Handwritten Mathematical Symbols Classification

Robert Lattus
Department of Electrical and Computer Engineering
Herbert Wertheim College of Engineering, University of Florida
Gainesville, Florida, USA
robertlattus@ufl.edu

Gadhaun Aslam
Department of Engineering Education
Herbert Wertheim College of Engineering, University of Florida
Gainesville, Florida, USA
gadhaunaslam@ufl.edu

Yu Chuang
Department of Mechanical and Aerospace Engineering
Herbert Wertheim College of Engineering, University of Florida
Gainesville, Florida, USA
yu.chuang@ufl.edu

Sebastian Fauré
Department of Computer & Information Science & Engineering
Herbert Wertheim College of Engineering, University of Florida
Gainesville, Florida, USA
sebastianfaure@ufl.edu

***Abstract*— The recognition of handwritten mathematical symbols is one of the popular classification problems. There are different algorithms used for this purpose. The algorithms used in this paper are the support vector machine, naive bayes algorithm, Fisher's LDA and convolutional neural networks (CNNs) with different architectures. Various architectures were tested with thorough iterations through different hyperparameter values. The best CNN model network (modified CNN-15) yielded an accuracy of 92.07% for the training set whereas the test accuracy came out to be 94.81%. This is because of the extensive hidden layers in comparison to other above-mentioned algorithms. More fine-tuning of the hyperparameters would yield an even better accuracy.**

Keywords—machine learning, neural networks, CNNs, symbols

I. Introduction

Machine learning has proven valuable to solve complex real-world problems in the fields of healthcare, finance, fraud detection, etc. It uses large amounts of data, tries to learn and identify the patterns in data and gives us insights and predictions about unknown data. It uses a variety of algorithms which work differently for different kinds of problems.

Handwritten mathematical symbols are one of the classical problems in which the goal is for the machine to learn to correctly classify handwritten symbols correctly. However, the recognition of handwritten mathematical symbols has its own challenges. The handwritten mathematical symbols are hard to detect because of several reasons. It may include the light exposure, the way in which they are written, resolution of the image, image quality, etc. Usually, the images are gray scaled and vectorized in order to enhance the recognition [1]. Each symbol has to be -

Naive Bayes Algorithm is used for classification tasks using the probabilities [2]. It performs well for some real-world applications. The support vector machine (SVM) is a learning algorithm that has become very popular for various biological applications. There are different versions of SVM including the hard-margin and the soft margin SVM. [3] The Support Vector Machine has been proven to be effective in detecting handwritten symbols with little training data for a multi-class problem [4].

Deep Learning is a subset of machine learning where artificial neural networks learn from large amounts of data. Deep Learning is now used for a wide range of applications including object detection, visual recognition, voice recognition, segmentation, and natural language processing. Convolutional Neural Networks are robust artificial neural network architectures which are used in this regard. They belong to the class of deep learning and are used in computer vision tasks. It may consist of convolution layers, pooling layers and fully connected layers that are designed to learn the feature using a backpropagation algorithm.

Many researchers have used several techniques for handwritten mathematical symbols classification. One technique used included a convolutional neural network that includes an input layer, two hidden layers and an output layer that yielded a test accuracy of 87.72% [5]. An algorithm for locating determining points for the newly written symbols using the determining points from the annotated average symbols that belong to the same category [6]. Another technique used in this case was using offline features for classifying handwritten math symbols with recurrent neural networks [7].

II. Implementation

This project detects ten (10) mathematical symbols that are handwritten. The data for this project was collected using the samples gathered from students enrolled in the Fundamentals of Machine Learning course. The symbols or classes include x, square root, plus sign, negative sign, equal, percent, partial, pi and summation sign. Each of these symbols were integer encoded as 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 respectively. Every student submitted a total of 100 images. Each of the images were labeled using a particular encoding. The initial coding system used was ID-trail-label. All the images were combined and led to a data of about 9032 images. Each class had about 900 images each. The methods used in this project include data processing techniques like data augmentation. The algorithms applied for the project data are naive bayes classifier, support vector machine, convolutional neural network, etc.

We used three different architectures for convolutional neural networks with different numbers of layers and hyperparameters. The first architecture of the convolutional neural network (CNN-4) used in the project consisted of four hidden layers. The activation function used in the hidden layers is Rectified Linear Unit (RELU). The activation function used for the output layer is Softmax function. The dropout rate was 0.5.

