Project 3

Piyush Nagasubramaniam

March 19, 2023

Abstract

In this project, we study the use of Neural Networks for Deep Learning. We show that the MLP architecture can be used to achieve 77.2% training accuracy and 75.1% testing accuracy for classifying Taiji poses from the PSU-TMM100 dataset. In addition to MLPs, we study the use of CNNs to classify Wallpaper groups from images. We achieve 82.81% training accuracy and 80.44% testing accuracy for classifying the wallpaper groups. Our custom CNN architecture has 15.84% testing accuracy when classifying wallpaper patterns from the challenge dataset. Lastly, we use SOTA CNN architectures to perform a design space exploration and achieve high accuracy on the challenge dataset. The highest testing accuracy achieved was ***62.16%*** using a pretrained ResNet101 model. We also present a competitive compute-friendly solution that achieves ***42.59%*** testing accuracy using a MobileNetv3 Network trained from scratch. A demo to conveniently reproduce the results is available at: https://github.com/piyushnags/PRMLProject3

Contents

1 Introduction

In this project, we will study how neural networks can be used to perform classification tasks and thus, show that they are useful for a variety of downstream tasks like object detection, image segmentation, etc. First, we will analyze the Multilayer Perceptron (MLP) as a classifier and then consider Convolutional Neural Networks (CNNs) for image classification. Lastly, we will conduct Transfer Learning experiments on pre-trained image classification models and study how effective different backbones are for detecting symmetry in wallpaper groups.

2 Approach

2.1 Data

We will use the Taiji pose dataset (PSU-TMM100) and the Wallpaper Pattern Images dataset to conduct experiments.

2.1.1 Part 1: Taiji Keypose Dataset

The experiments are performed on two versions of the PSU-TMM100 dataset: "lod4" and "full". There are a total of 49774 samples in the dataset. The "lod4" is a reduced version of the dataset that contains 67 features instead of the 1961 features in the "full" version. There are a total of 46 classes (including a transitional pose for non-key frames).

2.1.2 Part 2: Wallpaper Dataset

There are 2 subsets for this part of the project. The first wallpaper dataset contains 20,400 samples of 256x256 images. These images are divided into 17 classes since there are 17 wallpaper groups in two dimensions. The challenge dataset contains 3400 samples of the same wallpaper groups, but the images are significantly different in appearance compared to the basic wallpaper group images, which makes it a good dataset to evaluate the effectiveness of models on unseen data.

2.2 Methods

We will discuss the implementations and reasoning behind the choice of architecture of the neural networks in this section.

2.2.1 Part 1: Taiji Pose Classification using MLP

For part 1, we implement two different MLP architectures. The baseline model has only 1 hidden layer with 1024 neurons. In the first modified architecture, we use the same number of hidden neurons (i.e., 1024 neurons), but split them into two hidden layers containing half the total hidden neurons (i.e., 512 neurons per hidden layer). We also add ReLU layers after each hidden layer to improve the accuracy of the model and its ability to model complex non-linear decision boundaries.

In the second MLP architecture, we follow the same methodology of using two hidden layers like in the first architecture. However, we use an improved non-linear activation function i.e, ReLU6, which allows the model to learn sparse features earlier. To make the model converge faster, we also use batch normalization before the ReLU6 layers. Lastly, we add a dropout layer with a probability of 0.2 to prevent overfitting.

2.2.2 Part 2: Wallpaper Group Classification using CNN

We consider two modified CNN architectures for the wallpaper group classification task. In the first modified architecture, we replace the regular ReLUs with the Leaky ReLU activation. This non-linear activation speeds up the training process and fixes the dying ReLU problem by scaling negative values by 0.1. We also add batch normalization to speed up the training process. We do not make any changes to the fully connected layers. The convolution layers remain the same as the base architecture as well.

In the second modified CNN architecture, we build on the first modified architecture. We still use batch normalization and Leaky ReLU activations. However, we extract 64 feature maps using 3x3 convolution kernels in the first layer instead of 32 feature maps. After pooling and batch normalization, we generate 128 feature maps using 3x3 convolutions. To reduce the number of parameters, we perform pointwise convolutions to reduce the number of activation maps back to 64. We resize the images to 128x128 before sending the images to the model. The first fully connected layer maps a 4096-dimensional embedding to a reduced 1024-dimensional embedding. The last fully connected layer maps the 1024-dimensional embedding to a vector with a dimensionality of 17.

We choose a 4096-D embedding for the first FC layer because it is a good representation of natural image features as mentioned in [1]. The reduction to 1024 features results in minimal information loss as mentioned in [2].

Data augmentation is used to improve the performance of the model on unseen data. We apply random cropping, random translation, random scaling, random rotation, and random horizontal flips.

2.2.3 Transfer Learning and Design Space Exploration

To improve the performance on the challenge set of wallpaper groups, we use Transfer Learning on ResNet101 and DenseNet121 networks pretrained on the Imagenet dataset. We also train a MobileNetv3 (Small) model from scratch as it only has under 1 million trainable parameters in total. Mobilenet is a very light model and has fast inference time, Resnet101 is balanced between accuracy, training time, and inference time but has the most parameters, while Densenet has the slowest training time and a high GPU memory consumption. The models chosen have sufficient variation in the total number of parameters, training and inference time, and accuracy for a thorough design space exploration.

We also use a learning rate scheduler when training the MobileNetv3 small model for faster training. The learning rate scheduler used is the Cosine Annealing LR scheduler as per [3]. We do not implement the warm restarts as mentioned in the SGDR paper.

3 Results

We will discuss the results of all experiments including the transfer learning experiments for extra credit in this section.

3.1 Part 1: Taiji Pose Classification using MLP

In this section, we present the performance results of the base architecture and the two modified architectures. The results of the MLP experiments are presented in Table 1. For modified architecture 1 on the "full" version of the Taiji dataset, we can see that the training of subject 4 is the most unstable. Some of the noisy peaks can be attributed to the use of the Adam optimizer. This is a tradeoff between convergence time and stability when compared to an optimizer like SGD (with momentum). However, the main reason is because dropout is used. This is discussed at the end of this subsection.

Based on Figures 2c and 2d, we see that pose 10 is often misclassified as 12, which is why it has a low classification rate. Additionally, poses 26-29 are often misclassified among each other.

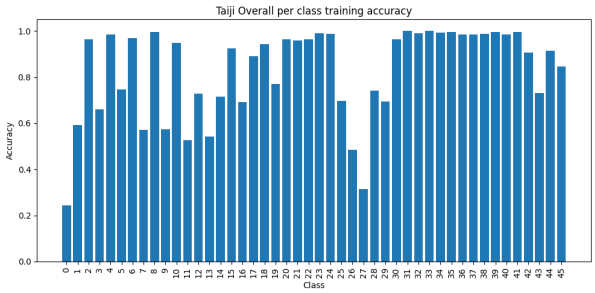
Let us consider the "lod4" version of the Taiji dataset which has only 67 features. It is clear that there is negligible performance degradation when using the dimensionally reduced "lod4" dataset. Thus, it is important to analyze the data and determine the least number of dimensions that can accurately represent the data by eliminating redundancy before using a model to make inferences on the dataset.

For modified architecture 2 on the "full" Taiji dataset, we can see that the overall performance has not improved, but it is less unstable. However, compared to architecture 1, the stability of Subject 4 is worse in this architecture. It is also worth noting that the other subjects are more stable compared to modified architecture 1 without batch normalization. The overall performance on the "lod4" version of the dataset is consistent with modified architecture 1, i.e. 67 features are a good representation of the dataset instead of all the 1961 features.

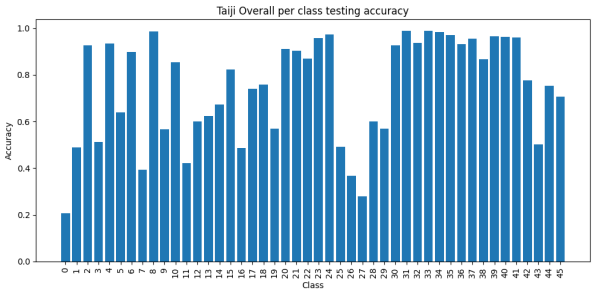
There is one interesting difference when comparing modified architecture 2 with the previous architectures. In Figure 5e, we see that the training accuracy is worse than the testing accuracy for some subjects. This anomalous behavior is because we use a dropout layer, unlike the previous architectures. Not all neurons are being used during training, which explains the relatively lower training accuracy. During inference, all neurons are used so the accuracy is not affected due to lower model complexity.

