Recognition of Handwritten Math Symbols Using Deep Learning

Contents

List of Figures

List of Tables

1. INTRODUCTION

Many researchers have worked on the recognition of handwritten mathematical symbols and expressions [1][2][3][4]. Character and digit recognition are very well-studied problems; the MNIST dataset is often used as a dataset to try out new machine learning models because it is so widely used [4]. Most academicians and researchers use pen and paper while solving a complex mathematical equation and later transform into digital manuscripts for publications. However, many tools are available to insert equations in softwares that creates documents like Microsoft Word, it is always better if a solution is available that directly converts the scanned handwritten mathematical equations to digital format for publications and presentations. Microsoft word has introduced a new feature of ink equation which lets a user write the equations digitally, however most academicians solve the long mathematical problems on pen and paper only. This project approaches towards the first step of identifying the mathematical expressions i.e. recognition of mathematical symbols from scanned images

This project is mainly focussed on the use of deep learning algorithms of image process to recognize the mathematical symbol which is available as scanned images. Convolutional Neural Networks or CNN have been proved to be the effective methodology for image processing. Each convolution layer identifies a specific feature in the image horizontal lines, vertical lines, edges, color, gradient orientation etc. Generally, the first convolutional layer identifies the low-level features such as color, gradient orientation etc. The next convolutional layers identify higher level features which gives the idea about the complete image. The convolutional layers are generally followed by a pooling layer which reduces the size of the image keeping the features intact. Pooling layers are added to reduce the computational time during the training of the image. The last layer is then feed into the fully connected networks with some hidden layers and the output layers with number of classes as number of nodes.

2. INPUT DATA

The dataset for scanned images of mathematical symbols have been acquired from Kaggle [5]. The data set available [5] contains over 100,000 images of mathematical symbols of different classes. The dataset available on Kaggle [5] contains 375,974 images of handwritten math symbols classified into 82 labels like +, -, cos, Δ, $\rightarrow$ etc. Each image is a 45x45 pixel grayscale image and hence not much pre-processing is required for this dataset. However, while going through some of the discussions on

Kaggle, it has been found that various images are exact matches on pixel-to-pixel level. The same user has also given a python script to delete the repeated images so that the sample space can be unique and false accuracies can be avoided. Hence, the pre-processing step will require running the script on available dataset to make the sample space unique.

After deleting the repeated images, the total available images are 83,501 consisting of 82 classes. Many of these classes like Delta contains 35 images, exists contains 4 images, for all symbol contains 4 images etc. Therefore, due to less training examples such classes will be omitted from the classification algorithms.

Training on 82 classes will require a deep architecture with large number of parameters which will require high computational requirements or the training time time will be too much. To overcome this problem, first the training size has been reduced, number of classes has been divided into various groups considering the similarity of images (similar types of images are put into one group like brackets, alphabets, Greek alphabets, numbers etc.). Moreover, Google Colab has been used to reduce the training time as the Google Colab uses GPU for training.

The input images consist of total 82 classes with the following distribution for each classes (after removal of duplicate images)

