Deep Learning : Convolutional Neural Networks

Skanda Bharadwaj

June 28, 2019

Contents

Abstract

The aim of this project is to build a convolutional neural net to train and classify the wallpaper dataset. The dataset consists of 17,000 images, 1,000 images per group, and each image containing a wallpaper pattern. The CNNs are trained to discover patterns in the wallpapers. The dataset is further augmented to increase the number of observations per class by a factor of 5 for a better classification. The results for augmented dataset are compared against the original dataset. Additionally, we also visualize the first layers of all the networks along with the t-SNE of the fully connected layers.

1 Introduction

In deep learning, a convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing images. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

2 Wallpapers Dataset

Wallpaper group is a two-dimensional repetitive pattern, based on the symmetries in the pattern. There are particularly characterised by subtle differences that may place similar patterns in different groups, while patterns that are very different in style, color, scale or orientation may belong to the same group.

The parameters of the wallpaper data set is as follows -

- Number of Classes : 17
- Dimension : 256 x 256 (single channel, greyscale image)
- Number of Training Observations: 17000
- Observations per class in training dataset:
  - Class 1 through Class 17 : 1000
- Number of Testing Observations: 17000
- Observations per class in testing dataset:
  - Class 1 through Class 17 : 1000

2.1 Data Augmentation

Data Augmentation is a technique commonly used in increasing image data set size for image classification task. It involves creating new images by transforming(rotate, translation, scaling, reflection) the ones in the data set. Each image in the dataset is augmented with four transformations. Each of them is briefly explained below.

2.1.1 Rotation

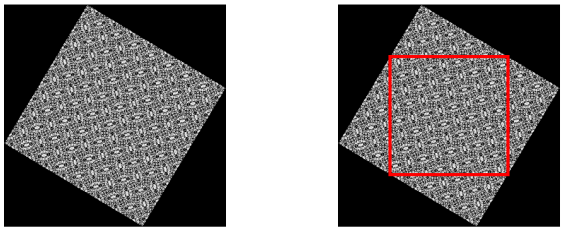
Each input image (256x256) is rotated by an arbitrary angle between $0^o - 360^o$. The resulting image is cropped such that the zero-padding is removed before resizing to 128x128. The cropping of images is done as follows -

- All non-zero vectors indices in both the dimensions are extracted.
- The non-zero vectors of equal length and length greater than 128 are extracted.
- Difference between two parameters is minimized in order to get the maximum of the image. First parameter is the difference between the indices of columns and rows. Second, size of the non-zero vector.

Algorithm –

---

for: *(all rows)* $\rightarrow$ *get non-zero vectors* end
for: *(all columns)* $\rightarrow$ *get non-zero vectors* end
for: *(i = No. of rows with non-zero vector)*
  for: *(j = No. of colusm with non-zero vector)*
    if: *(length(i) = length(j)) and (length(i) >128)*
      find $min((i - j) - length(i))$
    end if
  end for
end for



a b

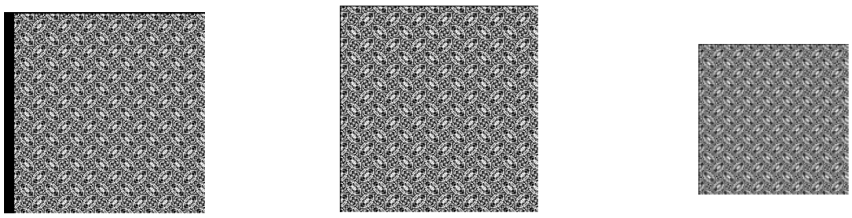
c d

Figure 1: Figure 1a represents the rotated image. In figure 1b, the red bounding box represents the amount of image to be cropped to remove the padding. Figure 1c represents the cropped image and figure 1d represents the image resized to 128x128.

Figure 1 represents the output of the above algorithm at intermediate stages. Figure 1a represents the rotate image. In figure 1b, the red bounding box represents the amount of image to be cropped to remove the padding. Figure 1c represents the cropped image and figure 1d represents the image resized to 128x128.

2.1.2 Translation

Image is translated in both the x and y axis by an arbitrary number of pixels chosen randomly in the range $(1, 20)$. The translated image is cropped to remove the zero-padding and then resized to 128x128.

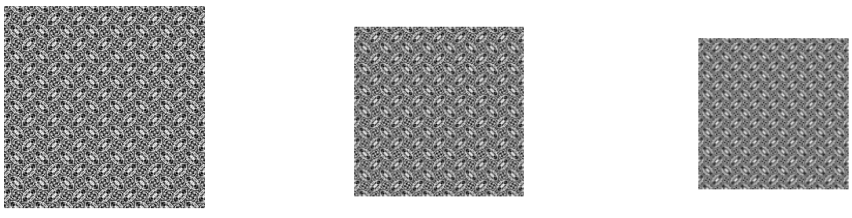


a b c

Figure 2: Figure 2a represents the translated image with zero-padding at the right end and top. Figure 2b, the cropped image. Figure 2c represents the image resized to 128x128.

2.1.3 Scaling

The original image of size 256x256 is randomly scaled in the range $(50\% - 100\%)$ and is resized to 128x128 pixels.



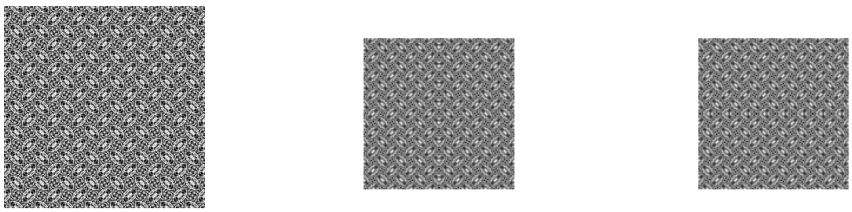
a b c

Figure 3: Figure 3a represents the original image of size 256x256. Figure 3b represents the image scaled to 64% of the original size. Figure 3c represents the image resized to 128x128.