Model	Dataset	Training Accuracy	Testing Accuracy	Training Class-wise Std	Testing Class-wise Std
Baseline	Taiji full	83.1%	73.9%	0.1870	0.2248
Baseline	Taiji lod4	83.1%	73.9%	0.1870	0.2248
Modified MLP 1	Taiji full	92.3%	76.5%	0.1159	0.2056
Modified MLP 1	Taiji lod4	92.3%	76.5%	0.1159	0.2056
Modified MLP 2	Taiji full	77.2%	75.1%	0.0668	0.2096
Modified MLP 2	Taiji lod4	77.2%	75.1%	0.0668	0.2096

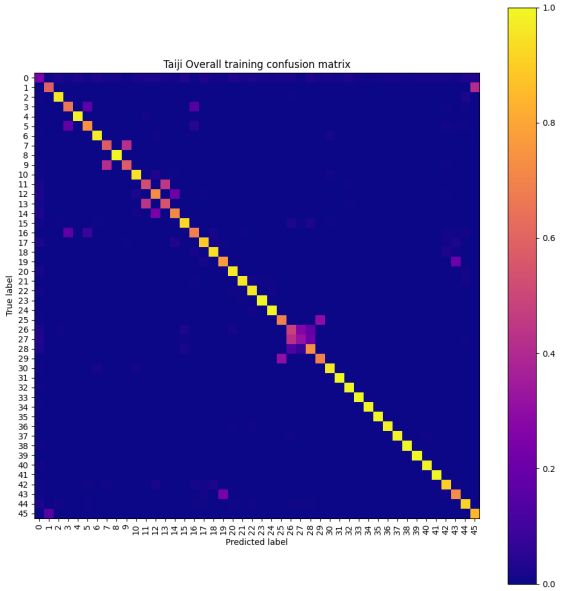
Table 1: Classification Results using different MLP Architectures



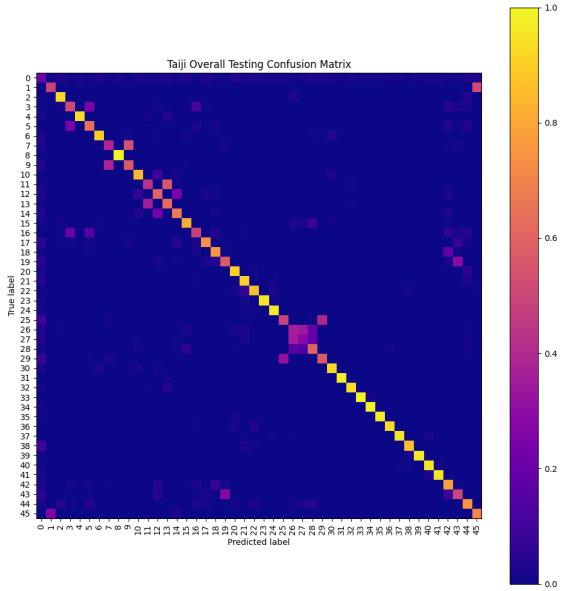
(a) Class-wise Training Accuracy using Baseline MLP



(b) Class-wise Testing Accuracy using Baseline MLP

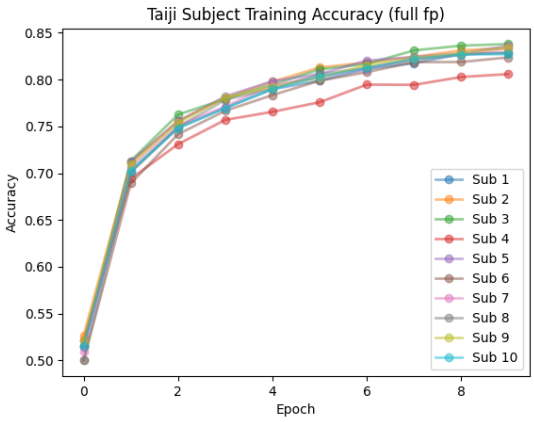


(c) Overall Training Confusion Matrix for Baseline MLP



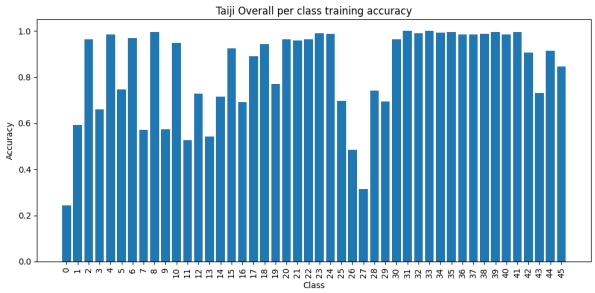
(d) Overall Testing Confusion Matrix for Baseline MLP

(e) Subject-wise Training Accuracy for Baseline MLP

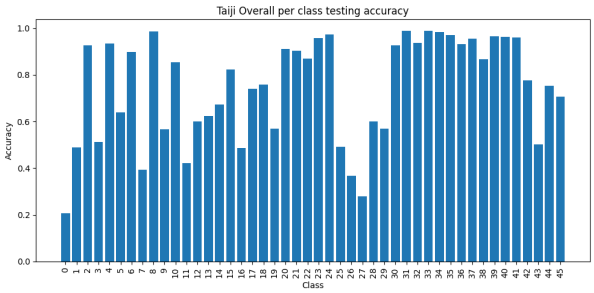


(f) Training Accuracy Curve for Baseline MLP

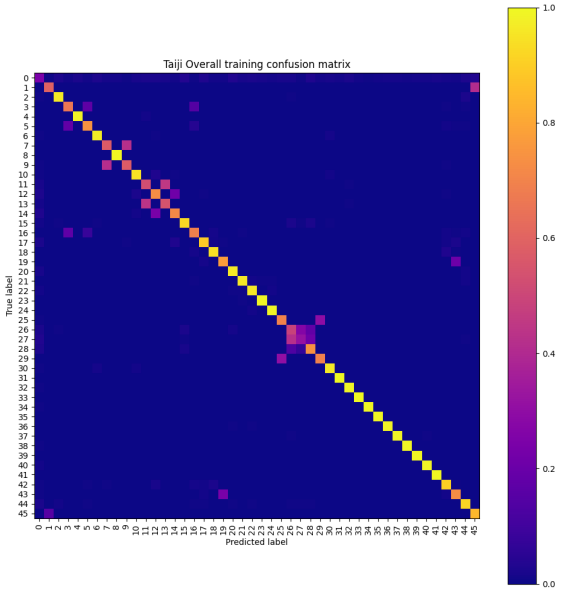
Figure 1: Classification Results of using Baseline MLP on Taiji full



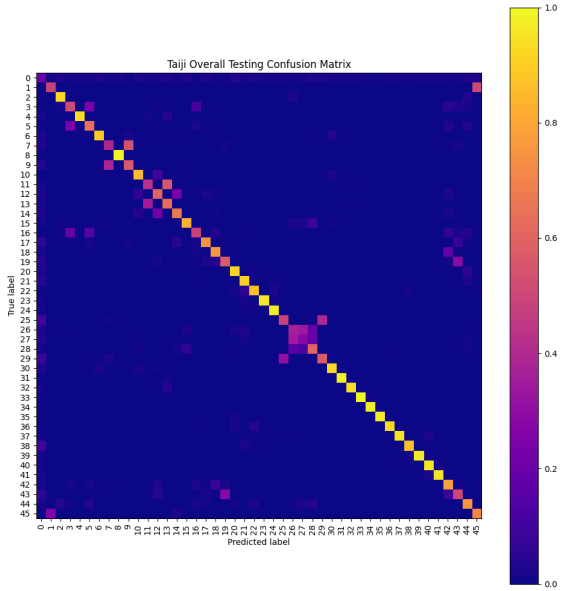
(a) Class-wise Training Accuracy using Baseline MLP



(b) Class-wise Testing Accuracy using Baseline MLP

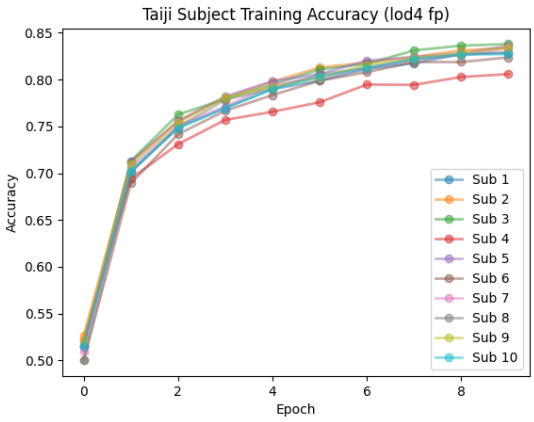


(c) Overall Training Confusion Matrix for Baseline MLP



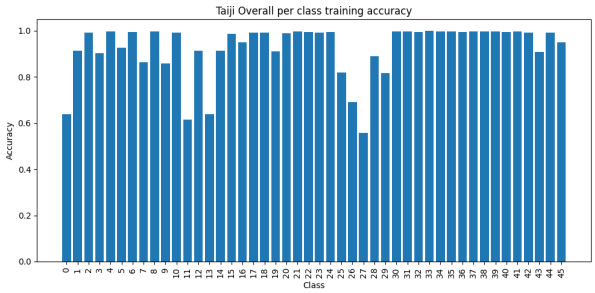
(d) Overall Testing Confusion Matrix for Baseline MLP

(e) Subject-wise Training Accuracy for Baseline MLP

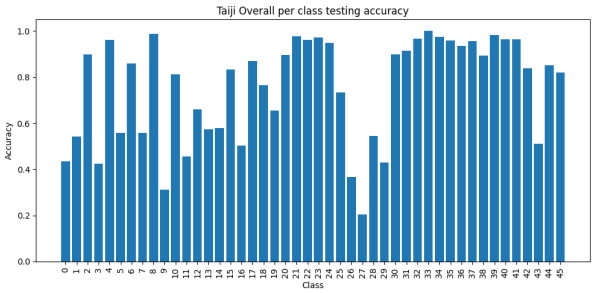


(f) Training Accuracy Curve for Baseline MLP

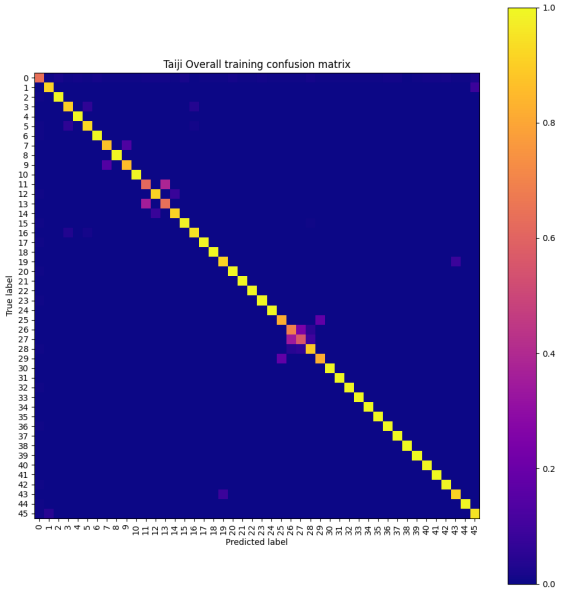
Figure 2: Classification Results of using Baseline MLP on Taiji lod4



