Handwritten Mathematical Symbols Classification

Ashwitha Anni
Department of ECE
University of Florida
Gainesville, Florida
Ashwithaanni@ufl.edu

Kartik Pandit
Department of ECE
University of Florida
Gainesville, Florida
kartikpandit@ufl.edu

Srikantnag Angondalli Nagaraja
Department of ECE
University of Florida
Gainesville, Florida
srikantnangondal@ufl.edu

Sruthi Chittipalli
Department of ECE
University of Florida
Gainesville, Florida
chittipalli.s@ufl.edu

***Abstract*—An overview of the project on the classification of handwritten mathematical symbols is provided in this publication. We developed a classifier with 10 labels using CNN and the trained model. In this project, we created a symbol classifier that can divide symbols into ten categories. A multi-label classifier is designed using the Inception v3 pre-trained model. The accuracy is increased as compared to earlier experiments utilizing the pre-trained model. Several techniques were used to increase classification accuracy. Over the dataset that was provided, the achieved accuracy rate is 97.82%, and the outcomes were confirmed using appropriate graphs for accuracy and loss. The use of careful experimental design and implementation to choose the model parameters has worked. We use a database of 9032 images of 10 symbols. This report highlights the model used, implementation details, our experiments, and findings.**

***Index Terms*—RESNET, Image Classifier, Convolutional Neural Networks, Dataset.**

I. Introduction

The objective of the project is to build a machine-learning model that can correctly recognize a collection of mathematical symbols. It was accomplished by the application of the following fundamental concepts.

- **CNN**: Convolutional Neural Networks increase the scalability of applications for object recognition and image classification, it is regularized version of multilayer Perceptron. Each neuron in one layer is connected to all neurons in the next layer. Full connectivity of these neurons leads to overfitting of data. The concept of transfer learning allows reusing or transferring information from previously learned tasks for the learning of new tasks has the potential to significantly improve the sample efficiency of a reinforcement learning agent.
- **Inception V3**: Convolutional neural network Inception v3 was developed as a plugin for Googlenet and is used to aid in object detection and picture analysis. The Google Inception Convolutional Neural Network, which was first introduced during the ImageNet project. Inceptionv3 was created with the goal of enabling deeper networks without allowing the number of parameters to become unmanageably large; it contains "under 25 million parameters," as opposed to 60 million for AlexNet.
- **Image augmentation** It is a technique of altering the existing data to create some more data for the model training process. In other words, it is the process of artificially expanding the available dataset for training deep learning models.
- **Fine tuning** Fine tuning is an approach to transfer learning. If we have 90% of training, then we train the same model with the remaining 10%. Usually, the learning rate is changed to a smaller one, so it does not have a significant impact on the already adjusted weights. A base model is used for a similar task and then freezing some of the layers to keep old knowledge when performing the new training session with the new data.

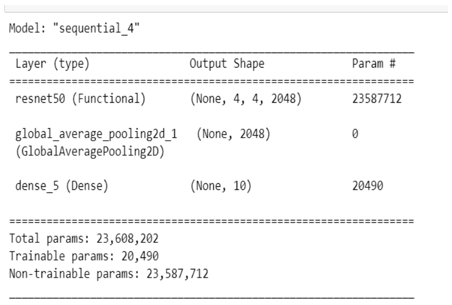
II. IMPLEMENTATION

A. Image Preprocessing

The data set contains 10 different symbols with various index numbers. Given 300*300 pixels image was scaled to 300*100 pixels. The Training and Test were split into a ratio of 3:1. In order to input these images to an Inception V3 model, images are reshaped to 100*100 pixels with 3 channels. After this data was normalized.

B. Model Building

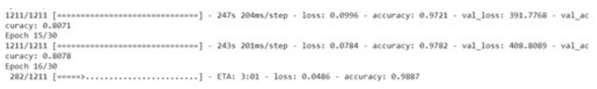
After building the initial model with SVM the accuracy rate was 58% on the test set. CNN was implemented to improve the accuracy of the test set with the following architecture.