2.1.4 Reflection

Reflection of the image is either done either vertically or horizontally. For any given input image, a random choice is made and the reflection is carried out on the original image of

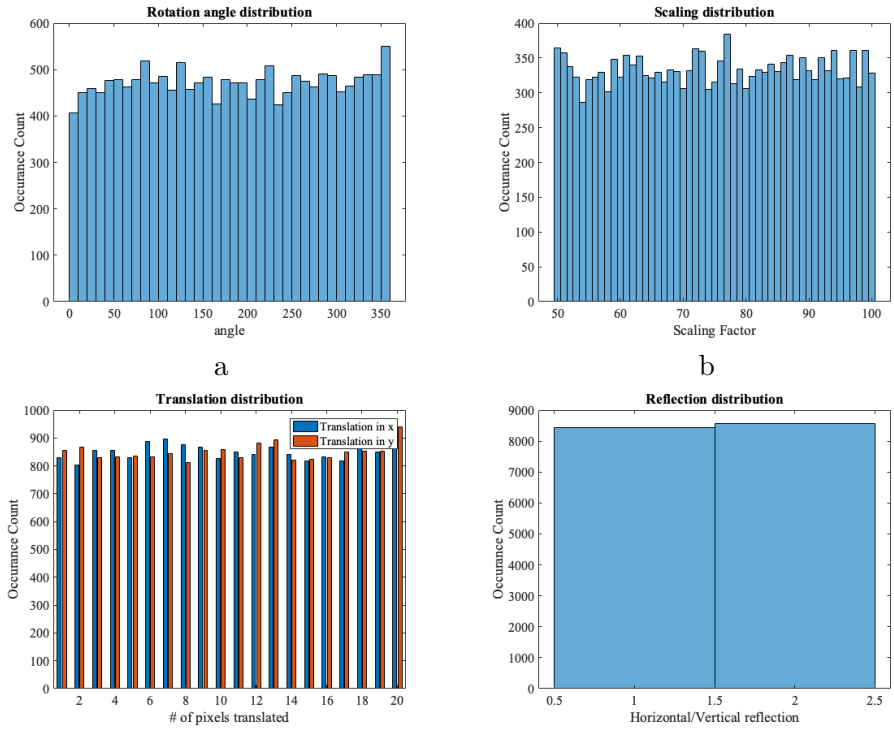
size 256x256. The image is then resized to 128x128 pixels.



a b c

Figure 4: Figure 4*a* represents the original image of size 256x256. Figure 4*b* represents the image reflected over the vertical axis and figure 3*c* represents the image reflected over the horizontal axis, both resized to 128x128.

Figure 5 represents histogram of distributions of augmentations.



a b c c

Figure 5: *a* represents the histogram of distribution of rotation angles. *b* represents the distribution of scaling factor. *c* represents the distribution of translating factors in x and y. *d* represents the distribution of choice of horizontal and vertical reflections.

Figure 6 represents the augmentation set for five different images belonging to five different classes.

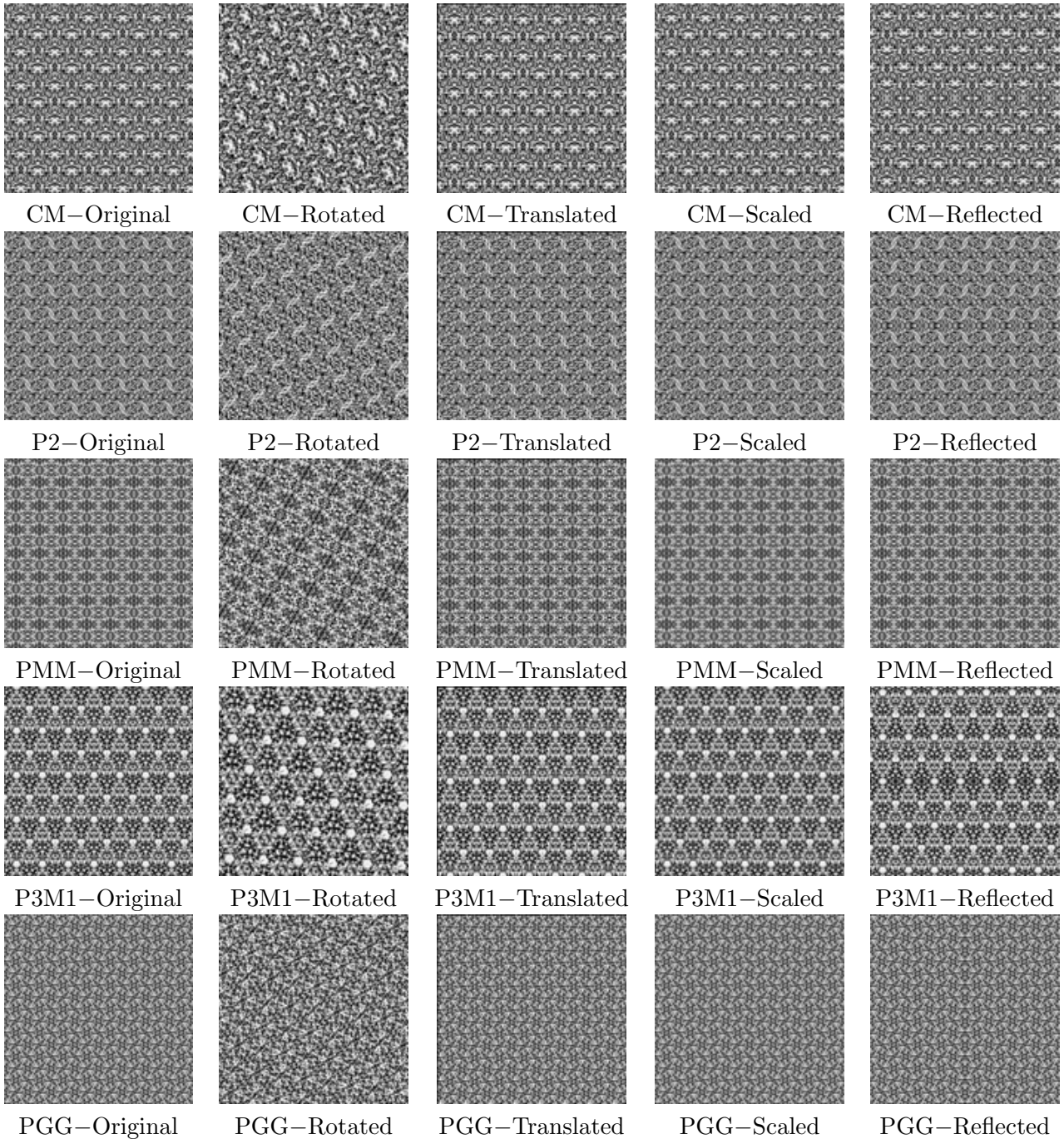


Figure 6: Augmentation data for 5 different images belonging to classes CM, P2, PMM, P3M1 and PGG of size 128x128.

