Handwritten Mathematical Symbols Classification Using ResNet Transfer Learning

Akash Kumar Kondaparthi
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL, USA
akash.kondaparth@ufl.edu

Prama Jati
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL, USA
prama.jati@ufl.edu

Nikhil Reddy Vanja
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL, USA
nikhilredd.vanja@ufl.edu

Thanyachanok Sutthanonkul
Department of Agricultural and Biological Engineering
University of Florida
Gainesville, FL, USA
t.sutthanonkul@ufl.edu

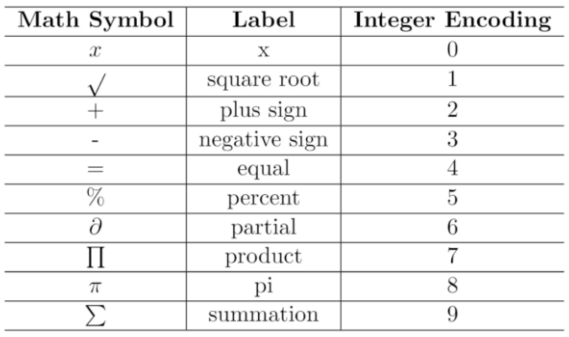
***Abstract*— This report is part of the final project for the course EEL5840 - Fundamentals of Machine Learning. The goal of this project is to make a model that can learn and classify handwritten mathematical symbols. The ResNet was implemented using transfer learning, which was imported from TensorFlow library, ResNet50. This model was chosen as ResNet was known to have delivered an astounding top-five error rate under 3.6% and won the won the ILSVRC 2015 challenge. The weights from the 'imagenet' data were used. The highest accuracy scores obtained from the validation and test sets were 92 percent and 93 percent respectively. The results are validated by plotting learning curves, classification reports, and confusion matrices.**

Keywords—Convolutional Neural Networks (CNN), ResNet, image classification, handwritten mathematical symbols

I. Introduction

The Convolutional Neural Networks (CNN) have the most effective results for the image dataset classifications. CNNs borrow the concept of cross-correlation and convolution from the signal processing domain and extract important features by implementing these on the images. CNN has multiple layers of convolution followed by pooling layers which reduce the feature size. At the lower layers, the low-level features will be identified and then the complicated patterns which are combinations of low-level features will be learned at the higher layers. This report explains the project where a variant of a Convolutional Neural Network, called a Residual Network (ResNet) was used for the image classification task.

The task of handwritten mathematical symbol recognition was to classify the math symbol with their correct label which was done using the Residual Network (or ResNet), which is a variant of a Convolutional Neural Network (CNN). The dataset had 10 math symbols with their respective class labels from 0 to 9 as shown in figure 1. During preprocessing, an additional class label was added for an unknown class with a class label of 10. Our training model can predict an unknown class which does not belong to any of the 10 classes.



Math Symbol	Label	Integer Encoding
x	x	0
$\surd$	square root	1
$+$	plus sign	2
-	negative sign	3
$=$	equal	4
$\%$	percent	5
∂	partial	6
$\prod$	product	7
π	pi	8
$\sum$	summation	9

Figure 1. Classes and their labels

II. Implementation

A. Data Collection

The data was collected by the entire class with the mathematical symbols being written by hand on different backgrounds. The handwritten symbols were then taken pictures of and cropped so that the images are square shaped, and such that the symbol occupied 90% of the image space. These images were then resized to 300x300 pixels with gray color map. These images were renamed such that their name included their target label, and their labels were automatically collected from the image names. All the images and labels collected from all the students were pooled and the obtained data and their labels were saved in numpy arrays. The training data was saved as 'data_train.npy' and their respective labels were stored as 'labels_train.npy'. The resultant data had 9,032 instances with 90,000 features (300x300).

B. Data Preprocessing

The data included many images which are mislabeled i.e., many of the images were assigned a wrong label. Moreover, the data also included instances which didn't belong to any of the classes provided. So, a preprocessing stage was necessary to rid the data of any mislabels and imperfections so that the model would have a better idea of the data and obtain better performance. In this preprocessing stage, any mislabeled data was corrected with the new correct labels and the images which belonged to no provided class were added into a new class named ‘unknown’ with label 10. Due to the restrictions on the computational resources available for the project (16GB of RAM on HiPerGator), data processing at full scale 300x300 pixels was impossible. Hence, OpenCV was used to resize the 300x300 pixel images to 200x200 pixel images. Another reason to reduce the number of dimensions was the original number of dimensions was 90,000 which is too many. This can lead to overfitting, giving bad accuracy for the test data. Hence, the data which previously had 90,000 features, now has 40,000 features. A ResNet works well on an image dataset with RGB channels. So, the image dataset was again expanded into the color space with 200x200x3. The final number of features for the input layer will be 120,000.

C. *Choosing the Model – Model Architecture*

The model used to achieve this classification was a ResNet (Residual Network), a variant of a CNN (Convolutional Neural Network). The innovation of ResNet is that it uses skip connections (also called shortcut connections). Skipping connections is adding the signal feeding into a layer directly to the output of a layer located a bit higher up the stack.