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 4, 4, 2048)        23587712

 global_average_pooling2d_1   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_5 (Dense)             (None, 10)                20490

=================================================================
Total params: 23,608,202
Trainable params: 20,490
Non-trainable params: 23,587,712
_________________________________________________________________
```

Fig. 1. Model 1.

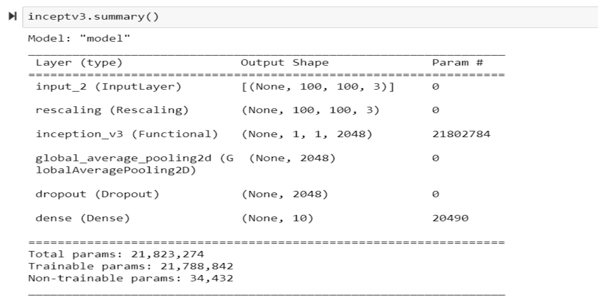
The maximum test accuracy obtained after implementing this model was 80% although training was overfitting with accuracy reaching 99%. Hence, Inception V3 Convolutional Neural Network (CNN) was employed for the



```
1211/1211 [==============================] - 247s 204ms/step - loss: 0.0996 - accuracy: 0.9721 - val_loss: 391.7768 - val_ac
curacy: 0.8071
Epoch 15/30
1211/1211 [==============================] - 243s 201ms/step - loss: 0.0784 - accuracy: 0.9782 - val_loss: 408.8089 - val_ac
curacy: 0.8078
Epoch 16/30
 282/1211 [=====>........................] - ETA: 3:01 - loss: 0.0486 - accuracy: 0.9887
```

Fig. 2. Epoch Outcome.

classifier's implementation. CNN was preferred over alternative techniques because of its increased classification accuracy.



```
inceptv3.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 100, 100, 3)]     0