3 Convolutional Neural networks

In this section we discuss the CNNs built to discover the patterns in the wallpaper dataset. Training of the wallpaper dataset was done in 3 different stages

- Stage 1: Training a CNN on the original dataset of 17,000 images.

- Stage 2: Training CNNs on augmented dataset of 85,000 images.
- Stage 3: Training original dataset on the best performing CNN from stage 2.

3.1 Stage 1: Training a CNN on the original dataset of 17,000 images

Firstly, a CNN was constructed for the training of original dataset of 17,000 images. We will call this network the default network. The architecture of the network was fairly simple with one convolutional layer, ReLu and a pooling layer. The details of the default network are given in Table 1.

Layers	Activation Shape	Activation Size	#Parameter
Input	(256, 256, 1)	65536	0
Conv1(f=5, s=2, p=2)	(128, 128, 20)	327680	520
MaxPool(s=2)	(64, 64, 20)	81920	0
FC1	(25, 1)	25	2048000
FC2	(17, 1)	17	426

Table 1: Architecture of the Default CNN

The entire network was for two different learning rates after concluding that the first network gave lesser accuracy. Table 2 summarizes outputs of the experiment.

Parameters	Network 1	Network 2
Learning Rate	5E-04	1E-04
No. of Epochs	10	10
Batch size	250	250
Training Accuracy(%)	74.46	81.61
Validation Accuracy(%)	70.12	75.65
Testing Accuracy(%)	–	75.15
Training Time (s)	304.75	160.30

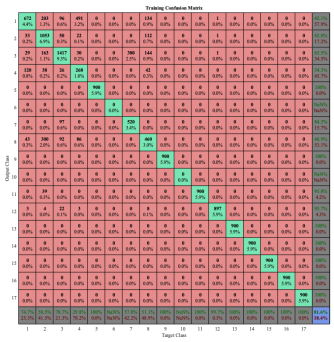
Table 2: Parameters and outputs of the 2 networks.

Figure 7 represents the confusion matrices of the above networks.

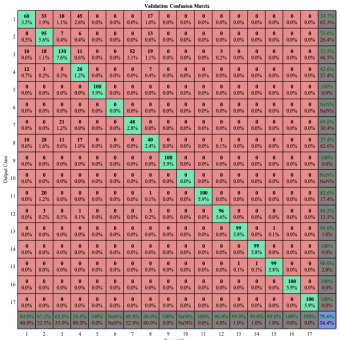
Figure 8 represents the graphs of loss and accuracy vs number of epochs for the two learning rates.

3.2 Stage 2: Training CNNs on augmented dataset of 85,000 images

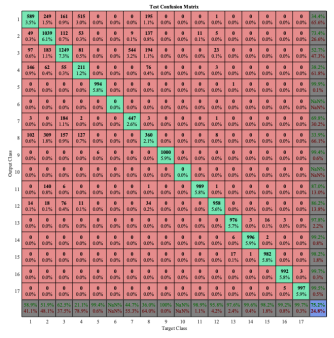
To check the effect of data augmentation, the augmented dataset was tested on the same CNN explained in the previous section. Later, different CNN networks – wide networks



a

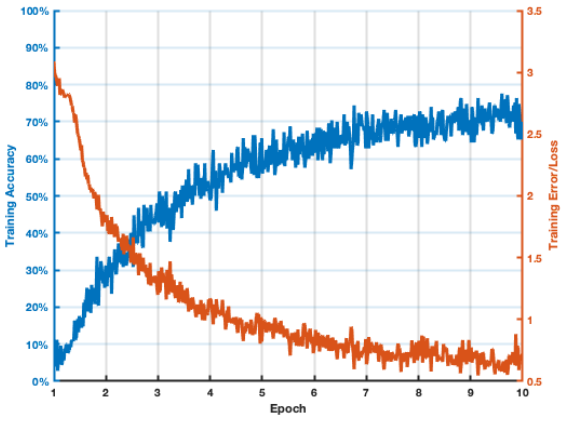


b

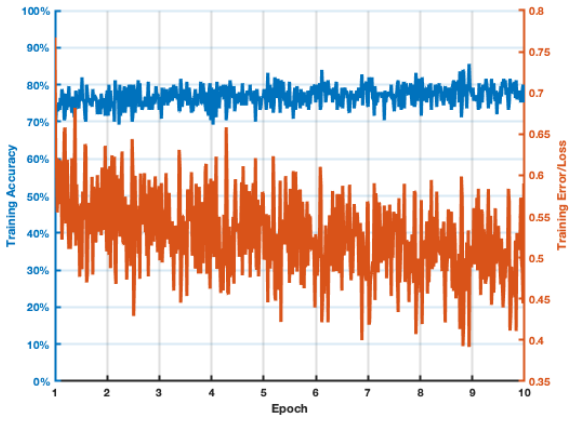


c

Figure 7: Figures a, b and c represent confusion matrices of training, validation and testing respectively.



a



b

Figure 8: Figures a and b represent loss and accuracy vs number of epochs for learning rate 5E-04 and 1E-04 respectively.

and skinny networks − were designed to improve the classification. This section gives details about all the different networks the augmented data was trained on.

3.2.1 Training augmented data on the Default CNN

The same network explained in the previous section was used with necessary changes to the input layer and few other parameters. The architectural changes are given below.

Layers	Activation Shape	Activation Size	#Parameter
Input	(128, 128, 1)	14884	0
Conv1(f=5, s=2, p=2)	(64, 64, 20)	81920	520
MaxPool(s=2)	(32, 32, 20)	20480	0
FC1	(25, 1)	25	512000
FC2	(17, 1)	17	426

Table 3: Architecture of CNN

The entire network was for two different learning rates after concluding that the first network gave a rather lesser accuracy. Table 2 summarizes outputs of the experiment.