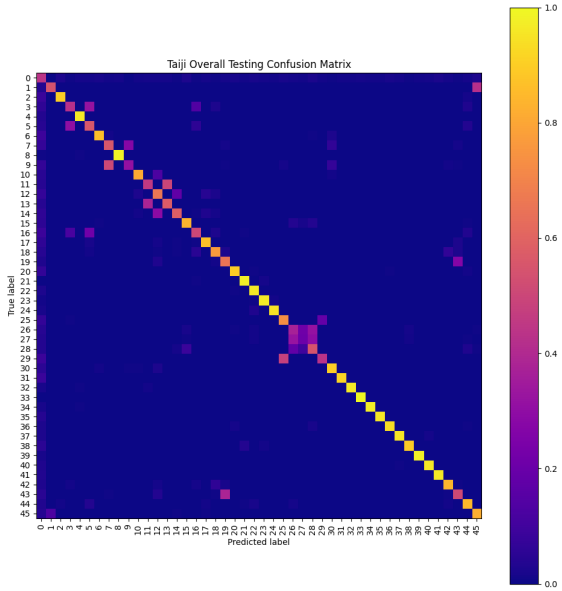
(a) Class-wise Training Accuracy using MLP Architecture 1



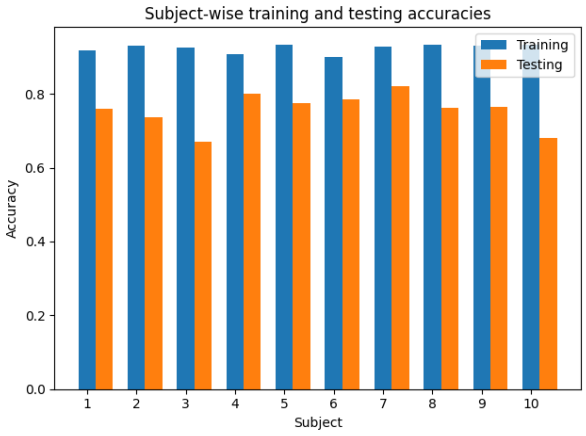
(b) Class-wise Testing Accuracy using MLP Architecture 1



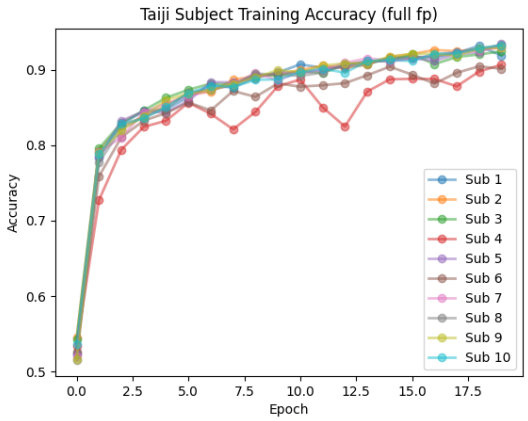
(c) Overall Training Confusion Matrix for MLP Architecture 1



(d) Overall Testing Confusion Matrix for MLP Architecture 1

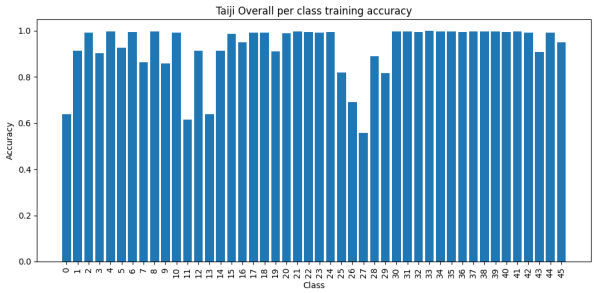


(e) Subject-wise Training Accuracy for MLP Architecture 1

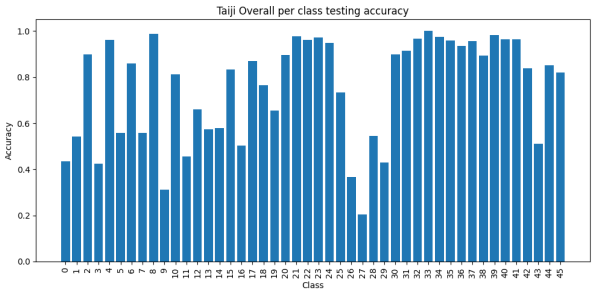


(f) Training Accuracy Curve for MLP Architecture 1

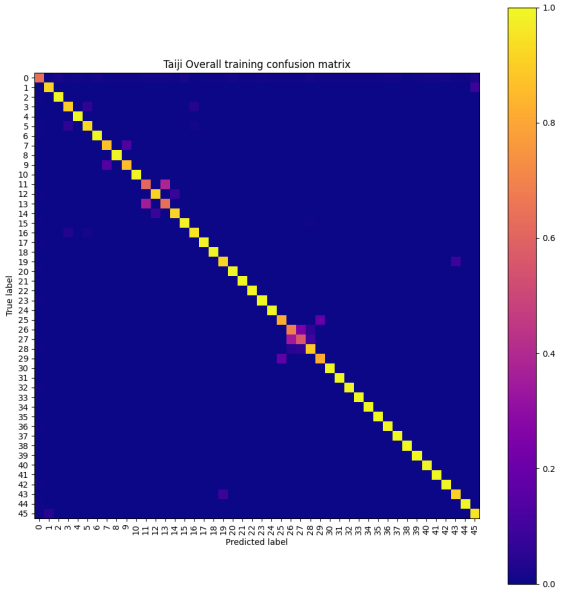
Figure 3: Classification Results of using MLP Architecture 1 on Taiji full



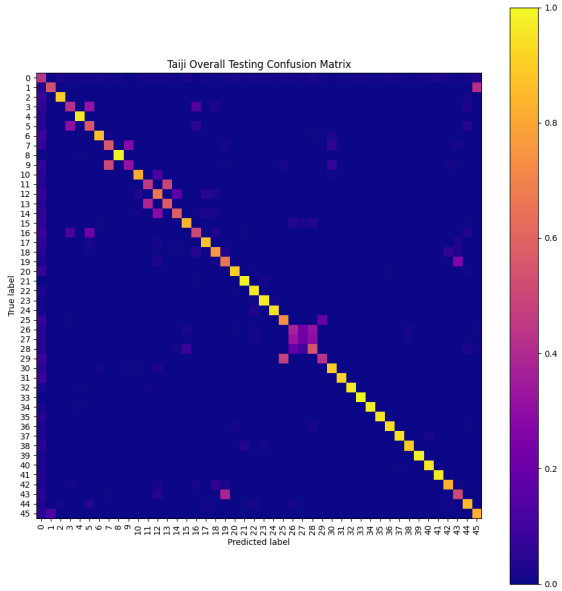
(a) Class-wise Training Accuracy using MLP Architecture 1



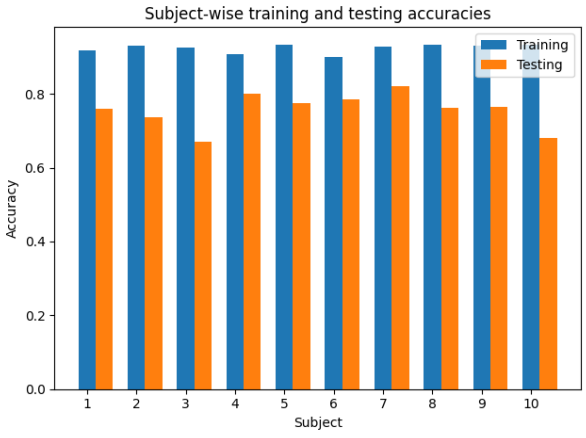
(b) Class-wise Testing Accuracy using MLP Architecture 1



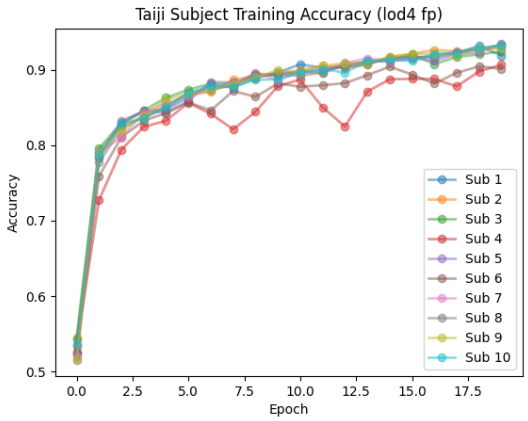
(c) Overall Training Confusion Matrix for MLP Architecture 1