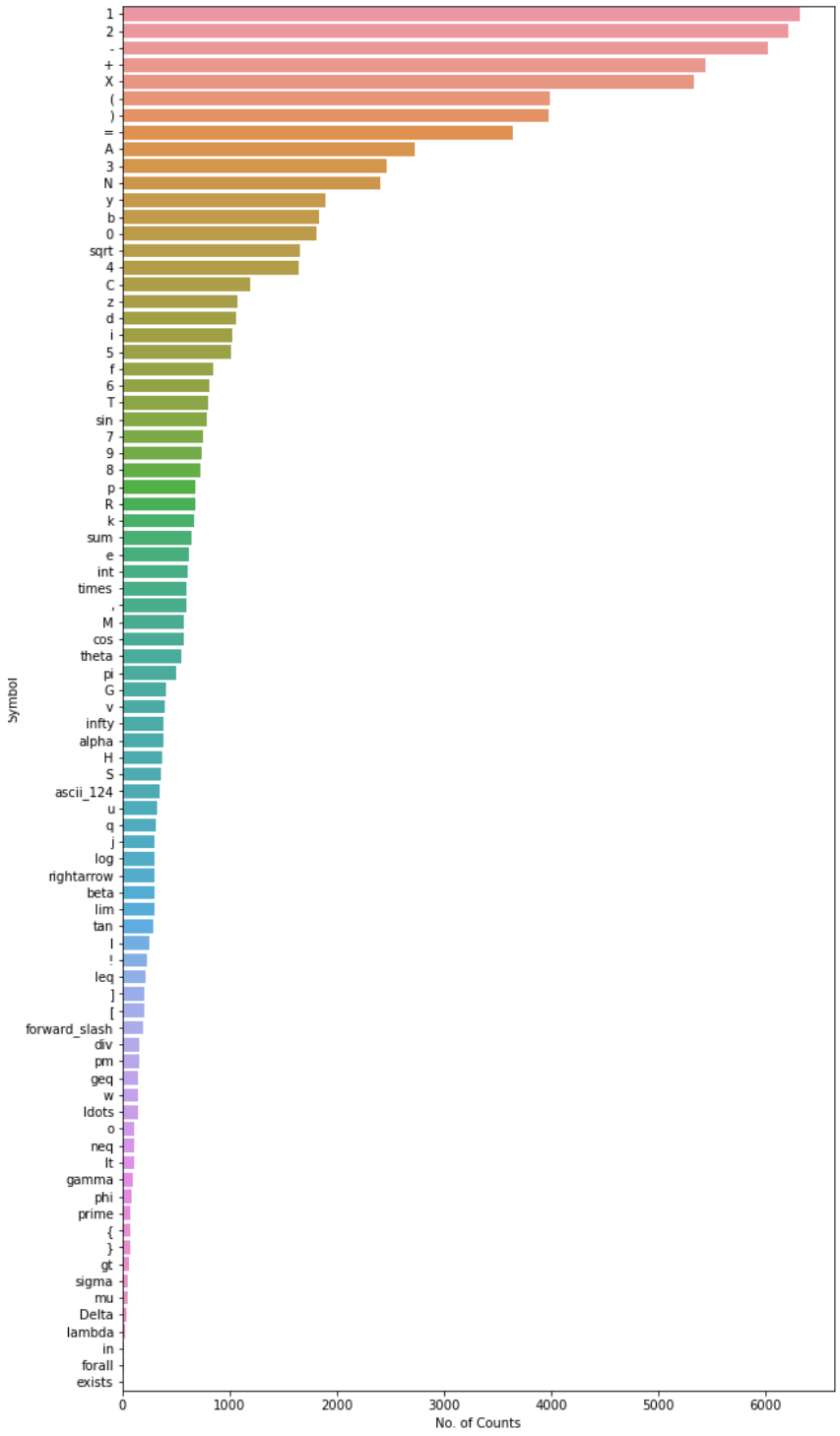


Figure 1: Distribution of Classes

3. DATA WRANGLING

The dataset considered for Capstone Three project contains huge number of images and it will be computationally difficult to train and fit the model on such a huge dataset. However, the model creation and training will be performed in Colab notebook, it still contains huge dataset. Hence this notebook will create the training and test dataset folders. The basic approach used to create training and test dataset is as follows:

- If a particular folder contains more than 200 images then first 100 images will be used as training data and next 100 will be used as test data.
- If the folder contains less than 200 images but greater than 50 images then it will be split in the ratio of 80% for training and 20% for test.
- If the folder consists of less than 50 images then it will not be considered for classification.

After creation of training and testing data further the classes have been grouped based on similarity because training on all 82 classes will take long time and following groups have been created:

Table 1: Details of groups created for training

Group Name	Symbols
Group-1	-, ,, !, (,), [,], {, }, +, =
Group-2	0 to 9
Group-3	A to M
Group-4	N to Z
Group-5	Alpha, beta, gamma, phi, pi, sigma, theta
Group-6	Ascii24, division, forward-slash, greater than equals, greater than, integral, idots, less than equals, less than, not equals, right arrow, square root, summation, multiplication, pluss minus
Group-7	Cos, infinity, limit, log, prime, sin, tan

4. MACHINE LEARNING EXPERIMENTS

After creation of training and test data for each group, the neural network architectures has to be decided for training of data. The images consist of 45 x45 pixel with grayscale values. There are several architectures reported in literatures for image classification problem like VGG16, ALexNet, SqueezeNet, Lenet etc. The architecture

of Lenet was first applied on similar kind of images like ours. Hence in this project also we have used architecture of Lenet but instead of using tanh as activation function relu is used since relu gives better accuracy. The Lenet architecture have been modified by modifying the parameters like stride, padding and number of nodes in a layer. Apart from modified Lenet architecture as reported in [4] has also been used for our experiments. The models used for training purpose are given below:

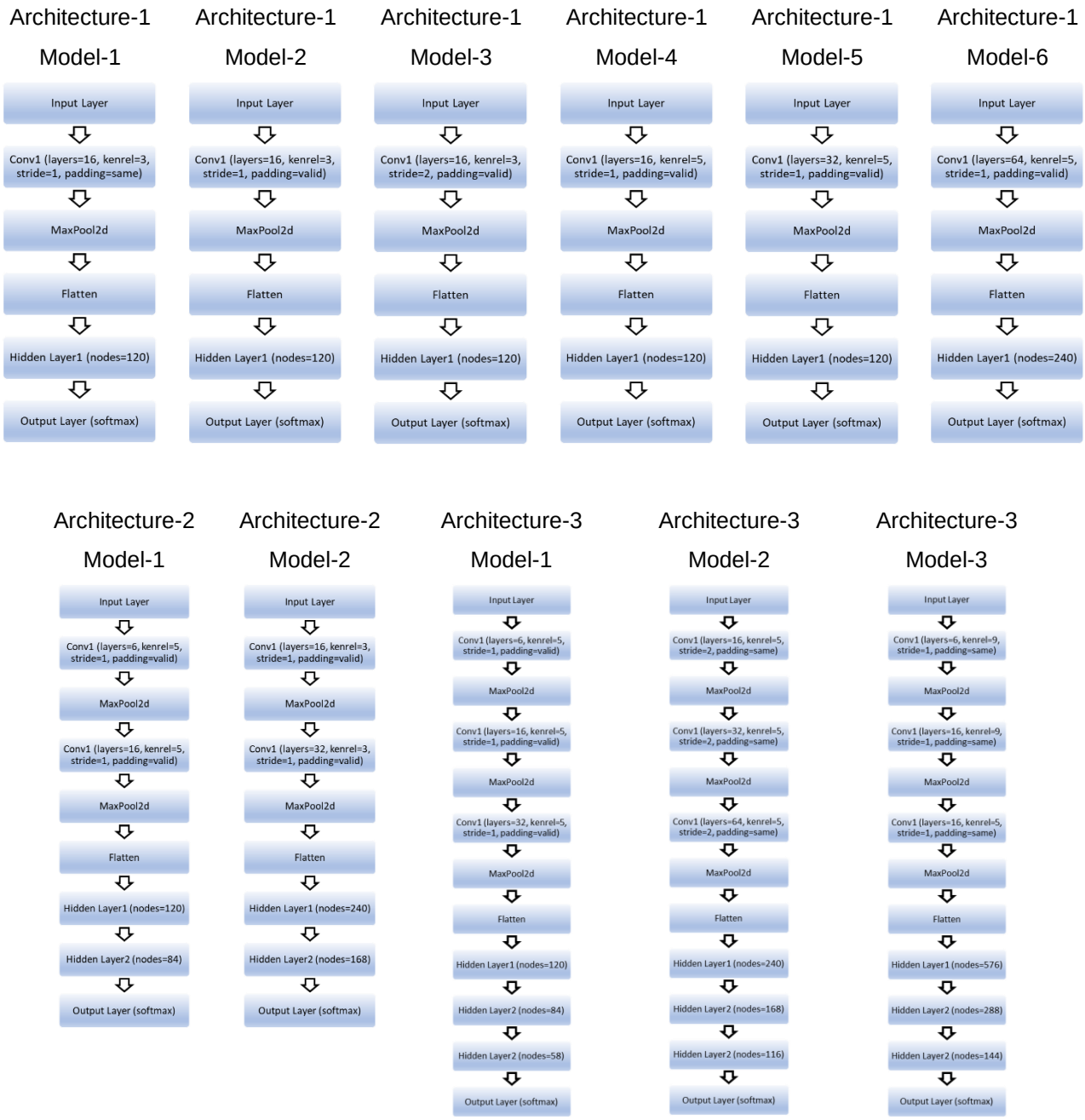


Figure 2: Details of Neural Network Architecture

The algorithms are implemented with open source programming language Python which offers various libraries for machine learning and deep learning. Tensorflow and Keras libraries are used for training of convolutional neural network algorithms. All the deep neural networks are tuned on their hyperparameters i.e learning rate and optimizer using a validation subset which is considered as 20% of the total training data. The hyperparameters chosen for tuning of deep learning models are given in Table 2. The algorithm with best parameters is chosen to calculate the accuracy on test data set.