Parameters	Network 1	Network 2
Learning Rate	5E-04	1E-04
No. of Epochs	20	20
Batch size	400	400
Training Accuracy(%)	73.10	80.63
Validation Accuracy(%)	60.06	62.11
Testing Accuracy(%)	–	61.40
Training Time (s)	5764.62	5752.46

Table 4: Parameters and outputs of the 2 networks.

Since the accuracies were not as descent enough compared to the previous run, confusion matrices and graphs have been excluded intentionally.

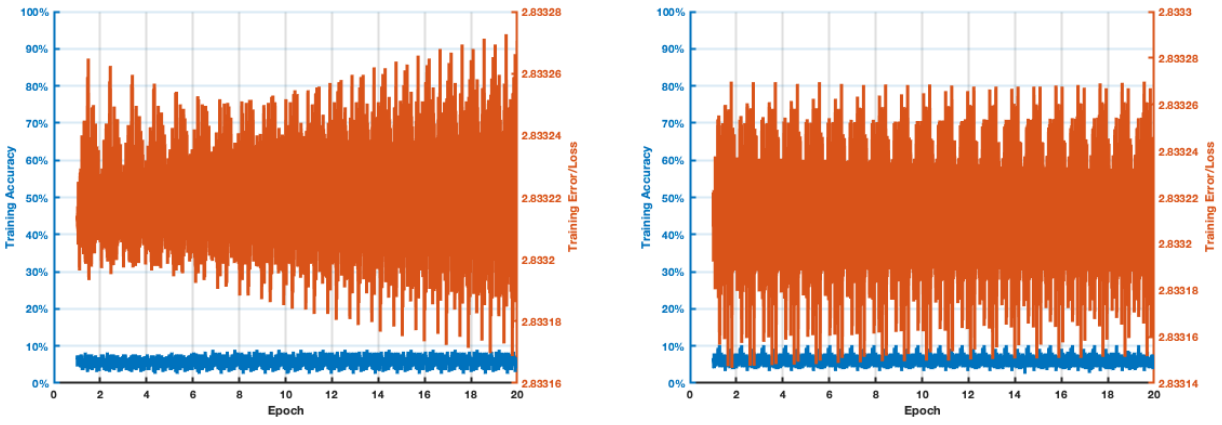
3.2.2 Training augmented data on skinny CNNs

The augmented dataset was then trained using the so-called skinny networks. These networks are characterized by more number of layers and less filters per layer. The architecture of the network is summarized below.

Layers	Activation Shape	Activation Size	#Parameter
Input	(128, 128, 1)	14884	0
Conv1(f=5, s=1, p=2)	(128, 128, 20)	327680	520
MaxPool(s=2)	(64, 64, 20)	81920	0
Conv2(f=5, s=1, p=2)	(64, 64, 30)	122880	780
MaxPool(s=2)	(32, 32, 20)	20480	0
Conv3(f=5, s=1, p=2)	(32, 32, 20)	20480	520
MaxPool(s=2)	(16, 16, 20)	5120	0
Conv4(f=3, s=1, p=1)	(16, 16, 40)	10240	400
MaxPool(s=2)	(8, 8, 20)	1280	0
Conv5(f=3, s=1, p=1)	(8, 8, 20)	1280	200
MaxPool(s=2)	(4, 4, 20)	320	0
FC1	(25, 1)	25	8000
FC2	(17, 1)	17	426

Table 5: Architecture of Skinny CNN 1

Figure 9 shows that the accuracy did not rise even after tweaking the learning rates. Just was the case for other skinny trials.



a b

Figure 9: Figures *a* and *b* represent loss and accuracy vs number of epochs for learning rate 5E-04 and 1E-04 respectively.

But the skinny network failed miserably with an accuracy of about 5.8%. The network did not converge even after several epochs. The network was tried to be tuned by tweaking parameters such as learning rate, number of filters, number of layers etc. Skinny network with 6 layers and 4 layers with adjusted learning rates failed just as miserably. Confusion matrices, plots and architectures of the failed networks have been excluded intentionally since lot of the values were NaNs.

3.2.3 Training augmented data on wide CNNs

The augmented dataset was then trained using the wide networks. These networks are characterized by more number of filters and less layers. The architecture of the network is summarized below.

Layers	Activation Shape	Activation Size	#Parameter
Input	(128, 128, 1)	14884	0
Conv1(f=5, s=1, p=2)	(128, 128, 40)	655360	1040
MaxPool(s=2)	(64, 64, 40)	163840	0
Conv2(f=3, s=1, p=1)	(64, 64, 60)	245760	600
MaxPool(s=2)	(32, 32, 20)	20480	0
FC1	(25, 1)	25	512000
FC2	(17, 1)	17	426

Table 6: Architecture of wide CNN

Table 7 summarizes the results of the wide network.

Parameters	Network 1	Network 2
Learning Rate	5E-04	1E-04
No. of Epochs	20	20
Batch size	400	400
Training Accuracy(%)	91.89	97.04
Validation Accuracy(%)	81.91	84.04
Testing Accuracy(%)	–	83.36
Training Time (s)	5764.62	5752.46

Table 7: Parameters and outputs of the 2 networks.

It was clearly observed that the accuracy increased with lowering the learning rate. Further, the network was tweaked at the architectural level to improve the accuracy. The architecture of the tweaked network is summarized in table 8.