(d) Overall Testing Confusion Matrix for MLP Architecture 1

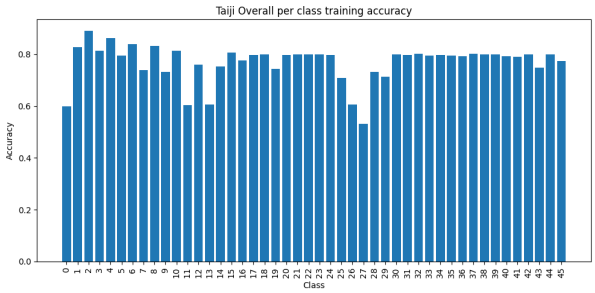


(e) Subject-wise Training Accuracy for MLP Architecture 1

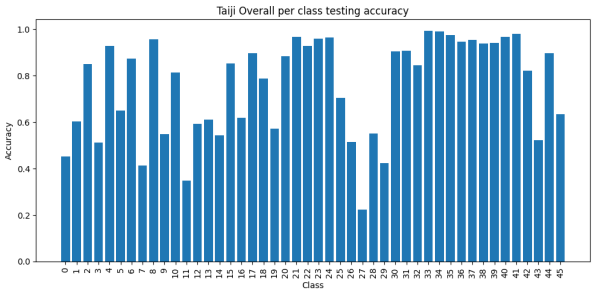


(f) Training Accuracy Curve for MLP Architecture 1

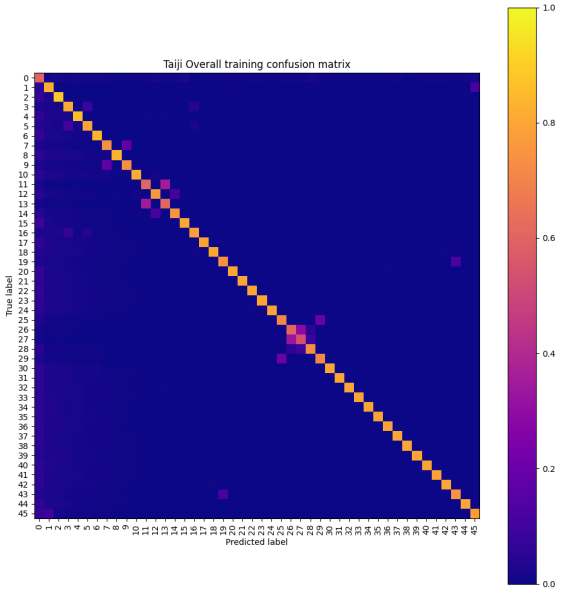
Figure 4: Classification Results of using MLP Architecture 1 on Taiji lod4



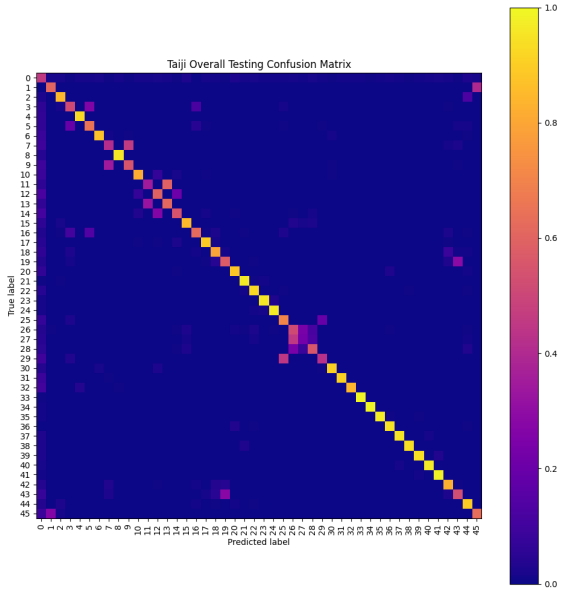
(a) Class-wise Training Accuracy using MLP Architecture 2



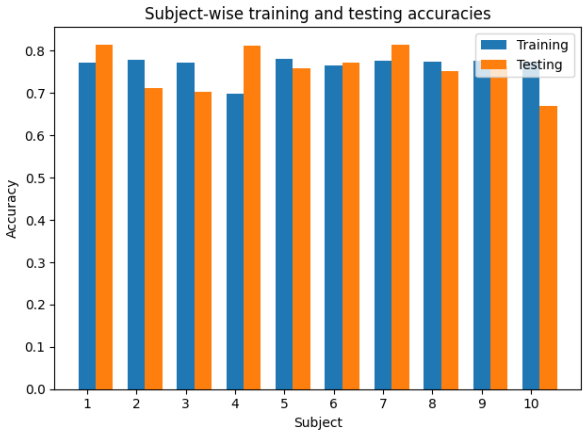
(b) Class-wise Testing Accuracy using MLP Architecture 2



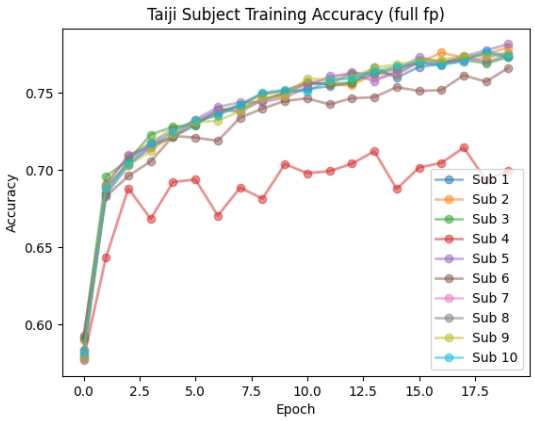
(c) Overall Training Confusion Matrix for MLP Architecture 2



(d) Overall Testing Confusion Matrix for MLP Architecture 2

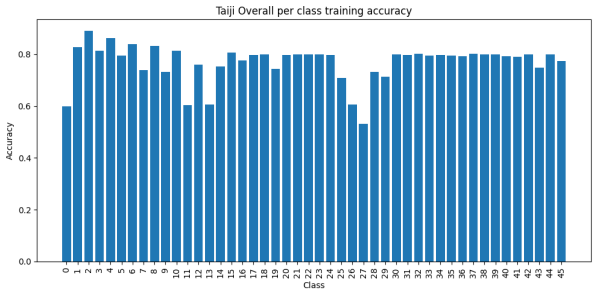


(e) Subject-wise Training Accuracy for MLP Architecture 2

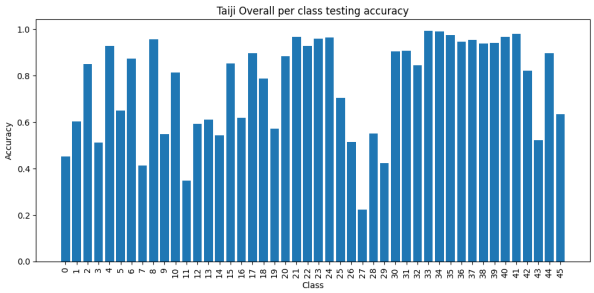


(f) Training Accuracy Curve for MLP Architecture 2

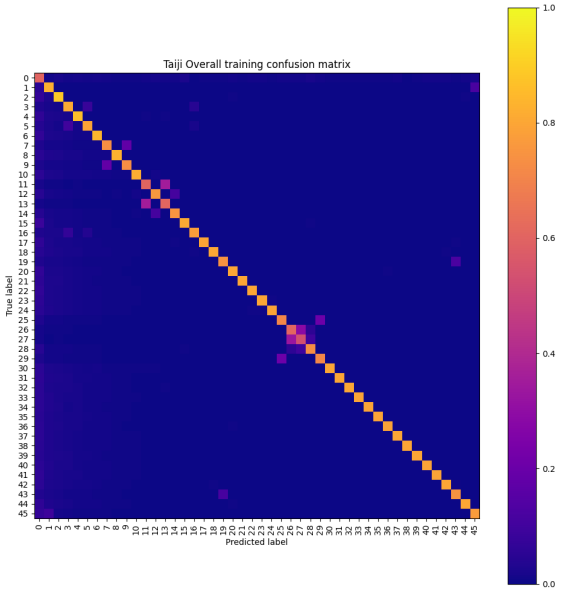
Figure 5: Classification Results of using MLP Architecture 2 on Taiji full



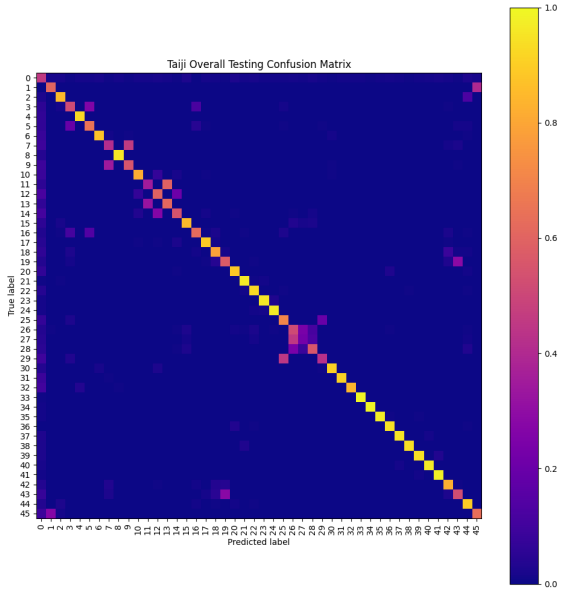
(a) Class-wise Training Accuracy using MLP Architecture 2



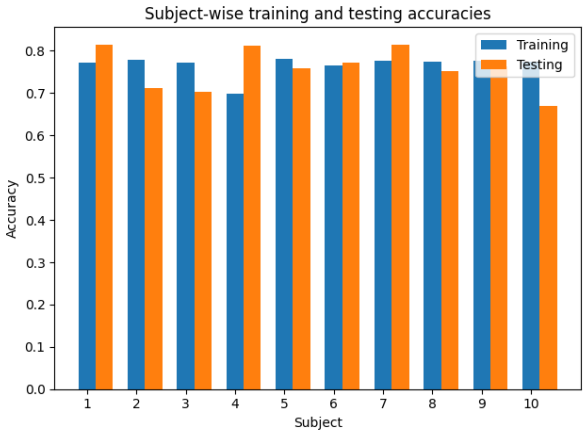
(b) Class-wise Testing Accuracy using MLP Architecture 2