Table 2: Hyperparameters and their values considered for tuning

S. No.	Hyperparameter	Values considered for tuning
1.	Optimizer	Stochastic Gradient Descent (SGD) Adam optimizer
2.	Learning Rate	0.01, 0.1

The accuracies for training dataset and test dataset with best hyperparameters are given in Table 3. The accuracy of the best model is highlighted in green.

Table 3: Training and test accuracies

Group	Architecture	Model	Optimizer	Learning Rate	Training Accuracy	Test Accuracy
group_1	arch_1	model_1	SGD	0.001	88.61%	86.64%
				0.01	88.61%	85.24%
			Adam	0.001	87.62%	86.75%
				0.01	84.65%	83.30%
		model_2	SGD	0.001	90.59%	85.56%
				0.01	92.57%	85.45%
			Adam	0.001	91.58%	85.45%
				0.01	82.67%	82.76%
		model_3	SGD	0.001	89.11%	87.18%
				0.01	88.61%	85.24%
			Adam	0.001	90.59%	87.18%
				0.01	92.57%	85.02%
		model_4	SGD	0.001	89.60%	86.31%
				0.01	88.12%	84.81%
			Adam	0.001	87.13%	86.75%
				0.01	89.60%	83.41%
		model_5	SGD	0.001	86.14%	86.96%
				0.01	87.62%	85.24%
			Adam	0.001	88.12%	86.96%

				0.01	87.13%	86.53%
		model_6	SGD	0.001	88.12%	86.85%
				0.01	88.61%	84.81%
			Adam	0.001	89.60%	85.56%
				0.01	40.59%	40.41%
	arch_2	model_1	SGD	0.001	87.13%	87.72%
				0.01	91.09%	89.66%
			Adam	0.001	88.12%	88.04%
				0.01	88.12%	88.90%
		model_2	SGD	0.001	89.11%	86.96%
				0.01	89.11%	88.04%
			Adam	0.001	92.08%	87.61%
				0.01	9.90%	10.78%
	arch_3	model_1	SGD	0.001	83.66%	87.50%
				0.01	89.11%	89.12%
			Adam	0.001	90.10%	91.92%
				0.01	83.66%	86.21%
		model_2	SGD	0.001	81.68%	85.34%
				0.01	89.11%	87.39%
			Adam	0.001	90.10%	88.69%
				0.01	9.90%	10.78%
		model_3	SGD	0.001	88.61%	91.16%
				0.01	93.56%	88.58%
			Adam	0.001	92.57%	91.27%
				0.01	9.90%	10.78%
group_2	arch_1	model_1	SGD	0.001	88.50%	84.50%
				0.01	88.00%	84.00%
			Adam	0.001	85.50%	83.70%
				0.01	81.00%	75.20%
		model_2	SGD	0.001	87.00%	83.10%
				0.01	86.50%	80.20%
			Adam	0.001	88.00%	82.50%
				0.01	81.00%	76.60%
		model_3	SGD	0.001	86.50%	83.00%
				0.01	84.50%	82.80%
			Adam	0.001	84.50%	82.70%
				0.01	89.00%	80.20%
		model_4	SGD	0.001	87.00%	81.40%
				0.01	85.00%	79.70%
			Adam	0.001	86.00%	82.70%
				0.01	83.50%	77.10%
		model_5	SGD	0.001	88.50%	84.80%
				0.01	84.50%	82.50%
			Adam	0.001	87.00%	81.80%

				0.01	10.00%	10.00%
		model_6	SGD	0.001	87.50%	83.00%
				0.01	86.00%	81.20%
			Adam	0.001	87.50%	82.40%
				0.01	10.00%	10.00%
	arch_2	model_1	SGD	0.001	85.50%	81.00%
				0.01	85.50%	78.80%
			Adam	0.001	90.50%	84.50%
				0.01	90.00%	83.30%
		model_2	SGD	0.001	87.00%	82.90%
				0.01	87.00%	83.30%
			Adam	0.001	93.00%	85.30%
				0.01	88.50%	84.20%
	arch_3	model_1	SGD	0.001	89.50%	83.40%
				0.01	88.00%	85.30%
			Adam	0.001	91.00%	85.40%
				0.01	10.00%	10.00%
		model_2	SGD	0.001	87.00%	84.80%
				0.01	94.00%	89.40%
			Adam	0.001	94.00%	89.10%
				0.01	10.00%	10.00%
		model_3	SGD	0.001	95.00%	90.60%
				0.01	89.00%	79.70%
			Adam	0.001	95.00%	91.10%
				0.01	10.00%	10.00%
group_3	arch_1	model_1	SGD	0.001	65.00%	65.46%
				0.01	65.00%	66.54%
			Adam	0.001	67.69%	66.85%
				0.01	67.69%	66.15%
		model_2	SGD	0.001	66.54%	64.69%
				0.01	64.23%	62.08%
			Adam	0.001	65.38%	66.77%
				0.01	53.85%	60.08%
		model_3	SGD	0.001	70.77%	66.77%
				0.01	70.38%	68.77%
			Adam	0.001	69.62%	67.62%
				0.01	65.00%	67.54%
		model_4	SGD	0.001	67.69%	65.00%
				0.01	65.00%	62.15%
			Adam	0.001	66.54%	64.85%
				0.01	60.00%	58.46%
		model_5	SGD	0.001	66.54%	64.69%
				0.01	64.62%	63.08%
			Adam	0.001	70.38%	65.69%
				0.01	58.85%	57.31%