The second architecture of the convolutional neural network (augmented CNN-12) used in the project consisted of twelve hidden layers with two pooling layers. The activation function used in the hidden layers is Rectified Linear Unit (RELU). The activation function used for the output layer is Softmax function. The dropout rate was 0.25. The L2 regularizer was also used in this architecture.

The third architecture of the convolutional neural network (modified CNN-15) used in the project consisted of fifteen hidden layers with two pooling layers. The activation function used in the hidden layers is Rectified Linear Unit (RELU). The activation function used for the output layer is Softmax function. The dropout rate was 0.25. The L2 regularizer was also used in this architecture. It has batch size of 32 and the number of epochs as 30. This architecture is inspired from the model used for MNIST dataset [8]. The libraries imported for the implementation includes tensorflow, keras, cv2, numpy, sklearn, and matplotlib. The packages included in keras are utils, layers and models.

III. Experiments

Different experiments were conducted in order to train and test the data. Initially, different preprocessing techniques were used including data augmentation. Initially, the labeling of the dataset was taken into account. All the incorrect data labels were corrected to ensure that the dataset is accurate. The symbols that did not represent any of the class were labeled as unknown (u) or misclassified. These were identified through data visualization. The images were converted into gray scale initially and then their color was inverted. Two of the main classes that looked challenging were classes 8 and 9. This was because a lot of the samples in these classes overlap, as the capital Pi looks similar to lowercase Pi. Also, classes 0 and 2 were most difficult, as the plus sign and x had some overlaps to them. The classes were not balanced, but in the data preprocessing part, the classes were balanced to the minimum by dropping a few samples that were questionable. These were used to make sure the images are correctly labeled. This was also done to ensure that the dimensions of the images were correct.

The correctly formatted images were considered later and feature scaling techniques were applied. The data was split into training images (70%) and test images (30%).

Initially, three different algorithms were used to see which one performed better for the dataset. For the Support Vector Machine, the training accuracy came out to be 91.56% and for the test data it was 63.40%. For the Fisher's LDA, the training accuracy came out to be 91.56% and for the test data it was 63.40%. The Naive Bayes algorithm worked the worst for both the training and the test data. The accuracy for both datasets came out to be 27%.

A comparison table which shows the accuracies from different algorithms for the training and test dataset is shown in Table I.

TABLE I. Comparison of accuracies for different algorithms

Algorithm	**Training Accuracy**	**Test Accuracy**
Support Vector Machine	91.56%	63.40%
Fisher's LDA	83%	29%
Naive Bayes	27%	27%

Several data augmentation techniques were applied to make the images in a better format. There were three main transformations. The first technique used was zooming. In this technique, a box was drawn around the symbol and the outer pixels were removed from all four edges. The second technique was flipping in which the images were rotated by 180 degrees. The third technique was the gray scale in which 40% of the samples were used. In this technique, the color was inverted for those specific samples. The support vector machine algorithm was run for all three transformations individually, and the training and the test accuracies are shown in the Table II.

TABLE II. Accuracies for different data augmentation techniques

Data Augmentation Techniques	Support Vector Machine	
	Training Accuracy	Test Accuracy
Zooming		
Flipping	88.5%	56.5%
Gray Scaling (40% samples)	85.65%	53.7%

The hyperparameters that were used and tested for different values include batch size, learning rate, epochs, and optimizers. The batch size varied from 1 to 512, learning rate was changed from 0.1 to 0.00001, and epochs were tested for 10 to 30 epochs.

For the first convolutional neural network architecture (CNN-4), the best values came out to be for the hyperparameter values mentioned here. The number of hidden layers were 4. The best batch size turned out to be a small mini batch of 16, with the learning rate of 0.0001, and the best number of epochs is 13. The optimizer that was used was Adam. The dropout rate was also used as 0.5. The training and the test accuracies for different image sizes have been listed in Table III.