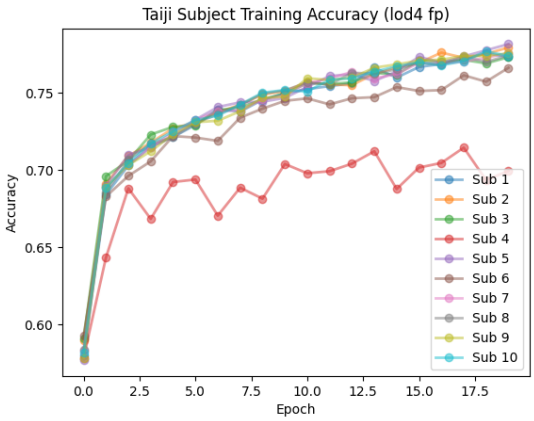
(c) Overall Training Confusion Matrix for MLP Architecture 2



(d) Overall Testing Confusion Matrix for MLP Architecture 2



(e) Subject-wise Training Accuracy for MLP Architecture 2



(f) Training Accuracy Curve for MLP Architecture 2

Figure 6: Classification Results of using MLP Architecture 2 on Taiji lod4

3.2 Part 2: Wallpaper Group Classification using CNN

In this section, we will discuss three architectures: Baseline CNN, Modified CNN 1, and Modified CNN 2. The results are presented in Tables 2 and 3.

Let us consider the regular Wallpaper group dataset. We observe that the baseline architecture quickly converges in 10 epochs of training due to its simple design. We use a batch size of 1024 and a learning rate of 0.002.

The modified architectures are more complex than the baseline architecture and have more parameters thus, requiring more epochs for the training and validation loss to converge. Based on Table 2, Modified CNN 1 clearly outperforms the baseline architecture. Note that the training curve is slightly noisy due to the use of the Adam optimizer in exchange for faster training time. Wallpaper groups P4G and P1 have the lowest testing accuracies.

For Modified CNN 2, we observe that the training accuracy curve is much more stable than Modified CNN 1. Clearly, Modified CNN 2 has the best accuracy. P4G and P6M have relatively lower testing accuracies compared to other wallpaper groups. However, the testing accuracy of P4G using Modified CNN 2 is still better than the corresponding accuracy using Modified CNN 1.

Now, let us consider the challenge dataset. The solution to improve the accuracy of the models on the challenge dataset is to use the data augmentation techniques described in the Methods section. It is clear that the baseline architecture, which was quick to converge to a high accuracy was overfitted. Also, we see that the use of data augmentation has improved the accuracy of CNN2 by 4%.

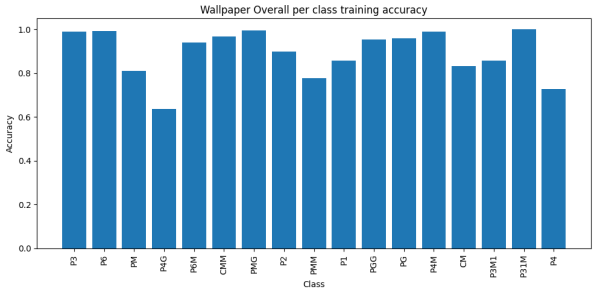
Figures 12 and 13 are visual representations of how CNN Architecture 2 has learned to identify Wallpaper patterns after augmentation. In Figure 13, we can see a well-defined cluster in the form of a ring. Thus, the representations in the first fully-connected layer are meaningful and not a result of overfitting.

Model	Dataset	Training Accuracy	Testing Accuracy	Training Class-wise Std	Testing Class-wise Std
Baseline CNN	Wallpaper Test	89.16%	91.265%	0.0869	0.1006
CNN Arch 1	Wallpaper Test	96.85%	95.06%	0.0346	0.0489
CNN Arch 2	Wallpaper Test	99.82%	96.85%	0.0028	0.0336

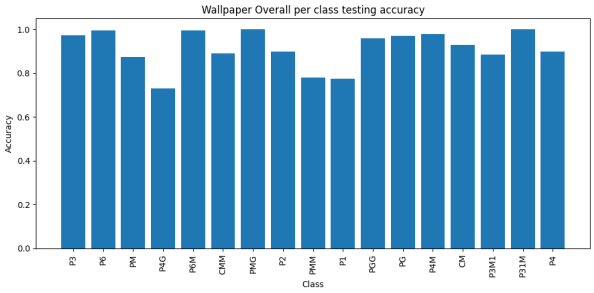
Table 2: Classification Results on Standard Wallpaper Test dataset without Data Augmentation

Model	Dataset	Training Accuracy	Testing Accuracy	Training Class-wise Std	Testing Class-wise Std
Baseline CNN	Wallpaper Test	36.68%	49.84%	0.1903	0.3727
Baseline CNN	Wallpaper Challenge	36.68%	11.41%	0.1903	0.37
CNN Architecture 2	Wallpaper Test	82.81%	80.44%	0.0748	0.1883
CNN Architecture 2	Wallpaper Challenge	82.81%	15.84%	0.0769	0.1813

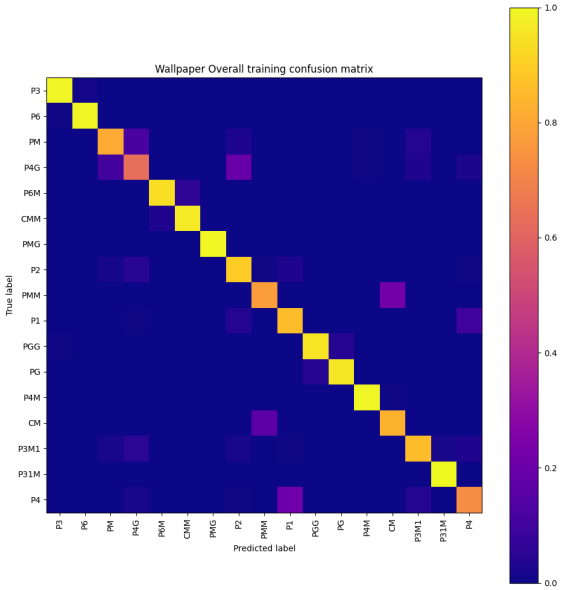
Table 3: Classification Results on Wallpaper Test and Challenge datasets using Data Augmentation



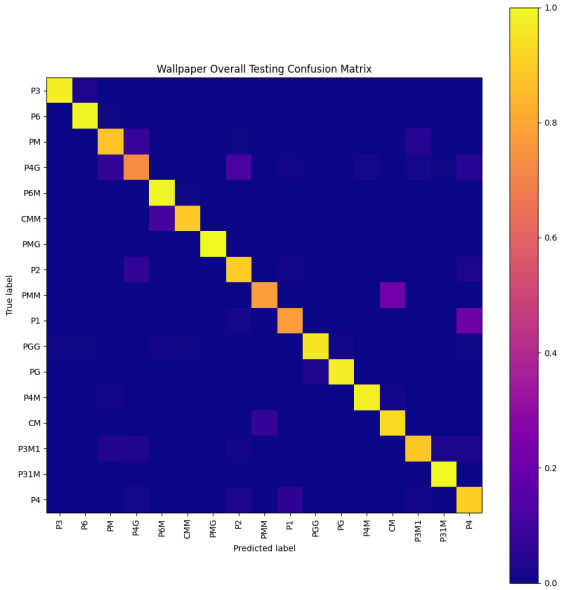
(a) Class-wise Training Accuracy using Baseline CNN Architecture



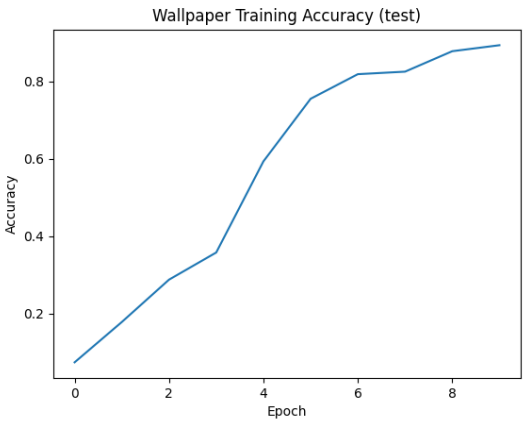
(b) Class-wise Testing Accuracy using Baseline CNN Architecture



