EEL 5840 Final Project - POW9001

1st Shantanu Kapoor
University of Florida
Gainesville, Florida, USA
s.kapoor@ufl.edu

2nd Blas Kojusner
University of Florida
Gainesville, Florida, USA
bkojusner@ufl.edu

3rd Shania Shakri
University of Florida
Gainesville, Florida, USA
shakrishania@ufl.edu

***Abstract*—Machine Learning can be leveraged to answer some of the most difficult challenges required of a computer. Within Machine Learning lies Neural Networks, a state of the art method for visual signals used in handwriting recognition. In this paper, we implement and compare past and current machine learning models for handwritten symbol recognitnon on 10 class labels. Our notable findings include understanding the capacity and scalability of simple and complex machine learning methods. Ultimately, the results show that high dimensionality proves to be a hindrance for kNN and Naive Bayes from achieving the best results, despite the models being simple and efficient to implement. Additionally, the study demonstrates SVM can handle higher dimensional data, yet it is CNN that ultimately manages to perform the best with the dataset provided.**

I. Introduction

Neural networks are a subset of machine learning that mimic the way biological neurons send signals to one another to leverage it for a high performance interpretation of a data set. They are comprised of a node layer containing the input, one or more hidden layers, and an output layer. Each node connects to one another and has an associated weight and threshold. If the output of a node is above the associated threshold value then the node is activated and it will send data to the next layer of the network. These networks rely on training data to learn and improve their accuracy over time as well. Once accurate to an acceptable degree, the networks can be used to reliably classify and cluster data at a high throughput. In this study, we want to see how we can leverage a neural network to create a model specifically intended to recognize mathematical symbols from an image.

Image classification refers to the task of identifying an image and categorizing it in one of the predefined classes. Image classification can leverage localization, which is when a bounding box is drawn around an object in an image in order to categorize it under a certain class. It can also include object detection which can categorizing multiple different objects in the image and localize them individually. There has been extensive research done in this field, referred to as Computer Vision, that can help guide the direction of future work to be done under the domain.

There are different architectures a neural network can undertake. Some of these are expounded upon in a paper written by Sultana et al. [1]. In this paper, advancements can be noted from models including ZFNet [2], VGGNet [3], GoogLeNet [4], ResNet [5], DenseNet [6], CapsNet [7], and SENet [10]. These are all models that have leveraged their technology to consistently lower the ImageNet [11] top-five error rate.

Previous work in the area has primarily focused on using data that has been recorded on a computer as a sequence of coordinates as mentioned by the work done by Nguyen et al. [8]. If the study uses different sorts of data, then it shifts focus towards exploring different relationships between elements in the architecture of a network, as done in Hu et al. [10]. The papers that focus on optimizing a networks performance usually tend to use data sets that do not consist of handwriting, but rather focus on images of objects or faces. Additionally, the models in previous work were usually trained on a similar background with equal size symbols which made it easier to extract symbol segments. However, in the presence of complex backgrounds and different width pen strokes the simple models suffered. As such, in this paper we wanted to combine using non-computer based data to explore the applications of optimized neural networks on recognizing and labeling symbols on diverse backgrounds.

To address this gap, we studied different models to best take on the task and ultimately propose the adoption of CNN when conducting this task. The CNN has various filters at different levels that can extract low level features like edges or different smoothing filters for different sized images, with complex filters at higher layers that can separate the foreground form the background which was usually ruled paper in our case.

II. Implementation

In this section, we discuss the implementation of the three most crucial parts of our study, which include the data and how it was generated and labelled, the model selection and the purpose for selecting each model, and the the final step which is the implementation that includes preprocessing, training, and evaluation.

A. Data

The dataset consists of 10 mathematical symbols. These symbols are x, square root, plus sign, negative sign, equal, percent, partial, product, pi, and summation.

1) Data Generation: The data is generated manually by students in the EEL5840 course. The students form groups and each individual in the group is to collect 10 images for each math symbol amounting to a total of 100 images per student. The students were allowed to take images of varying quality, draw symbols of varying sizes, and use different canvases and colors for the symbols they would draw.