Layers	Activation Shape	Activation Size	#Parameter
Input	(128, 128, 1)	14884	0
Conv1(f=5, s=2, p=2)	(64, 64, 70)	286720	1820
MaxPool(s=2)	(32, 32, 40)	40960	0
Conv2(f=6, s=2, p=2)	(16, 16, 60)	15360	2220
MaxPool(s=2)	(8, 8, 20)	1280	0
FC1	(25, 1)	25	32000
FC2	(17, 1)	17	426

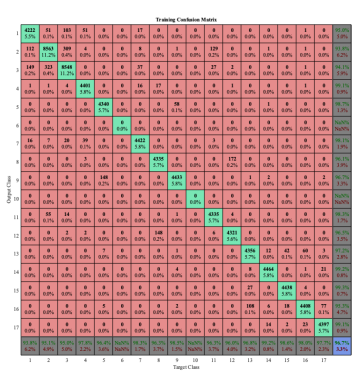
Table 8: Architecture of wide CNN

With the above changes to the network, keeping same learning rate, better results were obtained and are summarized below –

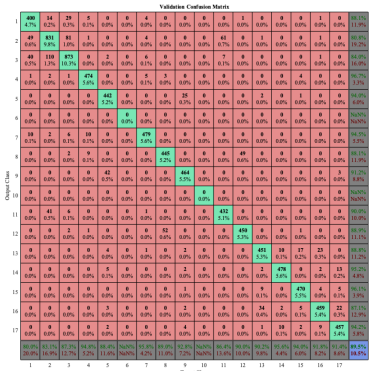
Parameters	Network 1	Network 2
Learning Rate	5E-04	1E-04
No. of Epochs	20	20
Batch size	400	400
Training Accuracy(%)	93.07	96.64
Validation Accuracy(%)	87.02	89.26
Testing Accuracy(%)	–	89.50
Training Time (s)	5764.62	5752.46

Table 9: Parameters and outputs of the 2 networks.

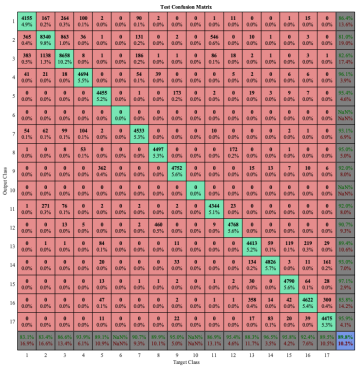
Confusion matrices of the training, validation and testing are represented in figure 10.



a



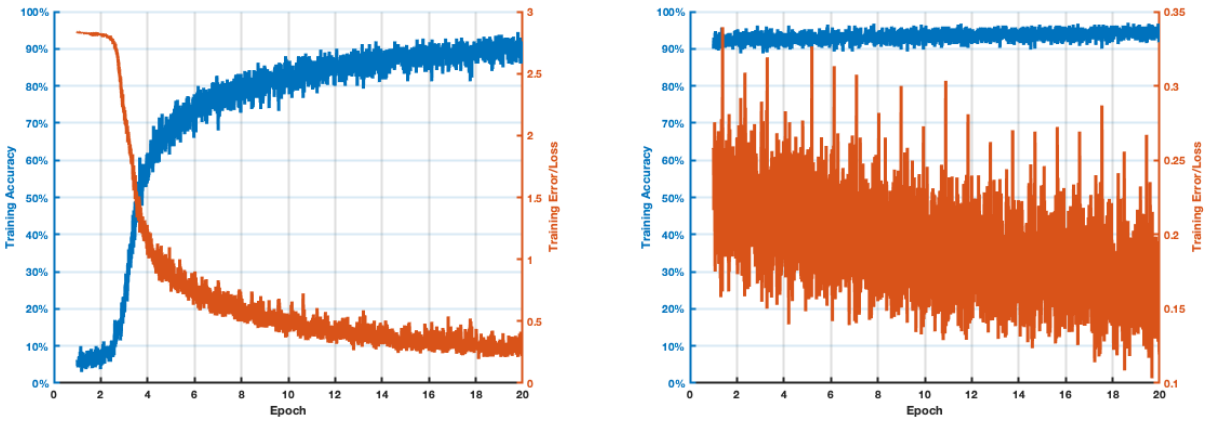
b



c

Figure 10: Figures a, b and c represent confusion matrices of training, validation and testing respectively.

Figure 11 represents the graphs of loss and accuracy vs number of epochs for the two learning rates.



a b

Figure 11: Figures *a* and *b* represent loss and accuracy vs number of epochs for learning rate 5E-04 and 1E-04 respectively.

3.3 Stage 3: Training original dataset on the best performing CNN from stage 2

Having trained the wide network for the augmented dataset, the same wide network was used on the original dataset. The architecture for the wide network is summarized in the below table

Layers	Activation Shape	Activation Size	#Parameter
Input	(256, 256, 1)	65536	0
Conv1(f=5, s=2, p=2)	(128, 128, 70)	1146880	1820
MaxPool(s=2)	(64, 64, 40)	163840	0
Conv2(f=6, s=2, p=2)	(32, 32, 60)	61440	2220
MaxPool(s=2)	(16, 16, 20)	5120	0
FC1	(25, 1)	25	128000
FC2	(17, 1)	17	426