(c) Overall Training Confusion Matrix for Baseline CNN Architecture

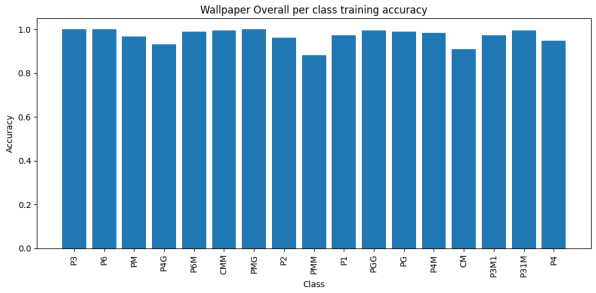


(d) Overall Testing Confusion Matrix for Baseline CNN Architecture

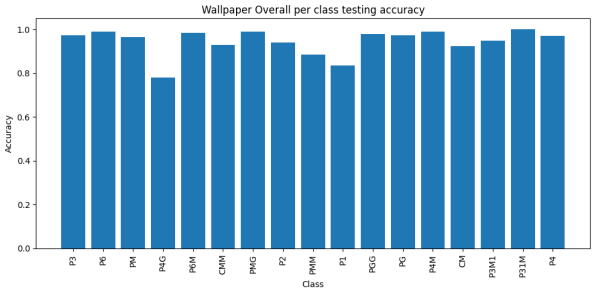


(e) Training Accuracy Curve for Baseline CNN Architecture

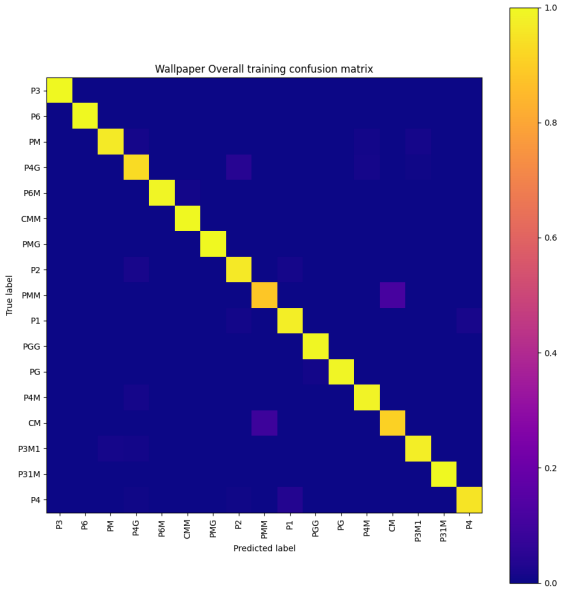
Figure 7: Classification Results of using Baseline CNN Architecture on Wallpaper Dataset without using Data Augmentation



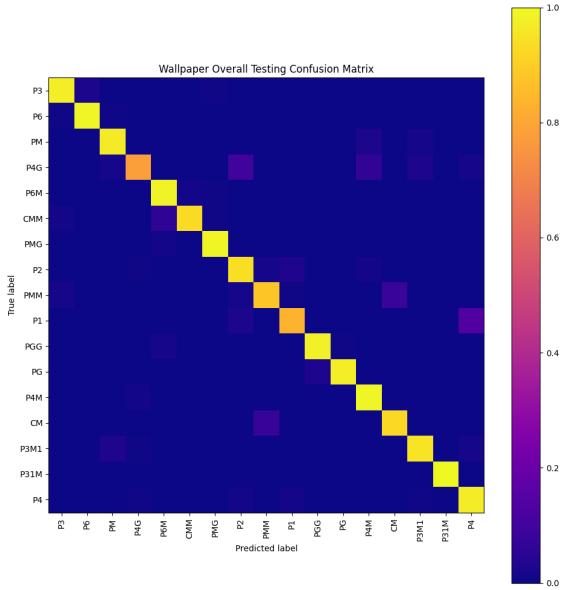
(a) Class-wise Training Accuracy using CNN Architecture 1



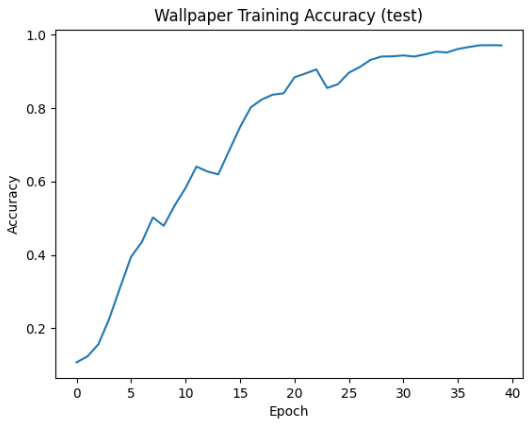
(b) Class-wise Testing Accuracy using CNN Architecture 1



(c) Overall Training Confusion Matrix for CNN Architecture 1

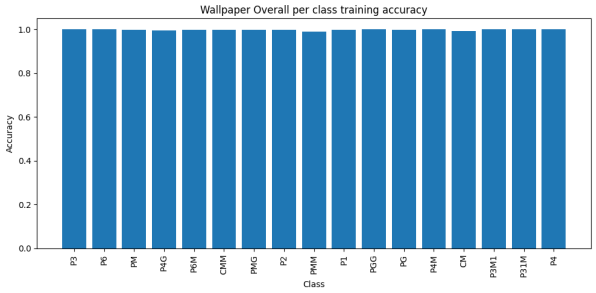


(d) Overall Testing Confusion Matrix for CNN Architecture 1

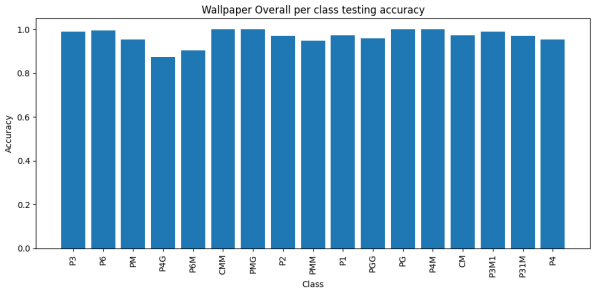


(e) Training Accuracy Curve for CNN Architecture 1

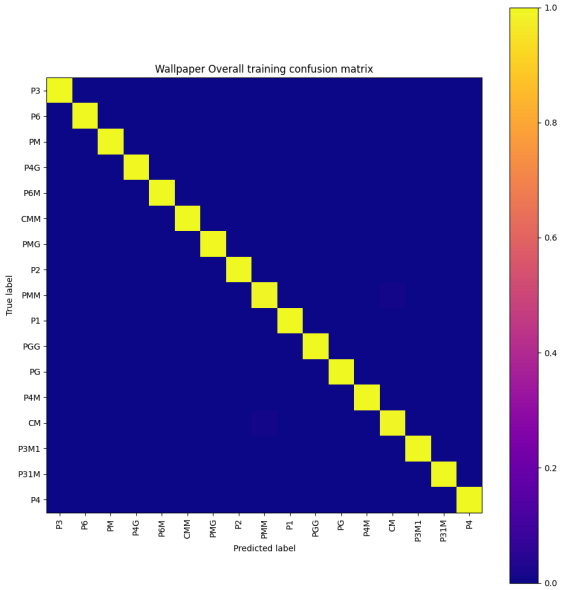
Figure 8: Classification Results of using CNN Architecture 1 on Wallpaper Dataset without using Data Augmentation



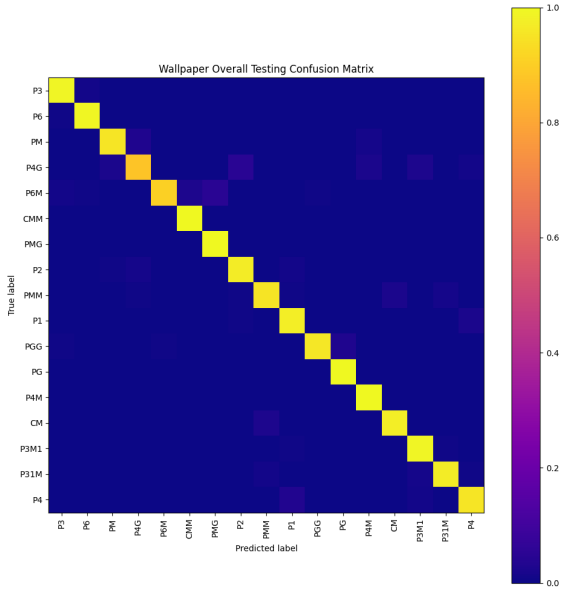
(a) Class-wise Training Accuracy using CNN Architecture 2



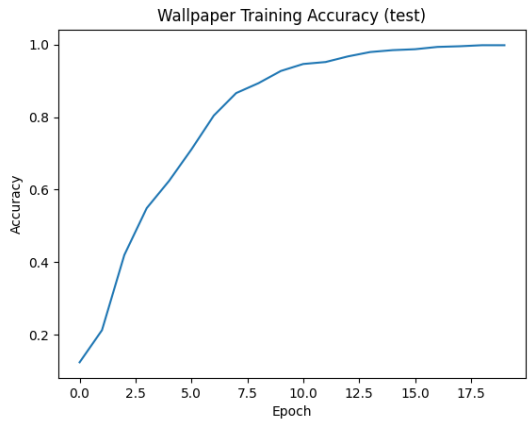
(b) Class-wise Testing Accuracy using CNN Architecture 2