TABLE III. Accuracies for different image sizes using CNN

Algorithm	**Image Size**	**Training Accuracy**	**Test Accuracy**
Convolutional Neural Network (CNN)	300*300	78%	58%
Convolutional Neural Network (CNN)	100*100	81%	72%

Convolutional Neural Network (CNN)	50*50	88.6%	79.5%
Convolutional Neural Network (CNN)	25*25	74%	72%

For the second convolutional neural network architecture (augmented CNN-12), the best values came out to be for the hyperparameter values mentioned here. The number of hidden layers were 12. The best batch size turned out to be a small mini batch of 32, with the learning rate of 0.001, and the best number of epochs is 30. The optimizer that was used was Adam. The dropout rate was also used as 0.25. The training and the test accuracies for different image sizes have been listed in Table IV.

TABLE IV. ACCURACIES FOR DIFFERENT IMAGE SIZES USING CNN

Algorithm	**Image Size**	**Training Accuracy**	**Test Accuracy**
Convolutional Neural Network (CNN)	300*300	92.72%	75.93%
Convolutional Neural Network (CNN)	100*100	94.19%	82.62%
Convolutional Neural Network (CNN)	50*50	95.37%	86.00%
Convolutional Neural Network (CNN)	25*25	87.32%	79.87%

As the image size was changed from 50*50 to 25*25, the training and the test accuracy dropped for both the convolutional neural network architectures (CNN-4 and augmented CNN-12).

Another experiment that was included was using a third CNN architecture (modified CNN-15). It has fifteen (15) hidden layers in addition to the input and output layer. It has six convolution layers. Batch normalization was used in this regard. A variable learning rate was increased in which it decreased significantly as the accuracy reached closer to 90%. The batch size used was 32. The dropout rate was 0.25. The best number of epochs is 30. The optimizer that was used was Adam. An early stopping criterion was incorporated by using a patience of 2. In this scenario, we had the following results as listed in Table V:

TABLE V. ACCURACIES FOR DIFFERENT IMAGE SIZES USING CNN

Algorithm	**Image Size**	**Training Accuracy**	**Test Accuracy**
Convolutional Neural Network (CNN)	300*300	95.04%	83.91%
Convolutional Neural Network (CNN)	100*100	96.46%	89.60%
Convolutional Neural Network (CNN)	50*50	92.07%	94.81%

This architecture did not run for 25*25 because of the kernels inside the hidden neurons would not align with the input data.

IV. CONCLUSION

Out of all the algorithms that have been used in the experimentations, the convolutional neural network (modified CNN-15) worked the best. These accuracies were achieved after hyperparameter tuning i.e. after trying multiple values of learning rate, image sizes, number of epochs and batch size. The highest training accuracy and test accuracy came out to be 92.07% and 94.81% respectively. Although the other algorithms were tested, the CNN-15 performed the best because of the increased number of hidden layers, flexibility in terms of dimensionality and ease of implementation. This model can be further fine tuned with the hyperparameter values to better increase the accuracy for training and test data.

REFERENCES

[1] Taxt, Torfinn, Jórunn B. Ólafsdóttir, and Morten Dæhlen. "Recognition of handwritten symbols." *Pattern Recognition* 23.11 (1990): 1155-1166.

[2] Taheri, Sona, and Musa Mammadov. "Learning the naive Bayes classifier with optimization models." *International Journal of Applied Mathematics and Computer Science* 23.4 (2013): 787-795.

[3] Noble, William S. "What is a support vector machine?." *Nature biotechnology* 24.12 (2006): 1565-1567.

[4] Malon, Christopher, Seiichi Uchida, and Masakazu Suzuki. "Mathematical symbol recognition with support vector machines." *Pattern Recognition Letters* 29.9 (2008): 1326-1332.

[5] Ramadhan, Irwansyah, Bedy Purnama, and Said Al Faraby. "Convolutional neural networks applied to handwritten mathematical symbols classification." *2016 4th international conference on information and communication technology (ICoICT)*. IEEE, 2016.

[6] Hu, Rui, and Stephen M. Watt. "Determining points on handwritten mathematical symbols." *International Conference on Intelligent Computer Mathematics*. Springer, Berlin, Heidelberg, 2013.

[7] Alvaro, Francisco, Joan-Andreu Sánchez, and José-Miguel Benedí. "Offline features for classifying handwritten math symbols with recurrent neural networks." *2014 22nd International Conference on Pattern Recognition*. IEEE, 2014.

[8] Gupta, J. (2020, May 4). *Going beyond 99%-mnist handwritten digits recognition*. Medium. Retrieved December 7, 2022, from https://towardsdatascience.com/going-beyond-99-mnist-handwritten-digits-recognition-cfff96337392