		model_6	SGD	0.001	67.69%	66.46%
				0.01	64.62%	64.15%
			Adam	0.001	66.92%	67.54%
				0.01	7.69%	7.69%
	arch_2	model_1	SGD	0.001	70.00%	66.08%
				0.01	75.38%	72.46%
			Adam	0.001	81.92%	76.31%
				0.01	73.08%	71.54%
		model_2	SGD	0.001	72.69%	67.85%
				0.01	72.31%	64.92%
			Adam	0.001	74.62%	71.31%
				0.01	73.08%	74.23%
	arch_3	model_1	SGD	0.001	67.31%	67.62%
				0.01	68.46%	67.31%
			Adam	0.001	77.69%	73.38%
				0.01	55.77%	59.38%
		model_2	SGD	0.001	67.31%	68.00%
				0.01	75.38%	71.15%
			Adam	0.001	70.38%	69.77%
				0.01	7.69%	7.69%
		model_3	SGD	0.001	82.31%	78.77%
				0.01	76.92%	72.54%
			Adam	0.001	85.00%	80.62%
				0.01	7.69%	7.69%
group_4	arch_1	model_1	SGD	0.001	76.83%	70.89%
				0.01	78.38%	71.94%
			Adam	0.001	77.61%	74.02%
				0.01	71.43%	68.29%
		model_2	SGD	0.001	78.76%	72.28%
				0.01	78.38%	73.94%
			Adam	0.001	74.13%	71.68%
				0.01	72.20%	70.20%
		model_3	SGD	0.001	77.22%	73.07%
				0.01	79.15%	73.68%
			Adam	0.001	80.69%	75.15%
				0.01	81.08%	75.41%
		model_4	SGD	0.001	79.54%	73.33%
				0.01	76.83%	69.77%
			Adam	0.001	74.52%	69.42%
				0.01	8.49%	2.43%
		model_5	SGD	0.001	76.83%	72.98%
				0.01	76.83%	69.07%
			Adam	0.001	78.38%	70.46%
				0.01	72.20%	66.46%
		model_6	SGD	0.001	77.61%	72.81%

				0.01	75.29%	70.20%
			Adam	0.001	77.99%	74.37%
				0.01	8.49%	2.43%
	arch_2	model_1	SGD	0.001	77.61%	71.50%
				0.01	83.78%	81.84%
			Adam	0.001	84.17%	81.23%
				0.01	81.47%	74.63%
		model_2	SGD	0.001	77.22%	73.41%
				0.01	77.99%	78.63%
			Adam	0.001	81.85%	75.50%
				0.01	82.24%	79.41%
	arch_3	model_1	SGD	0.001	77.99%	78.19%
				0.01	78.76%	75.07%
			Adam	0.001	81.47%	79.50%
				0.01	8.49%	2.43%
		model_2	SGD	0.001	81.47%	81.23%
				0.01	83.40%	80.89%
			Adam	0.001	87.26%	84.36%
				0.01	8.49%	2.43%
		model_3	SGD	0.001	84.94%	83.67%
				0.01	84.94%	82.62%
			Adam	0.001	85.71%	82.97%
				0.01	8.49%	2.43%
group_5	arch_1	model_1	SGD	0.001	86.09%	83.86%
				0.01	82.61%	82.96%
			Adam	0.001	90.43%	85.87%
				0.01	81.74%	82.96%
		model_2	SGD	0.001	87.83%	83.86%
				0.01	83.48%	80.72%
			Adam	0.001	89.57%	85.43%
				0.01	85.22%	80.27%
		model_3	SGD	0.001	90.43%	84.98%
				0.01	87.83%	83.86%
			Adam	0.001	86.09%	85.65%
				0.01	86.09%	84.30%
		model_4	SGD	0.001	90.43%	84.30%
				0.01	86.09%	81.61%
			Adam	0.001	86.96%	83.63%
				0.01	17.39%	22.42%
		model_5	SGD	0.001	84.35%	82.06%
				0.01	88.70%	85.43%
			Adam	0.001	88.70%	84.30%
				0.01	77.39%	77.58%
		model_6	SGD	0.001	88.70%	84.08%

				0.01	88.70%	83.63%
			Adam	0.001	87.83%	83.41%
				0.01	73.04%	71.75%
	arch_2	model_1	SGD	0.001	86.96%	87.67%
				0.01	92.17%	90.13%
			Adam	0.001	89.57%	88.79%
				0.01	17.39%	22.42%
		model_2	SGD	0.001	88.70%	83.41%
				0.01	89.57%	89.01%
			Adam	0.001	93.04%	89.01%
				0.01	92.17%	88.57%
	arch_3	model_1	SGD	0.001	83.48%	82.29%
				0.01	84.35%	88.34%
			Adam	0.001	91.30%	89.01%
				0.01	79.13%	73.99%
		model_2	SGD	0.001	79.13%	83.41%
				0.01	88.70%	89.24%
			Adam	0.001	90.43%	89.69%
				0.01	17.39%	22.42%
		model_3	SGD	0.001	88.70%	85.43%
				0.01	93.91%	88.79%
			Adam	0.001	93.04%	89.91%
				0.01	17.39%	22.42%
group_6	arch_1	model_1	SGD	0.001	89.94%	90.97%
				0.01	92.53%	89.77%
			Adam	0.001	91.56%	91.08%
				0.01	89.61%	89.34%
		model_2	SGD	0.001	91.88%	89.23%
				0.01	90.58%	90.10%
			Adam	0.001	87.66%	84.87%
				0.01	88.31%	86.94%
		model_3	SGD	0.001	90.91%	89.45%
				0.01	90.91%	89.88%
			Adam	0.001	88.96%	90.32%
				0.01	90.58%	88.90%
		model_4	SGD	0.001	89.94%	89.23%
				0.01	91.88%	90.86%
			Adam	0.001	89.94%	90.86%
				0.01	85.71%	84.22%
		model_5	SGD	0.001	91.88%	89.55%
				0.01	90.26%	89.01%
			Adam	0.001	89.61%	89.55%
				0.01	89.61%	86.72%
		model_6	SGD	0.001	90.91%	90.10%
				0.01	91.56%	89.23%