 rescaling (Rescaling)       (None, 100, 100, 3)       0

 inception_v3 (Functional)   (None, 1, 1, 2048)        21802784

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 10)                20490

=================================================================
Total params: 21,823,274
Trainable params: 21,788,842
Non-trainable params: 34,432
_________________________________________________________________
```

Fig. 3. Inception V3 model architecture.

The accuracy after using the Inception V3 model was around 90%. To improve accuracy further, image augmentation and fine-tuning were used, which resulted in an accuracy of 97.82%.

III. EXPERIMENTS

A. Table of Data

Each student in this course has collected 100 samples from 10 different classes. Data transformation was carried out by scaling and normalizing the data.

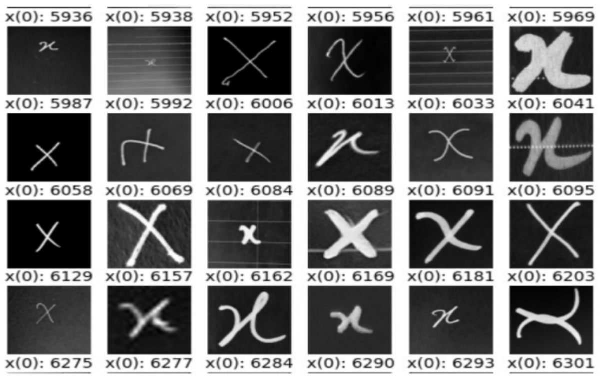


Fig. 4. Sample Images from the dataset.

Initially, LDA, KNN, and SVM with feature extraction models like PCA, Image Inversion, Edge Detection along with gird search are used. Even with varying parameters, the test accuracy did not pass more than 60%. Even though approximately 60 percent accuracy was achieved for the test set when tested manually, the model produced numerous false positives and misidentified comparable forms. Since a convolutional neural network (CNN) design does a better job of handling image data than other models, it was tried for the second iteration. The accuracy has significantly improved as compared to another model. CNN model is built with an image resolution of 50*50 which gave low accuracy. The resolution is the changed to 300*300 with batch sizes of (128,64,32) and epochs from (10-30) for which accuracy is varied between 78-82%. In the final Model Inception, V3 is used to reshape the image to (D, D, 3) format which broke the threshold level of test accuracy and reached 90% in 3 Epochs. Fine-tuning method is then

IV. OBSERVATIONS

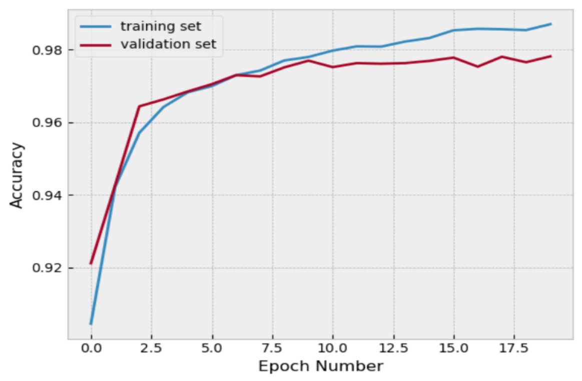
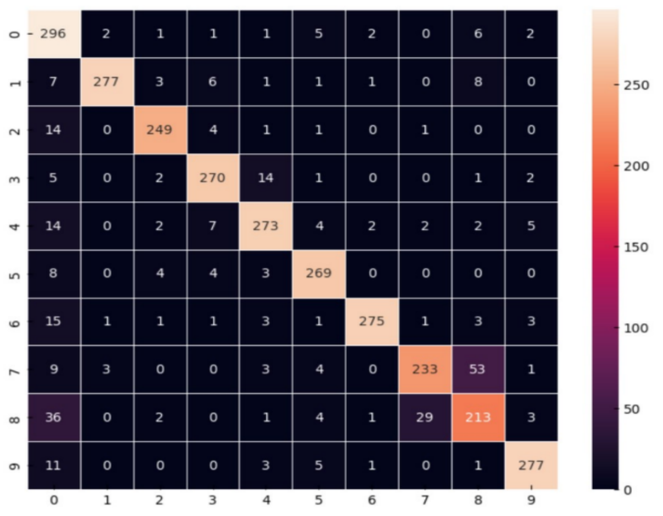


Fig. 5. Accuracy v/s Epoch plot for test and training set.

The above plot shows improved accuracy with each Epoch, initially training set accuracy is lower than that of test set accuracy, this is because of including dropout in the CNN layer. The training set accuracy reached up to 99%, whereas the maximum test accuracy saturated at 97%.



	0	1	2	3	4	5	6	7	8	9
0	296	2	1	1	1	5	2	0	6	2
1	7	277	3	6	1	1	1	0	8	0
2	14	0	249	4	1	1	0	1	0	0
3	5	0	2	270	14	1	0	0	1	2
4	14	0	2	7	273	4	2	2	2	5
5	8	0	4	4	3	269	0	0	0	0
6	15	1	1	1	3	1	275	1	3	3
7	9	3	0	0	3	4	0	233	53	1
8	36	0	2	0	1	4	1	29	213	3
9	11	0	0	0	3	5	1	0	1	277

Fig. 6. Confusion Matrix.

The Confusion matrix obtained clearly shows misclassification mostly in Pi (Integer encoding: 8) and Summation

(Integer encoding: 9) symbols. All the other symbols were classified with least or no error.

V. CONCLUSION

In this project, with given real world data a model is built using inception v3 CNN to classify image samples. An accuracy of 97.82% was observed for the constructed neural network for the validation set." handwritten_math_recognition_cnn.h5" is created to test any other test data.

ACKNOWLEDGMENT

We appreciate the University of Florida HiperGator for providing the tools and coordinated environments needed to carry out and put our experiments into practice. We sincerely thank Professor Catia D. Silva for providing us with this fantastic opportunity to gain practical knowledge of the fundamentals of machine learning. Additionally, we would like to thank Marquez Jones and all TAs for their coaching and guidance.

REFERENCES

[1] Christopher M. Bishop: Pattern Recognition and Machine Learning, Chapter 4.3.4
[2] K. Taunk, S. De, S. Verma and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," 2019
[3] F. Zhuang et al., "A Comprehensive Survey on Transfer Learning," in Proceedings of the IEEE, vol. 109, no. 1, pp. 43-76, Jan. 2021, doi: 10.1109/JPROC.2020.3004555
[4] L. Xiao, Q. Yan and S. Deng, "Scene classification with improved AlexNet model," 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), 2017, pp. 1-6, doi: 10.1109/ISKE.2017.8258820.
[5] Y. Gurav, P. Bhagat, R. Jadhav and S. Sinha, "Devanagari Handwritten Character Recognition using Convolutional Neural Networks," 2020 International Conference on Electrical, Communication, and Computer.
[6] itish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929– 1958.
[7] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
[8] https://www.tensorflow.org/tutorials/images/transfer_learning