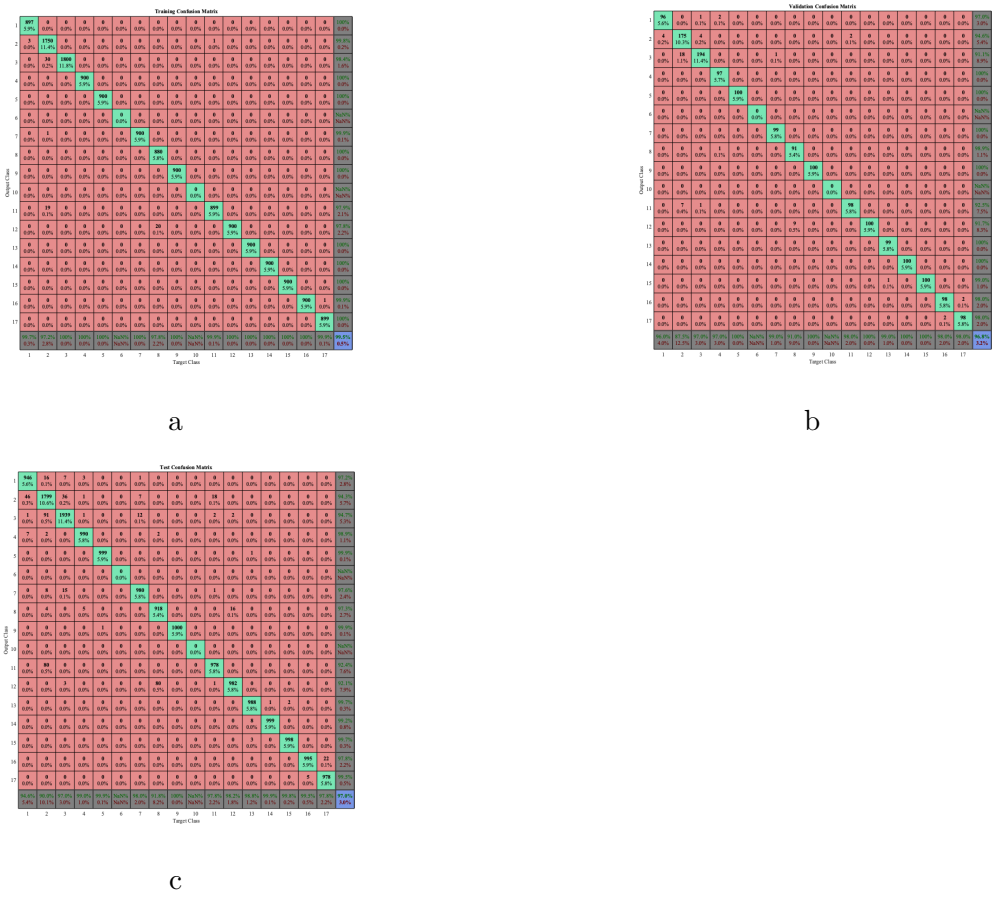
Table 10: Architecture of wide CNN

With the above changes to the network, keeping same learning rate, better results were obtained and are summarized below –

Parameters	Network 1	Network 2
Learning Rate	5E-04	1E-04
No. of Epochs	20	20
Batch size	400	400
Training Accuracy(%)	98.21	99.51
Validation Accuracy(%)	95	96.76
Testing Accuracy(%)	–	96.98
Training Time (s)	639.61	577.84

Table 11: Parameters and outputs of the 2 networks.

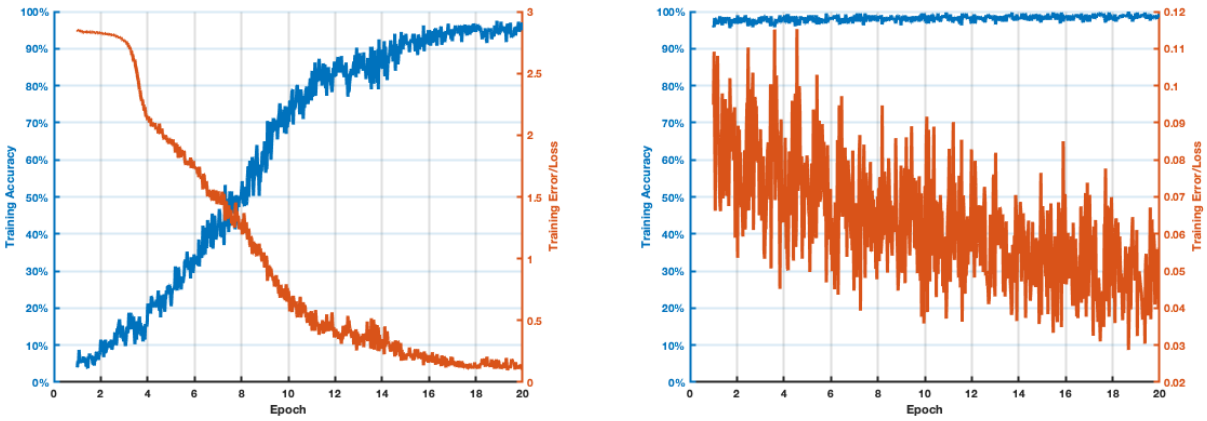
Confusion matrices of the training, validation and testing are represented in figure 12.



a b c

Figure 12: Figures a, b and c represent confusion matrices of training, validation and testing respectively.

Figure 13 represents the graphs of loss and accuracy vs number of epochs for the two learning rates.



a b

Figure 13: Figures *a* and *b* represent loss and accuracy vs number of epochs for learning rate 5E-04 and 1E-04 respectively.

3.4 Summary of Networks

Given below is the summary of all the networks that were tried.

Parameters	Default for Orignial Data	Default for Augmented Data
Learning Rate	1E-04	1E-04
No. of Epochs	10	20
Batch size	250	400
Training Accuracy(%)	81.61	80.63
Validation Accuracy(%)	75.65	62.11
Testing Accuracy(%)	75.15	61.40
Training Time (s)	160.30	5752.46
Total no. of network parameters	2,048,946	512,946
Total no. of network activations	475,178	117,326

Table 12: Parameters and outputs of the 2 networks.

Parameters	Skinny for Orignial Data	Skinny for Augmented Data
Learning Rate	1E-04	1E-04/1E-03/5E-04
No. of Epochs	10	20
Batch size	250	400
Training Accuracy(%)	5.66	5.88
Validation Accuracy(%)	4.59	5.88
Testing Accuracy(%)	5.56	5.88
Training Time (s)	160.52	7570.95
Total no. of network parameters	–	10,420
Total no. of network activations	–	606,606

Table 13: Parameters and outputs of the 2 networks.

Parameters	Wide for Orignial Data	Wide for Augmented Data
Learning Rate	1E-04	1E-04
No. of Epochs	20	20
Batch size	400	400
Training Accuracy(%)	99.51	96.64
Validation Accuracy(%)	96.76	89.26
Testing Accuracy(%)	96.98	89.50
Training Time (s)	577.84	5752.46
Total no. of network parameters	132,466	36,466
Total no. of network activations	1,442,858	359,246

Table 14: Parameters and outputs of the 2 networks.

4 Transfer Learning using AlexNet

Further to the networks built in this project, AlexNet was considered for classification of wallpaper dataset. Using a pre-trained network for classification with the modification of the fully-connected layer is called transfer-learning. Figure 14 represents the architecture of the original AlexNet.