			Adam	0.001	88.96%	90.21%
				0.01	87.01%	83.57%
	arch_2	model_1	SGD	0.001	90.91%	92.06%
				0.01	92.21%	94.78%
			Adam	0.001	91.88%	93.58%
				0.01	88.64%	90.21%
		model_2	SGD	0.001	89.61%	90.75%
				0.01	91.88%	91.29%
			Adam	0.001	90.91%	91.73%
				0.01	90.58%	91.51%
	arch_3	model_1	SGD	0.001	90.58%	86.07%
				0.01	93.18%	93.47%
			Adam	0.001	95.13%	91.95%
				0.01	88.31%	83.13%
		model_2	SGD	0.001	91.56%	89.99%
				0.01	94.16%	93.47%
			Adam	0.001	94.16%	92.38%
				0.01	10.06%	4.35%
		model_3	SGD	0.001	94.16%	94.12%
				0.01	92.86%	94.12%
			Adam	0.001	94.48%	93.47%
				0.01	10.06%	4.35%
group_7	arch_1	model_1	SGD	0.001	89.39%	85.88%
				0.01	89.39%	84.74%
			Adam	0.001	90.91%	88.15%
				0.01	89.39%	83.44%
		model_2	SGD	0.001	89.39%	86.69%
				0.01	92.42%	84.90%
			Adam	0.001	92.42%	86.36%
				0.01	90.91%	87.50%
		model_3	SGD	0.001	90.91%	87.82%
				0.01	93.18%	86.04%
			Adam	0.001	93.18%	85.55%
				0.01	91.67%	87.18%
		model_4	SGD	0.001	93.18%	87.01%
				0.01	91.67%	84.42%
			Adam	0.001	90.15%	85.39%
				0.01	82.58%	77.92%
		model_5	SGD	0.001	91.67%	87.34%
				0.01	89.39%	84.09%
			Adam	0.001	89.39%	86.04%
				0.01	78.03%	75.32%
		model_6	SGD	0.001	92.42%	86.69%
				0.01	90.91%	84.90%
			Adam	0.001	90.91%	84.74%

				0.01	80.30%	71.75%
	arch_2	model_1	SGD	0.001	89.39%	83.77%
				0.01	96.21%	89.12%
			Adam	0.001	92.42%	89.77%
				0.01	87.88%	87.66%
		model_2	SGD	0.001	90.91%	85.88%
				0.01	86.36%	83.60%
			Adam	0.001	88.64%	87.99%
				0.01	90.91%	91.72%
	arch_3	model_1	SGD	0.001	95.45%	88.15%
				0.01	91.67%	84.09%
			Adam	0.001	97.73%	88.96%
				0.01	88.64%	84.25%
		model_2	SGD	0.001	78.03%	64.61%
				0.01	89.39%	82.47%
			Adam	0.001	88.64%	80.84%
				0.01	81.06%	69.32%
		model_3	SGD	0.001	93.94%	84.74%
				0.01	90.15%	89.45%
			Adam	0.001	93.94%	92.69%
				0.01	15.15%	16.23%

5. RESULTA AND DISCUSSIONS

The above experiments indicates that the best model for each group are as follows:

Table 4: Accuracy on test dataset

S. No.	Group	Architecture	Model	Optimizer	Learning Rate	Training Accuracy	Test Accuracy
1	Group-1	Arch_3	Model_3	SGD	0.01	93.56%	88.58%
2	Group-2	Arch_3	Model_3	SGD	0.001	88.61%	91.16%
3	Group-3	Arch_3	Model_3	Adam	0.001	92.57%	91.27%
4	Group-4	Arch_3	Model_2	Adam	0.001	90.1%	88.69%
5	Group-5	Arch_3	Model_3	SGD	0.01	93.56%	88.58%
6	Group-6	Arch_3	Model_1	Adam	0.001	90.1%	91.92%
7	Group-7	Arch_3	Model_1	Adam	0.001	97.73%	88.96%

The effect of hyperparameters i.e. learning rate and optimizer for the best model are shown in figure below which indicates that for Adam optimizer a learning rate of 0.01 is not good, however for SGD optimizer both the learning rates yield comparative accuracies.

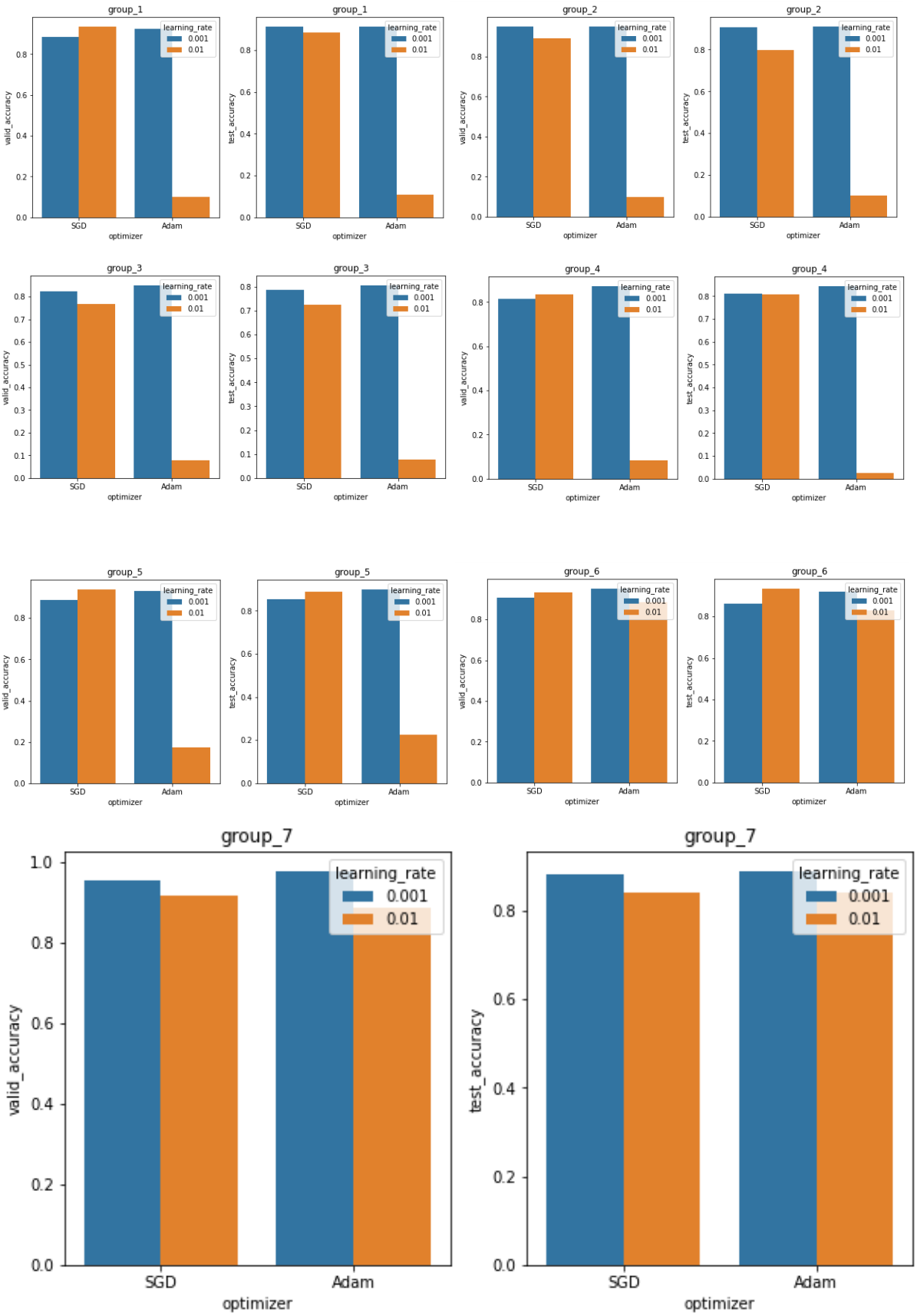
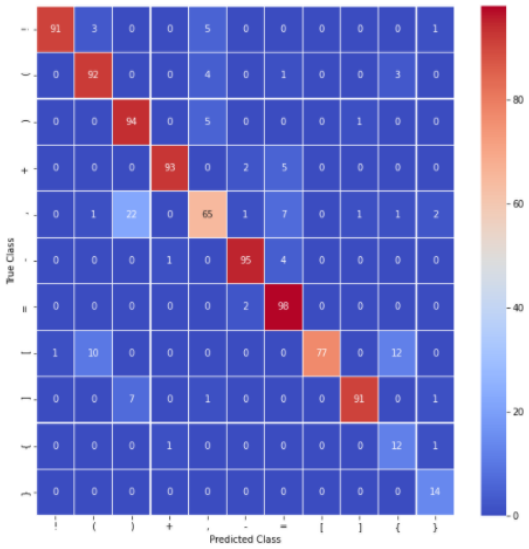