Using the TensorFlow library, ResNet50 was imported to apply transfer learning. This imported ResNet has 175 deep layers with the weights obtained from ‘imagenet’. As the training dataset is available in 200x200x3 images, the input shape for the network is also [200,200,3] followed by the 175 deep layers architecture with the weights already available. This imported network will be addressed as the ‘base model’ from here on. The first 170 layers of the base model (i.e., except the last 5 layers) are frozen, which means that the weights in these layers remain unchanged during training. This base model is connected to an additional 5 fully connected layers of the architecture as shown.

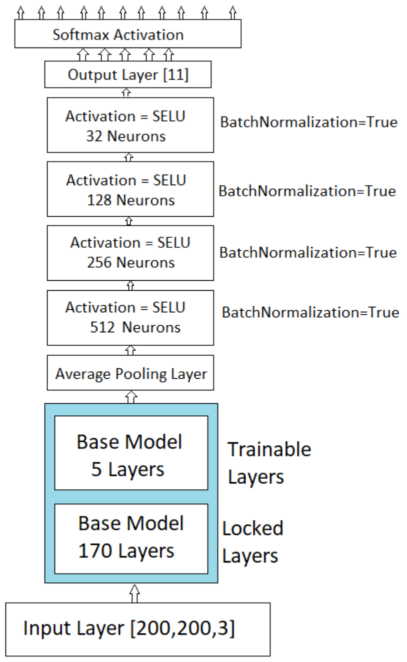


Figure 2. Model Layout

The output from the last trainable layer from the base model is sent into the Average Pooling Layer which would be the base for classification. The next hidden layer has 512 neurons, with batch normalization applied and the kernel initializer used was ‘lecun_normal’. The activation for this layer was SELU and dropout rate 0.7. The next hidden layers parameters are: {Number of neurons = 256; kernel initializer = ‘lecun_normal’; Activation function = SELU; Dropout = 0.5; with Batch Normalization}.

The following hidden layer has the following parameters:
{Number of neurons = 128; kernel initializer = ‘lecun_normal’; Activation function = SELU; Dropout = 0.3; with Batch Normalization}.

The last hidden layer has the following parameters: {Number of neurons = 32; kernel initializer = ‘lecun_normal’; Activation function = SELU; Dropout = 0.2; with Batch Normalization}.

This layer is then followed by the output layer, which has the same number of neurons as there are the number of classes since this is a multi-label classification task. As for the obtained dataset, including the unknown class, there are 11 classes. Hence, there are 11 neurons in the output layer. Since the classes are exclusive, a SoftMax activation function is applied to the output layer to get the probability for each class.

D. *Training the Model*

The training data was split into full training and test sets with shuffling and fixed random state with stratification. The data was split into 85% full training set and 15% test set. The full training set was further split into 80% training set and 20% validation set using the same random state, shuffle and stratification set to true.

Using Adam optimizer with a learning rate of 0.01 and using Sparse categorical cross entropy loss function the model was run on the training and validation data after applying early stopping callback by monitoring the validation loss and patience set to 10. The number of epochs was set to 50 and the batch size was set to 32.

III. EXPERIMENTS

A. *Phase 1: Initial Experiments*

To classify the handwritten mathematical symbols, we implemented different classifiers and tested the given dataset. Initially, we choose Naïve Bayes Classifier, Fisher's Linear Discriminant Analysis Classifier, k-Nearest Neighbour Classifier and Support Vector Machines for training and testing the dataset. Data was scaled using MinMaxScaler and applied dimensionality reduction techniques such as Linear Discriminant Analysis (LDA) and Principal component analysis (PCA). k-Fold cross validation was performed wherever applicable with GridSearchCV to tune each classifier's hyperparameters. In this phase we achieved maximum accuracy of 62% with SVM.

B. *Phase 2: Intermediate Experiments*

As the hand-written symbols are complex, to capture the complexity of each image, we have chosen more complex classifiers. In this phase we selected Neural Network to train our model. First, we implemented Artificial Neural Network (ANN) with different number of layers and different optimizers such as Stochastic Gradient Descent (SGD), and Adam. We were able to achieve 55% accuracy with Adam optimizer and SELU activation function.

Next, we implemented a training model using Convolution Neural Network (CNN). In this implementation, we used 3 convolution layers followed by MaxPooling layers and Batch Normalization. RELU activation functions are used in convolution layers. Then for the classification, we added flatten layer and 3 Fully connected layers with SELU activation function. Finally, the output layer was added with softmax activation function. With this architecture, we achieved an accuracy rate of 75%.