(c) Overall Training Confusion Matrix for CNN Architecture 2

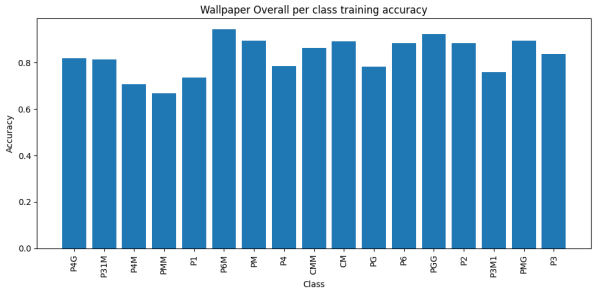


(d) Overall Testing Confusion Matrix for CNN Architecture 2

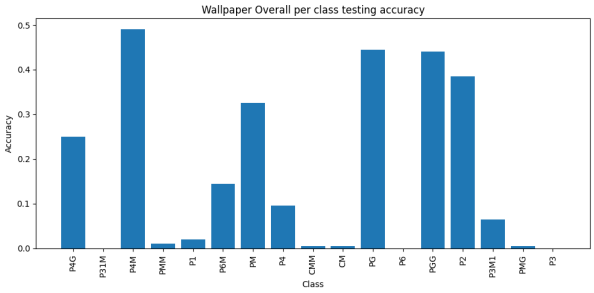


(e) Training Accuracy Curve for CNN Architecture 2

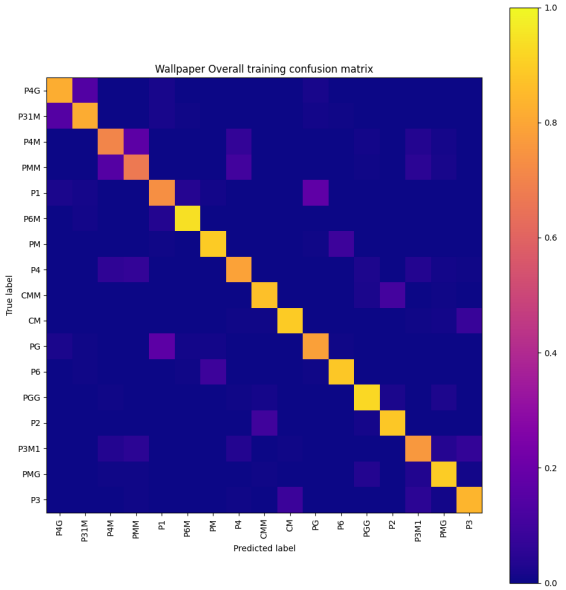
Figure 9: Classification Results of using CNN Architecture 2 on Wallpaper Dataset without using Data Augmentation



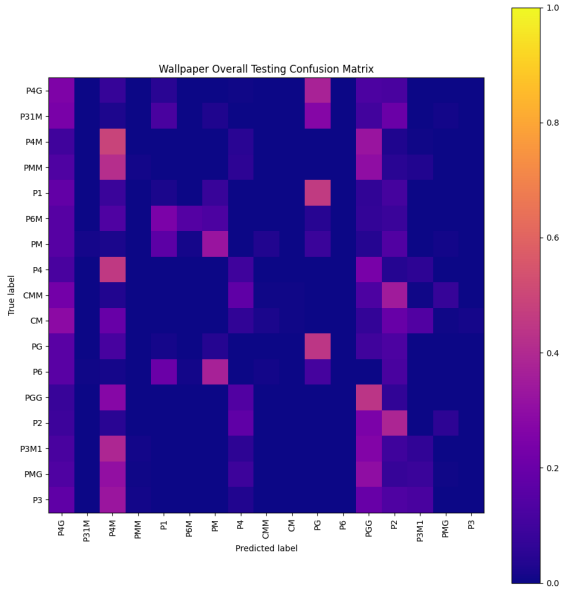
(a) Class-wise Training Accuracy using CNN Architecture 2



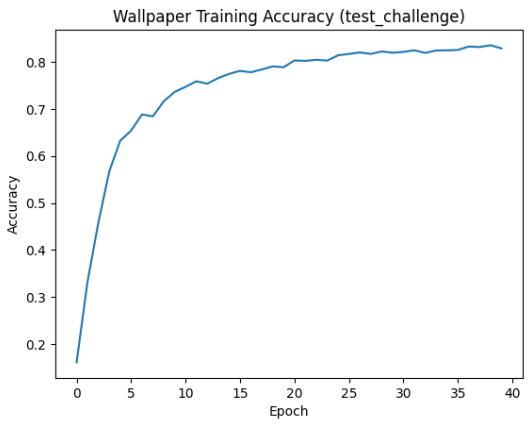
(b) Class-wise Testing Accuracy using CNN Architecture 2



(c) Overall Training Confusion Matrix for CNN Architecture 2

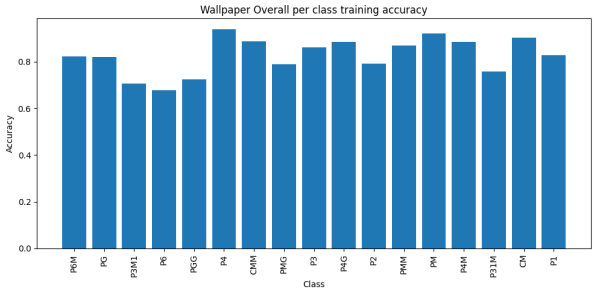


(d) Overall Testing Confusion Matrix for CNN Architecture 2

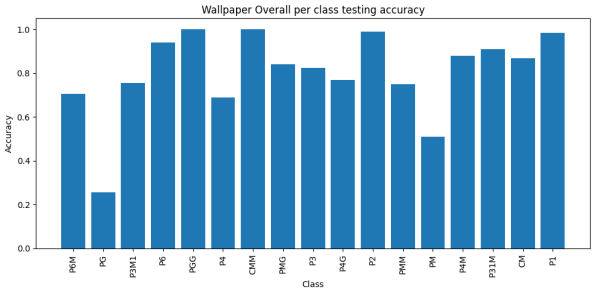


(e) Training Accuracy Curve for CNN Architecture 2

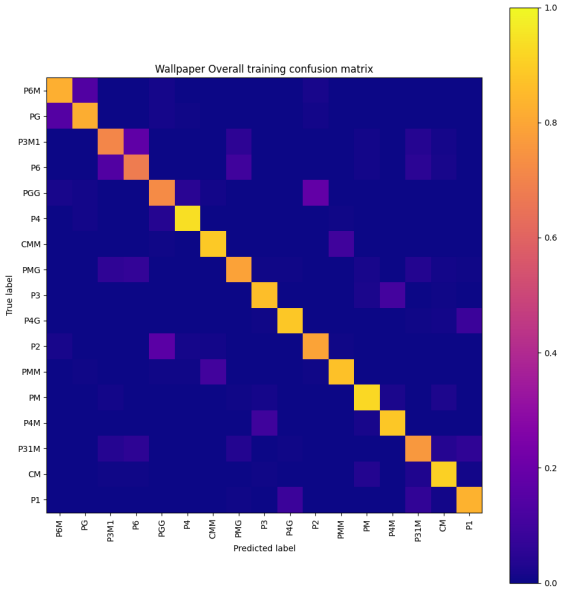
Figure 10: Classification Results of using CNN Architecture 2 on Challenge Wallpaper Dataset using Data Augmentation



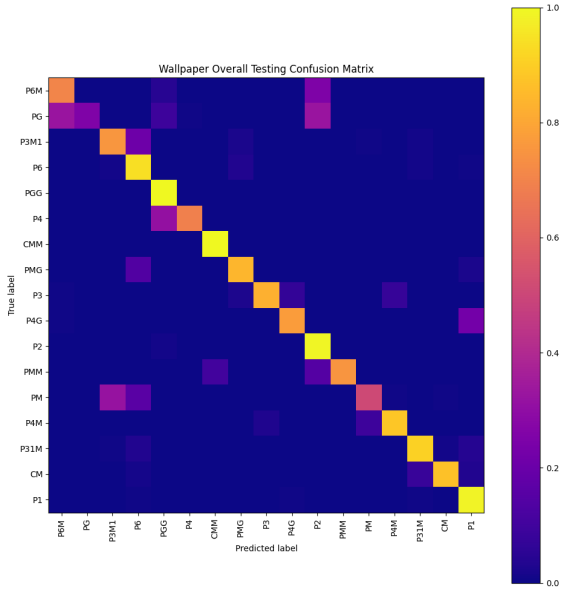
(a) Class-wise Training Accuracy using CNN Architecture 2



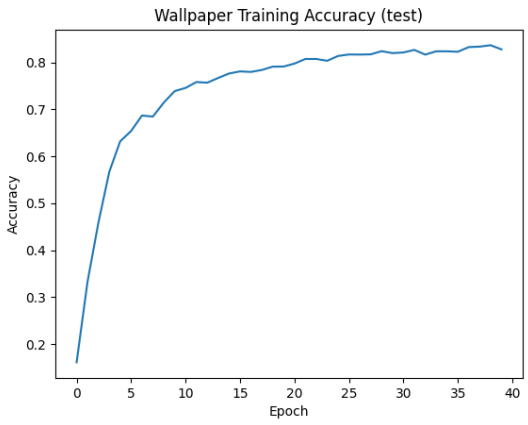
(b) Class-wise Testing Accuracy using CNN Architecture 2



(c) Overall Training Confusion Matrix for CNN Architecture 2

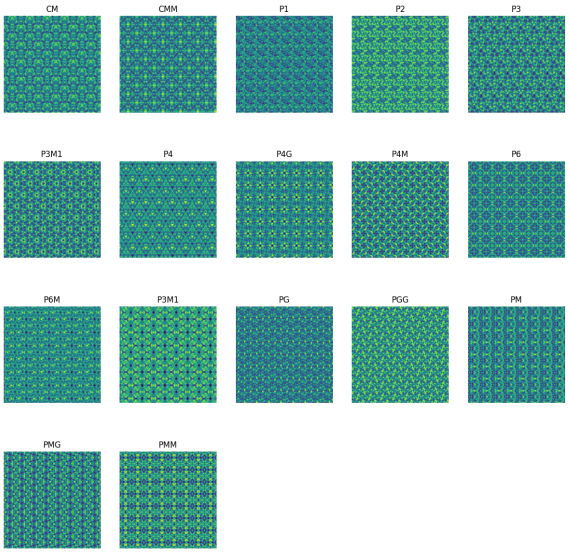


(d) Overall Testing Confusion Matrix for CNN Architecture 2

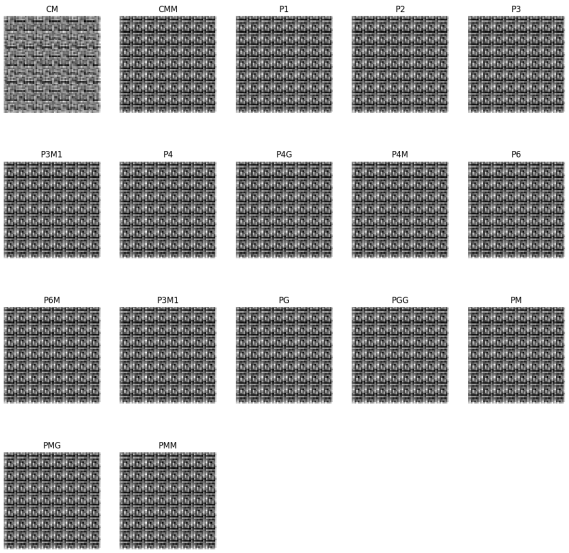


(e) Training Accuracy Curve for CNN Architecture 2

Figure 11: Classification Results of using CNN Architecture 2 on Test Wallpaper Dataset using Data Augmentation