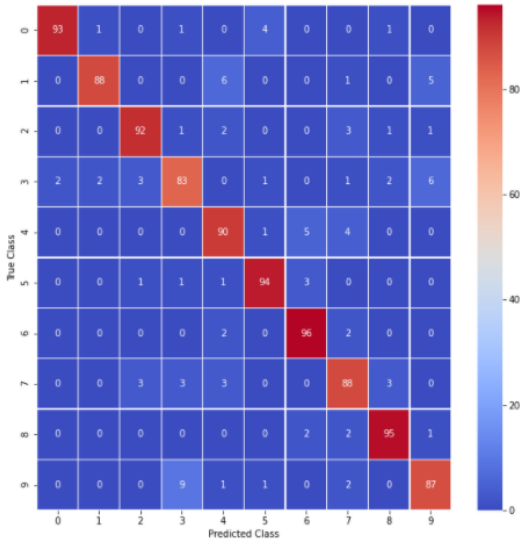
Figure 3: Effect of Hyperparameters on training and test accuracy

The confusion matrix is given in figure below which indicates the number of correctly classified and misclassified images.

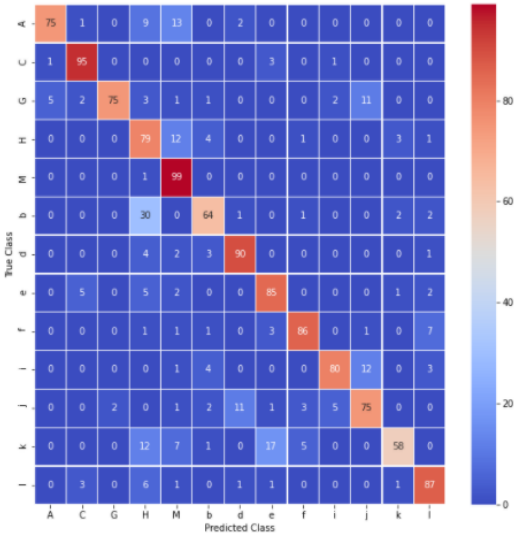
Group-1



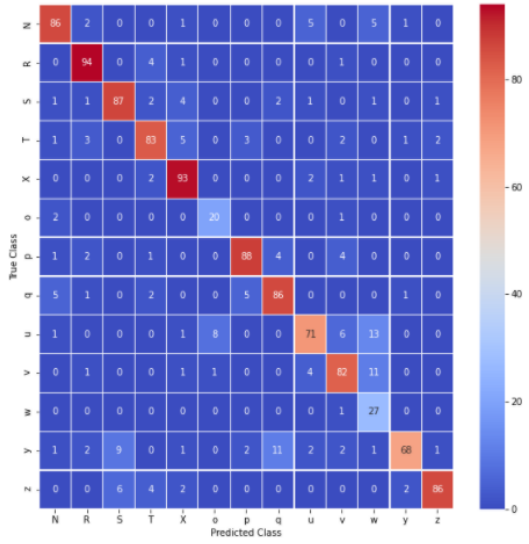
Group-2



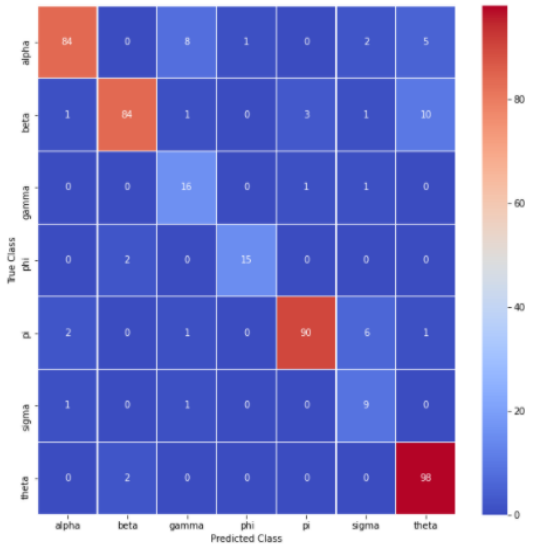
Group-3



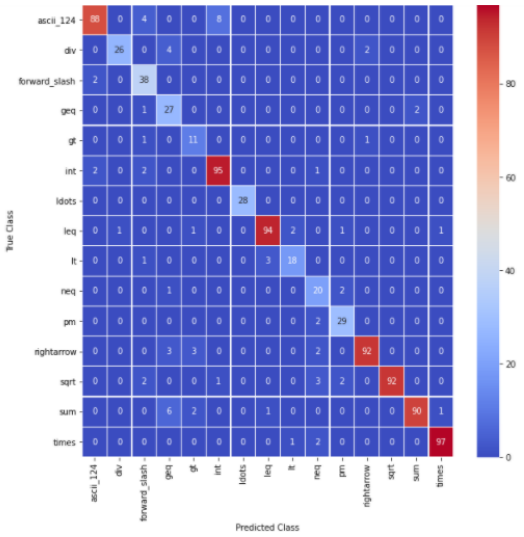
Group-4



Group-5



Group-6



Group-7

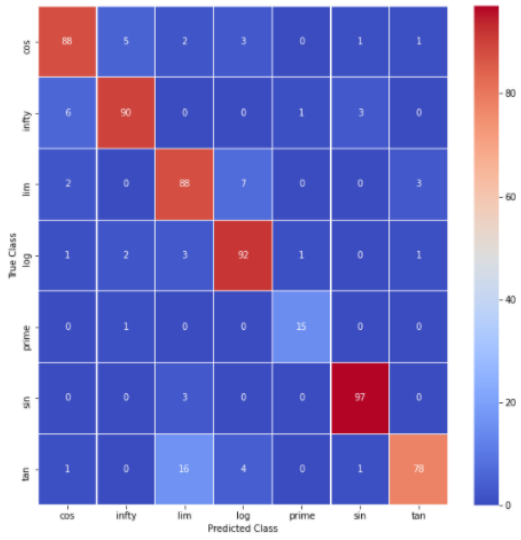
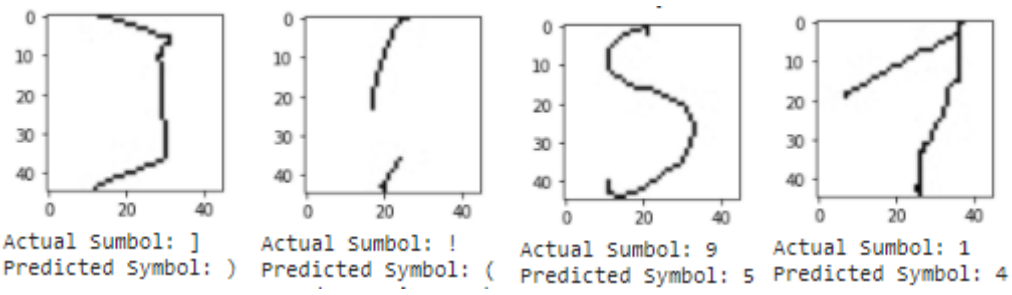