C. *Phase 3: Final Experiment*

As we have a limited number of training samples (using around 6100 for training after splitting data for training, validation, and test), the models were not able to perform well. To capture the complexity with these limited numbers of samples, we decided to use transfer learning method. We have selected ResNet with pre-trained 'imagenet' weights.
As our dataset contains different images, we have allowed a few top layers (5 layers at the top) to be weight tunable, freezing the rest of the layers. We have decoupled the output layer of this pre-trained model and added our own model for the classification task. For classification layers, we have baselined our phase 2 model which was already tuned for good accuracy.
Then, for tuning this model and selecting the best architecture, we tried using a different number of layers and units and modified it after observing the behavior of the model. For example, when the model started over-fitting, we opted for Dropouts. The dropout proportion was made considering the number of units in the layer.
We have experimented with different batch sizes where we observed when the batch sizes are increased accuracy is getting affected though the learning curve is smooth. Therefore, we selected batch size as 32 (mini-batch learning) to achieve faster convergence and good accuracy.

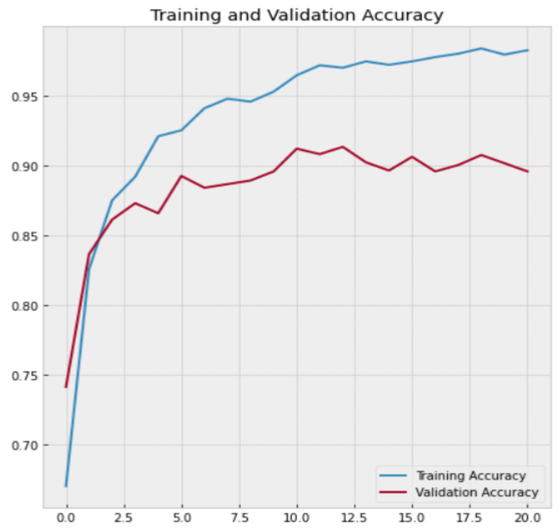


Figure 3. Training and Validation Accuracies vs Number of Epochs

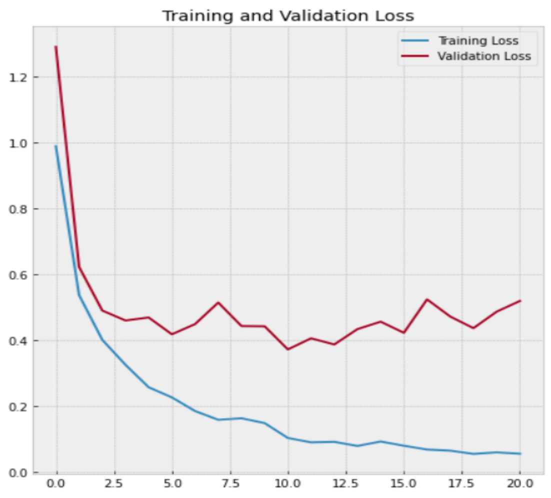


Figure 4. Training and Validation losses vs Number of Epochs

With these different experiments, we have chosen our final training model architecture as mentioned in section II c. Our final model was able to achieve a persistent accuracy of over 91% in individual run. The final learning curves for the implemented model are shown in figure 3, and their loss values are also shown in figure 4 as a function of the number of epochs. The normalized confusion matrix obtained for the test set is shown in figure 5.

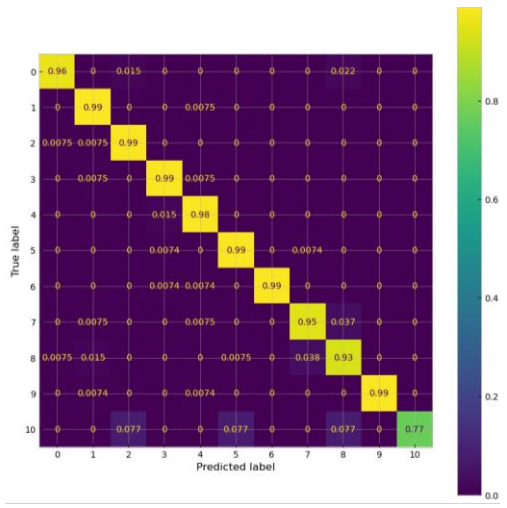


Figure 5. Confusion Matrix for the test data

IV. CONCLUSIONS

ResNet, a variant of convolutional neural network was used in this project to classify the handwritten mathematical symbols into 10 different classes with additional unknow class. From the experiments, it was apparent that the feature reduction using OpenCV reduced the resolution of the image which enabled us to run the model with less computational cost and reduced the chance of overfitting. It was also observed that a very high number of layers led to overfitting and bad accuracy in the test sets. Using batch normalization and dropout methods helped the model to achieve better accuracy. An accuracy of 92% was observed in the validation sets and 93% for the test set. Finally, the ResNet proved to be a reliable model to classify the handwritten mathematical symbols with high accuracy.

REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, : ImageNet Classification with Deep Convolutional Neural Networks

[1] Wayne Lu, Elizabeth Tran: Free-hand Sketch Recognition Classification

[3] Olga Russakovsky et al: ImageNet Large Scale Visual Recognition Challenge

[4] Online Materials: Convolutional Neural Networks : https://cs231n.github.io/neural-networks-3/