(a) Original Wallpaper Images (Grayscale colormap)



(b) Feature Maps

Figure 12: Feature Map Visualization of all 17 Wallpaper Groups from the Second Convolution layer (1st map)

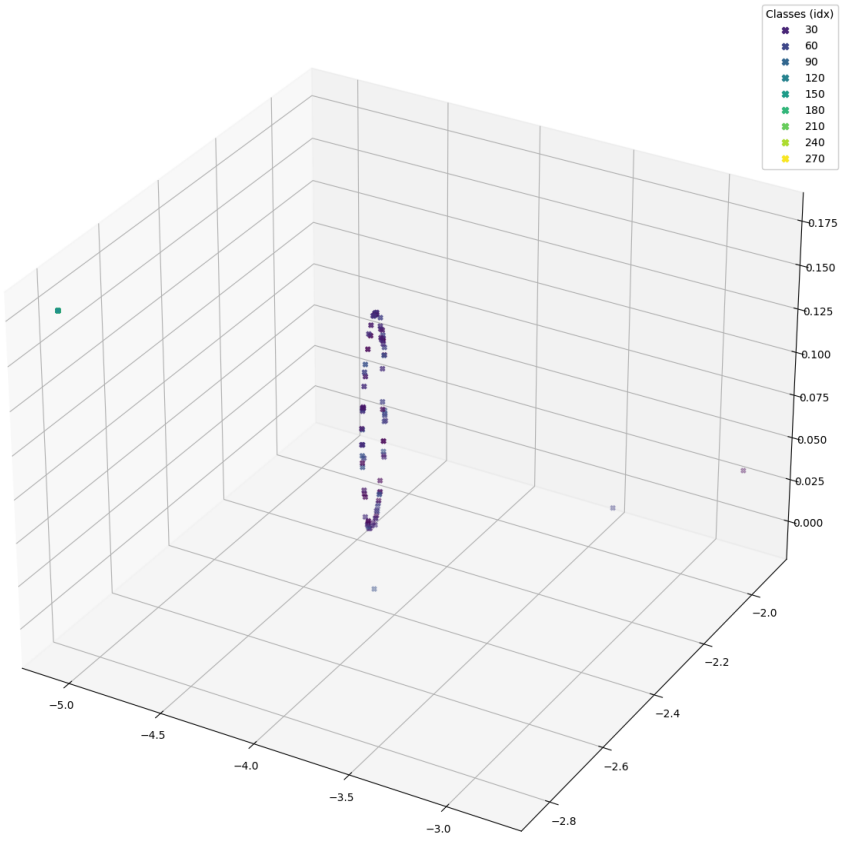


Figure 13: T-SNE Visualization of layer FC1 from Modified CNN Architecture 2 for 340 samples (20 samples from each class)

3.3 Transfer Learning and Design Space Exploration

In this section, we discuss the results of leveraging existing State-of-the-art CNNs to achieve the best possible accuracy on the challenge dataset. The test accuracies of the models on the Wallpaper challenge set are given in Figure 14.

It is clear that ResNet101 has the highest testing accuracy on the challenge dataset. However, Mobilenet still has relatively high accuracy with the fewest parameters and is also suitable for real-time inference on mobile devices [4].

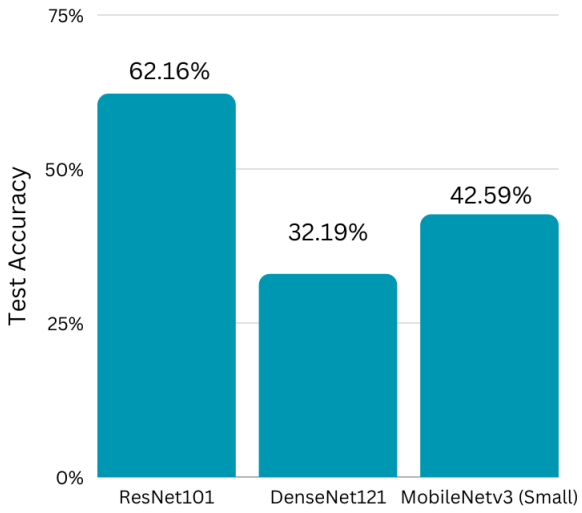


Figure 14: Test Accuracies of SOTA CNNs on the Challenge Dataset

Consider the number of trainable parameters given in Table 4. As shown, Transfer Learning was used to train the top layers of the Resnet and Densenet networks while keeping the backbone frozen. Resnet achieves the highest accuracy for this particular challenge dataset, however, the effectiveness of these models still depends on the compatibility of the dataset and Imagenet weights. The values shown in Table 4 account for the lower layers being frozen during transfer learning.

In the case of the Mobilenet model, the whole model can be trained from scratch with lesser resources for real-time inference.

Model	Number of Trainable Parameters	Transfer Learning	Average Inference Time Per Image (seconds)
ResNet101	20.91M	Yes	0.12
DenseNet121	6.97M	Yes	0.09
MobileNetv3 (Small)	**0.936M**	No	**0.019**

Table 4: Comparison of SOTA CNN Architectures used for the Challenge Dataset

3.3.1 ResNet101 Experiments

We have used a pretrained ResNet101 model and replaced the last fully connected layer with a new fully connected layer that has 17 output neurons instead of the standard 1000

neurons for ImageNet classification. The last 4 bottleneck layers, adaptive average pool layer, and the fully connected layer were unfrozen for training. After data augmentation, 34000 samples were used for training. We used a batch size of 256, a learning rate of 3e-4, no learning rate scheduler, and an Adam Optimizer with default betas for 35 epochs.

3.3.2 DenseNet121 Experiments

Two approaches were used while training the Densenet model. In the naive approach, we replace the fully connected layer with one suited for Wallpaaper group classification and use transfer learning to retrain the last few layers. However, the Densenet model training is quite slow (2x slower). To reduce the number of trainable parameters and improve GPU utilization, we add an Adaptive Average Pool layer before the last fully connected layer. The training time improves and the overall test accuracy is roughly the same. The overall test accuracy of the naive implementation is 32.14%.

We used a batch size of 256, a learning rate of 3e-4, no learning rate scheduler, and an Adam optimizer with default betas for 30 epochs.

3.3.3 MobileNetv3 (Small) Experiments

Mobilenet is a light model with several practical advantages, however, it is quite tricky to train. By adapting the training approach mentioned in [4], we use an RMSProp Optimizer with momentum (0.9), an initial learning rate of 0.1, a batch size of 64, weight decay of 1e-5, a high epsilon value of 0.0316 for improved training stability. We use the StepLR learning rate scheduler to decay the learning rate every 2 epochs with a gamma value of 0.973. After 40 epochs, the overall testing accuracy of the model on the challenge set converges to 28.7%. This was the best result obtained with different learning rates ranging from 1e-3 to 0.2.

We propose using a Cosine Annealing LR Scheduler (without restarts) for training the model. The learning rate is annealed periodically, which results in periods of high learing and periods of low learning. The periods when the learning rate is high help the model escape local minima and the low learning rate periods help the model converge to a global minimum.

We use a batch size of 64, Adam optimizer with default betas, an initial learning rate of 0.1, and a period of 5 epochs for the scheduler with a minimum learning rate of 1e-6 for a total of 65 epochs. Using this appraoch we are able to achieve 42.59% accuracy as shown in Figure 14.

Hardware Details: All training and experiments were performed on Nvidia A100 or T100 GPUs available through Google Colab. Only 1 GPU was used at a time. None of the experiments warranted split model or split batch training.

4 Conclusion

In conclusion, Neural Networks are powerful architectures for a variety of tasks in Computer Vision including image classification, segmentation, object detection detection, etc. We have performed image classification experiments to evaluate the effectiveness of various Neural Network architectures. While the MLP was useful in dealing with MoCap data, CNNs are more suitable for dealing with images. In addition to the basic architectures explored, a design space exploration using State-of-the-Art CNN architectures has shown that pretrained Resnet models can achieve the best accuracy on unseen Wallpaper patterns, but require significant computational power to train due to a large number of trainable parameters. Mobilenet is the optimal solution because it is compute-resource friendly and has better testing accuracy than Densenet which requires the longest training time.

References

[1] J. C. Peterson, J. T. Abbott, and T. L. Griffiths, "Evaluating (and improving) the correspondence between deep neural networks and human representations," 2018. 3

[2] M. Attarian, B. D. Roads, and M. C. Mozer, "Transforming neural network visual representations to predict human judgments of similarity," 2021. 3

[3] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," 2017. 3

[4] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019. 19, 20