Figure 4: Confusion matrix of each group

Few examples of misclassified images are given below:



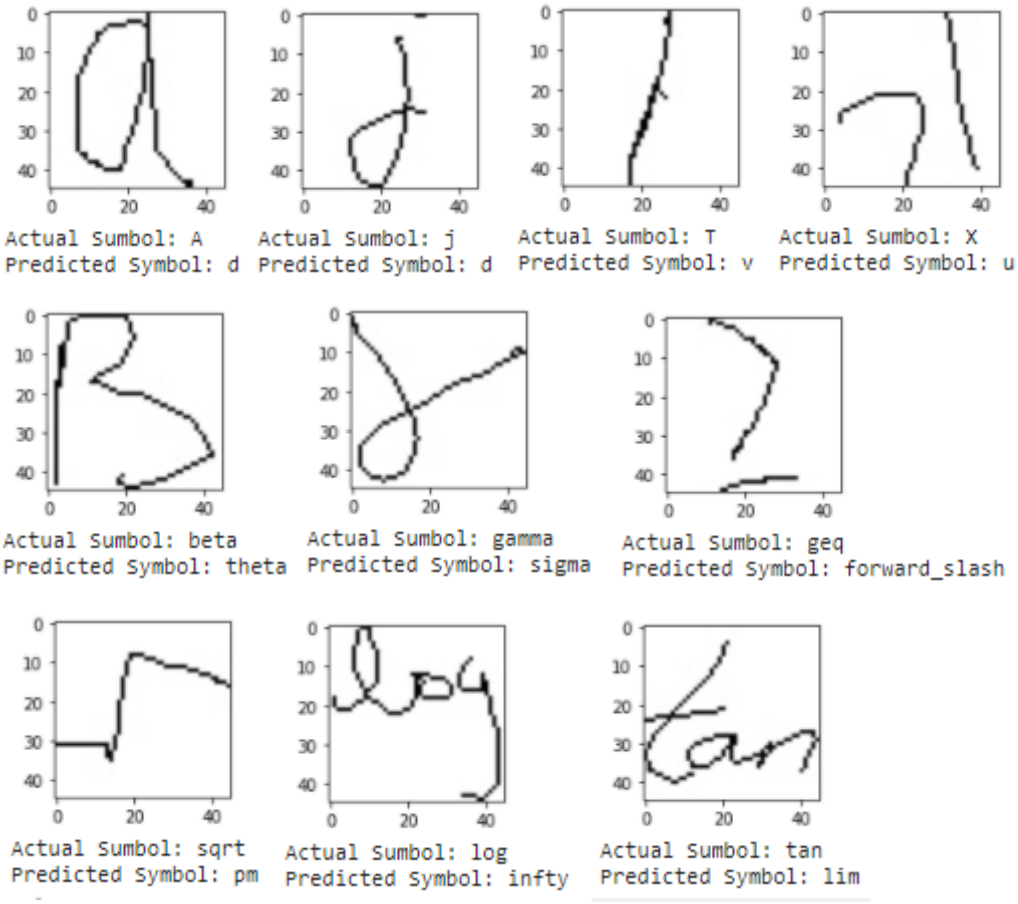


Figure 5: Confusion matrix of each group

The architecture of CNN can be improved by understanding how each layer of the model interprets the images. The figures below show typical image of symbol ‘3’ that how the images are modified after each convolution and maxpooling layer. The figures show that initial layers of CNN predicts the low level features and advanced layers indicate prediction of higher level features.