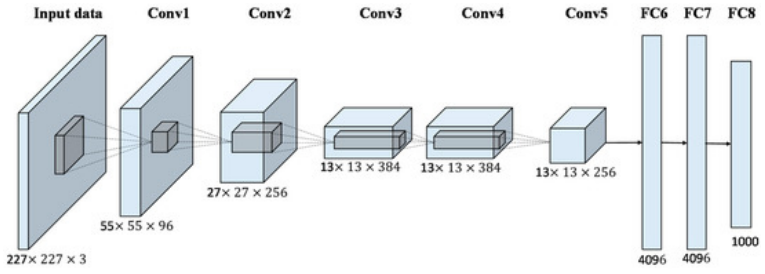
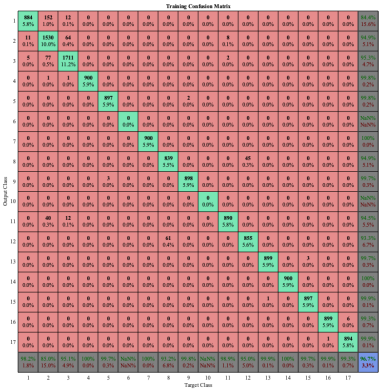
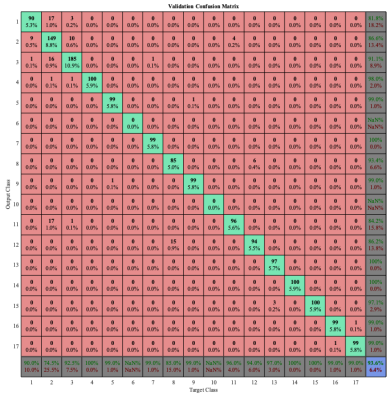


Figure 14: Architecture of the original AlexNet.

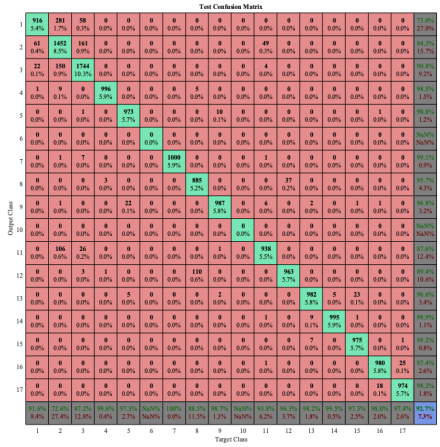
Images from both original dataset of size 256x256 and the augmented dataset of size 128x128 were resized to the input size of the AlexNet. The fully-connected layers of AlexNet (FC6(4096), FC7(4096), FC8(1000)) were changed to the FC6(4096), FC7(4096), FC8(17). Further, AlexNet was trained for the wallpaper dataset for 2 different learning rate for both original dataset and the augmented datasets. Tables 15 summarizes the results of AlexNet in original wallpaper dataset. While, figure 15 represents the confusion matrices for training, validation and testing, figure 16 represents the accuracy and loss over epochs. After a couple of tweaks, it was found that learning rate 1E-04 worked better than other learning rates such as 3E-04, 5E-05. Due to limitation of time, exhaustive tuning was not considered.



a



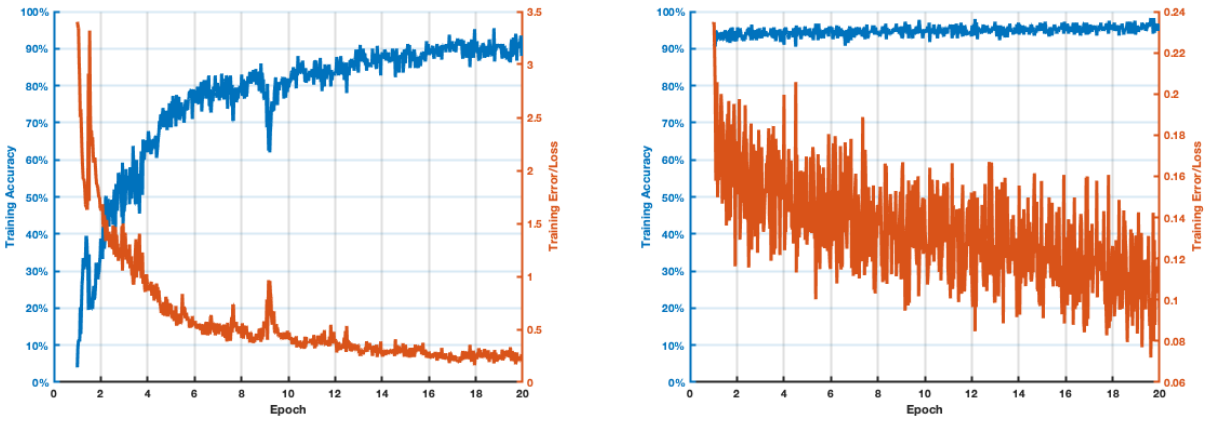
b



c

Figure 15: Figures a, b and c represent confusion matrices of training, validation and testing respectively for AlexNet on original dataset.

The same experiment was conducted on the augmented dataset and given below are the results.



a b

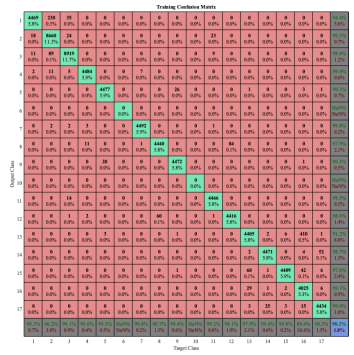
Figure 16: Figures *a* and *b* represent loss and accuracy vs number of epochs for learning rate 5E-04 and 1E-04 respectively for AlexNet on original dataset.

Parameters	Network 1	Network 2
Learning Rate	5E-04	1E-04
No. of Epochs	20	20
Batch size	400	400
Training Accuracy(%)	91.36	96.66
Validation Accuracy(%)	88.71	93.53
Testing Accuracy(%)	–	92.65
Training Time (s)	2403.10	2475.43

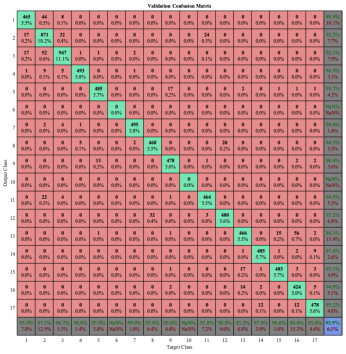
Table 15: Parameters and outputs of AlexNet on original dataset.

