer 1 Visualization

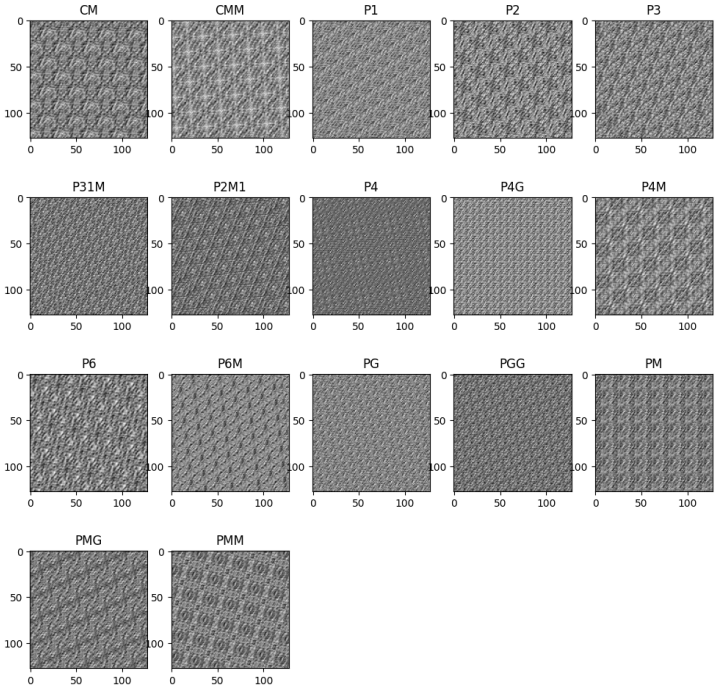
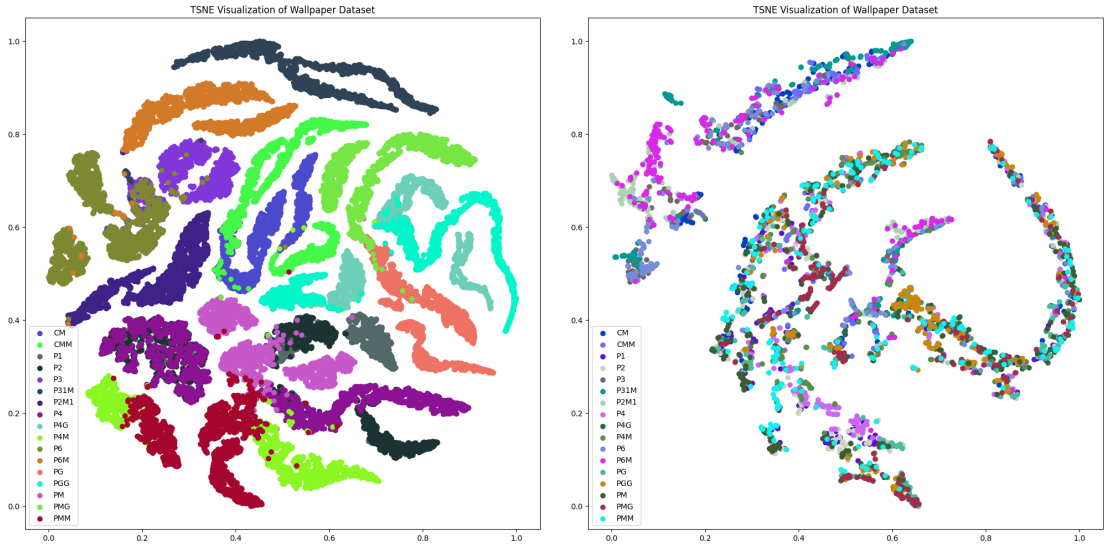


Figure 49: Data Augmented Improved Model Layer 1 Visualization: test_challenge

T-SNE Visualization



(a) TSNE: Train Set (b) TSNE: Test Set

Figure 50: Data Augmented Improved Model TSNE Visualization: Wallpaper(test_challenge)

Observations: test and test_challenge sets

- Quantitatively, the improvements in the model performance are very noticeable over the use of the baseline model through the use of data augmentation strategy.
- Training the model with random flips, rotations, and cropping has helped the improved model to learn good sets of features when used for testing the test_challenge set.
- In the normal test set, the performance was really good in terms of both accuracy and layer visualization.
- In the test_challenge set, better performance was observed compared to the baseline model which is clearly seen in the TSNE visualization.
- The train set seems to be quite randomly laid around which might be tough for the model to train on, however having 16 layers and parameters orders of magnitude larger than the baseline model help classify those challenging outlier classes in the test set.
- As for as confusion matrices are concerned, the test challenge set in the improved model showed more strong contrast in the diagonal elements, suggesting a better classification performance.

3.4 Analysis

Model	Test Set	Train Accuracy	Test Accuracy	Standard Deviation Train	Standard Deviation Test
Baseline	test	75.049	88.853	0.11341	0.11931
Improved	test	92.843	96.824	0.10790	0.04495
Baseline	test_challenge	78.200	18.824	0.10022	0.13155
Improved	test_challenge	87.541	22.647	0.12735	0.17205

Table 4: Data Augmentation Wallpaper Dataset Model Analysis

- Its clearly evident from the Table that the improved model has performed way better than the baseline model when the dataset got bigger.
- The baseline model is not clearly robust enough to learn all the augmentation strategies employed.
- There is still room for improvement in the case of test_challenge set classification, but however an increase in performance has been observed in this case.
- Adding more layers, batch normalization, dropout and non-linearity through the use of activation function has helped improving the performance.
- Also, data augmentation has been shown to produce promising ways to increase the accuracy of classification tasks. In a modern sense, techniques enabled by CycleGAN and other similar networks are more promising compared to traditional data augmentation tasks [5].

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. 12

[2] Wikipedia, "Wallpaper group," https://en.wikipedia.org/w/index.php?title=Wallpaper_group&oldid=1143826674, 2023, [Online; accessed 13-March-2023]. 12

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. 33

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. 33

[5] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," 2017. 39