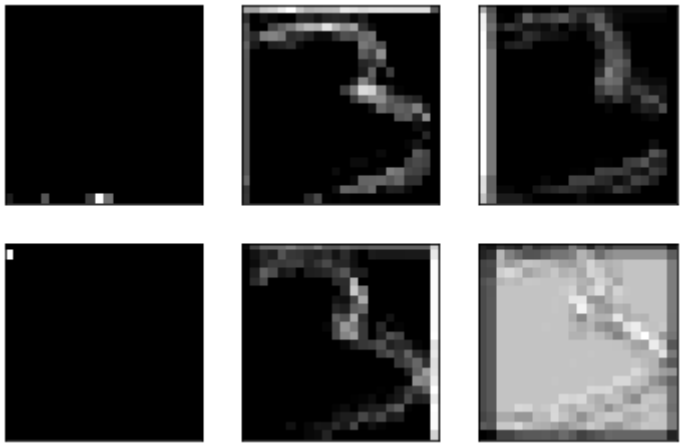


Figure 6: Visualization of image after first layer of convolution and maxpooling

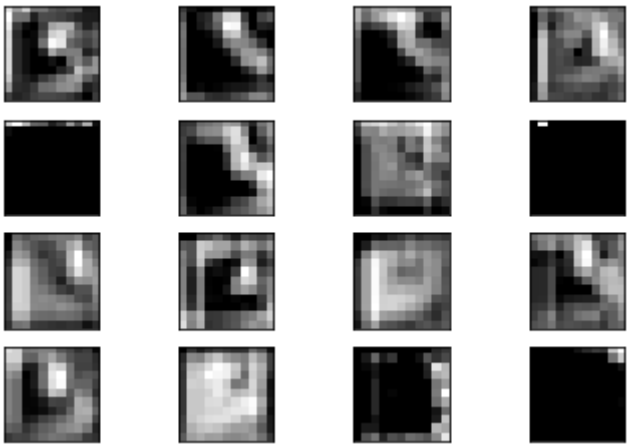


Figure 7: Visualization of image after second layer of convolution and maxpooling

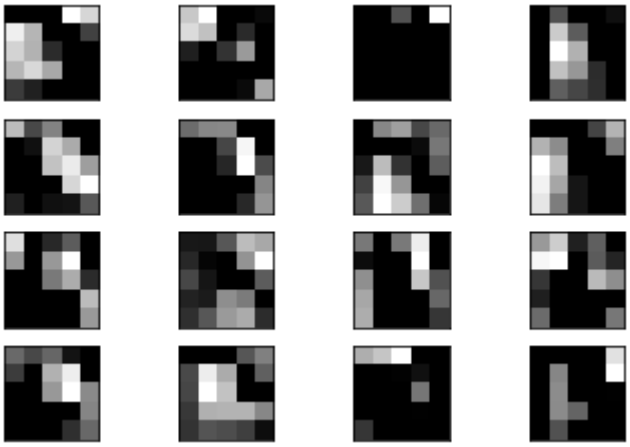


Figure 8: Visualization of image after third layer of convolution and maxpooling

6. CONCLUSION

This project aims at recognition of the handwritten mathematical symbols. The dataset has been acquired from Kaggle [5] which contains over 10,000 images of 82 symbols. The data wrangling and cleaning removed the duplicate images and the classes having a smaller number of images which cannot be used for training. This results in 72 classes of symbols which are further grouped into 7 images for reducing the training time of each group. The image classification has been trained using the approach of Convolutional Neural Network (CNN) which is the most popular algorithm for image processing. The architectures of CNN used have been taken from popular image identification architecture Lenet and Modified Lenet as reported in [4].

The dataset is suitably split into test, validation and training dataset. Each group has indicated the training accuracies in the range of 88% to 93% and test accuracies of 88% to 91% which is considered as reasonable accuracies.

The models have also been tuned with learning rate of 0.01 & 0.001 and with optimizers as SGD and Adam. The experiment indicates that for Adam optimizer, learning rate of 0.01 fails to train on this dataset while other combinations have yield the comparative results. Best models for each group has been considered for predicting on the test dataset.

7. SCOPE OF FUTURE WORK

There are several directions for future work, the most important of which is to predict all 71 classes at once using a deep architecture which will take longer time to train. The second scope would be to club the image segmentation techniques with image recognition of each character which will be used for real life problems and converting handwritten mathematical equations to digital form. The third direction could be use of transfer learning algorithms of already specified CNN architectures to reduce the training time.

8. REFERENCES

1. Nazemi Azadeh, Tavakolian Niloofar, Ftizpatrick Donal, Fernando Chandrika & Suen Ching Y. (2019). Offline Handwritten Mathematical Symbol Recognition Utilising Deep Learning. (https://arxiv.org/pdf/1910.07395.pdf)
2. Matsakis Nicholas E. (1999). Recognition of Handwritten Mathematical Expressions. Submitted to the Department of Electrical Engineering and Computer Science on May 21, 1999, in partial fulfillment of the requirements for the degrees of Bachelor of Science in Electrical Engineering and Computer Science and Master of Engineering in Electrical Engineering and Computer Science.
3. Jimenez Nicolas D & Nguyen Lan. Recognition of Handwritten Mathematical Symbols with PHOG Features. (http://cs229.stanford.edu/proj2013/JimenezNguyen_MathFormulas_final_paper.pdf)

4. Lu Catherine and Mohan Karanveer, Recognition of Online Handwritten Mathematical Expressions Using Convolutional Neural Networks (http://cs231n.stanford.edu/reports/2015/pdfs/mohan_lu_cs231n-project-final.pdf)
5. Handwritten math symbols dataset (Kaggle) (https://www.kaggle.com/xainano/handwrittenmathsymbols?select=data.rar)
6. Jupyter Notebook – Data Wrangling for Capstone Three. https://github.com/pgupta88/Springboard/tree/master/Capstone_Three/Data_Wrangling
7. Jupyter Notebook – Modelling for Capstone Three. https://github.com/pgupta88/Springboard/tree/master/Capstone_Three/Modeling