Parameters	Network 1	Network 2
Learning Rate	5E-04	1E-04
No. of Epochs	20	20
Batch size	400	400
Training Accuracy(%)	95.20	98.19
Validation Accuracy(%)	91.24	93.86
Testing Accuracy(%)	–	94.02
Training Time (s)	11776.19	14432.26

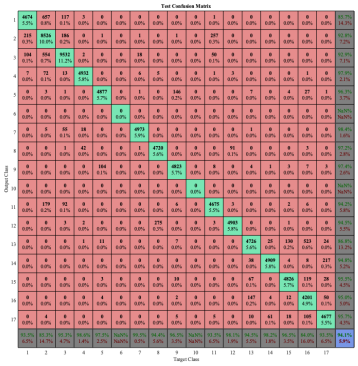
Table 16: Parameters and outputs of AlexNet on augmented dataset.



a

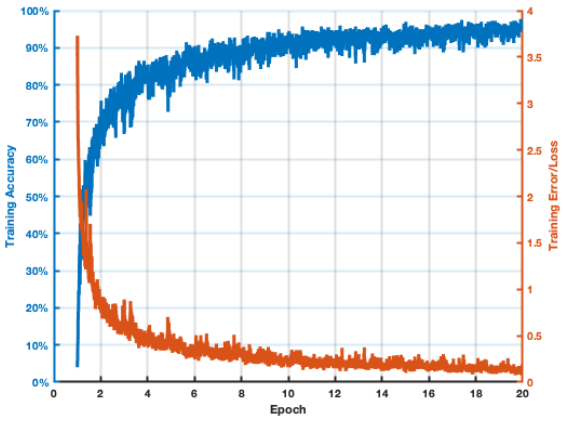


b

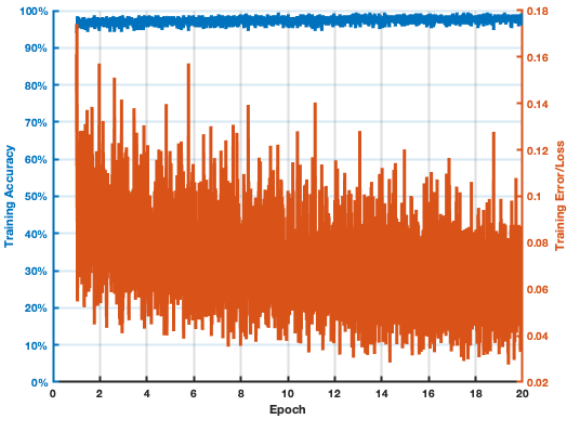


c

Figure 17: Figures a, b and c represent confusion matrices of training, validation and testing respectively for AlexNet on augmented dataset.



a



b

Figure 18: Figures a and b represent loss and accuracy vs number of epochs for learning rate 5E-04 and 1E-04 respectively for AlexNet on augmented dataset.

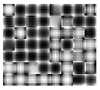
5 Visualization

5.1 First Layer Filter Visualization

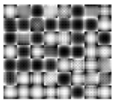
In order to understand the learning of CNN, the first layers of the all of the networks were visualized. Figures 19 represents the filters of all the networks.

Default network
on original dataset

Default network
on augmented dataset



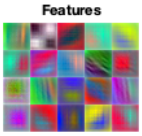
Wide network
on original dataset



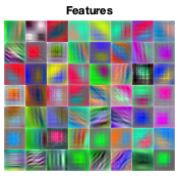
Wide network
on augmented dataset

Skinny network
on original dataset

Skinny network
on augmented dataset



Alex network
on original dataset

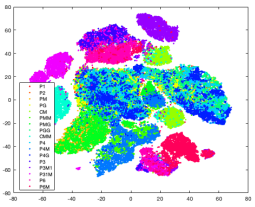


Alex network
on augmented dataset

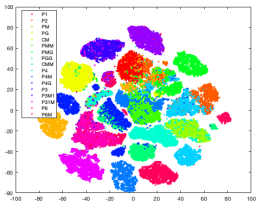
Figure 19: Figures above represent the visualizations of the first layer filters of all the networks on both original and augmented dataset.

5.2 t-distributed Stochastic Neighbour Embedding

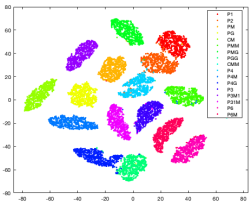
t-Distributed Stochastic Neighbour Embedding (t-SNE) is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. t-SNE was visualized for the default and wide networks. Figure 20 represents the t-SNE of the three variants. t-SNE was not performed on skinny network due to its miserable performance on the data.



Default network
on original dataset



Wide network
on augmented dataset



Wide network
on original dataset

Figure 20: Figures above represent the visualizations of the first layer filters of all the networks on both original and augmented dataset.

It can be seen that, at first, the default networks separation of the data is not quite distinct. But when a wide network was used on augmented data, the separation got better due to more exhaustive learning. Further ahead, when the same wide network was used to learn the original data, the separation is at its best and as a result, the accuracy for this network is better than all other networks as shown in Table 11.

6 Conclusions and Inferences

Convolutional neural nets are extremely handy compared to the conventional classification methods especially when the the number of classes are high. This project illustrated the efficacy of CNNs for image classification. It also helped in understanding the designing procedure of the network, such as constructing a convolutional layer after the preceding layers. Taking into account the strides, padding and the image size that is output by each of the layer. This in turn helps to calculate the number of activation and the number of parameters of the entire network. Thus, this project was an illustration of comprehensively designing and training deep learning networks.

When it comes to accuracy and loss of the networks, lower the loss, better the model (unless the model has over-fitted to the training data). The loss is calculated on training and validation and its interpretation is how well the model is doing for these two sets. Unlike accuracy, loss is not a percentage. It is a summation of the errors made for each example in training or validation sets. As long as the loss is decreasing the model is getting better. Although loss and accuracy have inverse proportionality relationship, accuracy is the summation of zero-one errors whereas loss is the summation of floating point numbers. Therefore, 0.001% decrease in the loss does not necessarily mean 0.001% increase in the accuracy.

References

[1] https://lvdmaaten.github.io/tsne/

[2] https://res.mdpi.com/remotesensing/remotesensing-09-00848/article_deploy/html/images

[3] https://en.wikipedia.org/wiki/Convolutional_neural_network