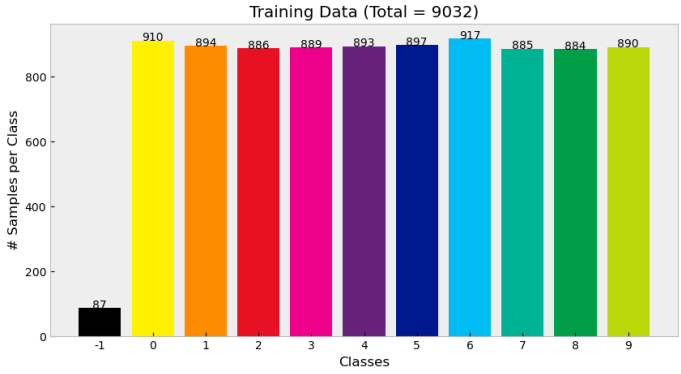

Fig. 1. Adjusted Training Data Distribution.

2) Labeling: The data set is then compiled and delivered to all students in the course. There are clear misclassifications that need to be adjusted in order to have more representative data for each of the labels to train our model. Figure 1 demonstrates the number of samples per class after the students in group POW9001 have gone in and manually verified the images in each set. The figure also shows a label -1. This label represents samples that do not fit under any of the 10 mathematical symbols for the study.

B. Model Selection

1) Naive Bayes: The Naive Bayes model is simple to implement and can work with large data sets like the one presented in this study. This model is based on Bayes' Theorem and carries the assumption that predictors are independent. It essentially assumes that just because one feature is present in a class, it does not relate to the presence of any other feature in the same class. Implementing this model can also help make conclusions about the different features that are present in a class.

2) kNN: The kNN classifier for this study is modeled after the work done by Das et al. [9] as it is one of the simplest computing ML algorithms. This is because kNN does not require to train a model or fine tune parameters, hence making it a faster computing algorithm when compared to other classifiers like SVM or linear regression. All this model requires is sufficient memory to store the data that is used directly and learned from at the time of prediction. The model works by comparing samples and although the requirement of large amounts of memory increases costs, kNN does a fair job with providing accurate predictions. Another reason for selecting this classifier is that it cna easily incorporate new data.

3) SVM: The SVM model is chosen because, as opposed to kNN, the SVM classifier works well with high dimensional data. The risk of over-fitting in SVM is also lower than the previous two models since the data points are separated by a hyperplane. This becomes a helpful tool in the separation of unstructured data correctly into different groups as the separation marks the decision boundary. The separation is also said to be good for a hyperplane that has the most distance to the nearest training data point. The selection of the hyperparameter for this model has to be such that we attain maximum accuracy since after reaching a certain value of Gamma the accuracy starts decreasing.

4) CNN: This model was selected as it is a more complex and involved implementation that would likely yield the best results in our study. This model allows us to make an architecture that can take in an input image and then assign weights and biases to various aspects of the image such that it will be able to differentiate and properly classify other images. The specific implementation for this study is a custom architecture that takes the first layer of Alexnet and uses fewer features in order to minimize computational power consumption. The architecture is represented in Figure 2.

C. Implementation

The study is conducted on HiPerGator as it offers the resources necessary to be able to run the model training for our experiments.

1) Preprocessing: For this study, the group has implemented the data preprocessing strategies proposed in a paper written by Jimenez et al. [8]. The data preprocessing techniques can be split into the two techniques we used. The first technique used was applied to CNN and the second technique used was applied to Naive Bayes, kNN, and SVM.

The first technique used would first convert the image to be gray scale and would then normalize the image and augmented it by rotating it 30 to 40 degrees and horizontally flipping it.

The second technique also converted the image to gray scale at first, however, it would then deploy a Gaussian filter of 7x7 to remove any high frequency noise. We then applied thresholding to segment the image and afterward we performed image dilation which is a morphologoical method that thickens lines within an image. The final step in this technique was to skeletonize the image so that it is a 1-bit image.

Lastly, HOG makes an effective feature extractor for the handwritten symbols as the high intensity gradients will be more concentrated towards different strokes highlighted as 1 bit image. In each cell size of NxN the gradients were calculated for 8 orientations and one with the maximum number of votes from pixels in each cell were chosen [8]. For our case we used two cells one $5x5$ and $10x10$, forming a pyramid of gradients at different cell resolution. This produced a feature vector of 4000 length that was passed to traditional machine learning models.

2) Training: The Naive Bayes model was trained by iterating through the data set and calculating the data likelihoods of each image in each of the 10 different classes from this study. This step made a call to $multivariate\_normal.pdf()$ and passed through the image, the mean, and the standard deviations for each class. once the likelihoods were calculated, the posterior is then calculated using the posterior formula. The returned value is the value with the largest posterior probability of the 10 classes.

The kNN model is a lazy learner and does not require training. The model is instead tested to start finding the

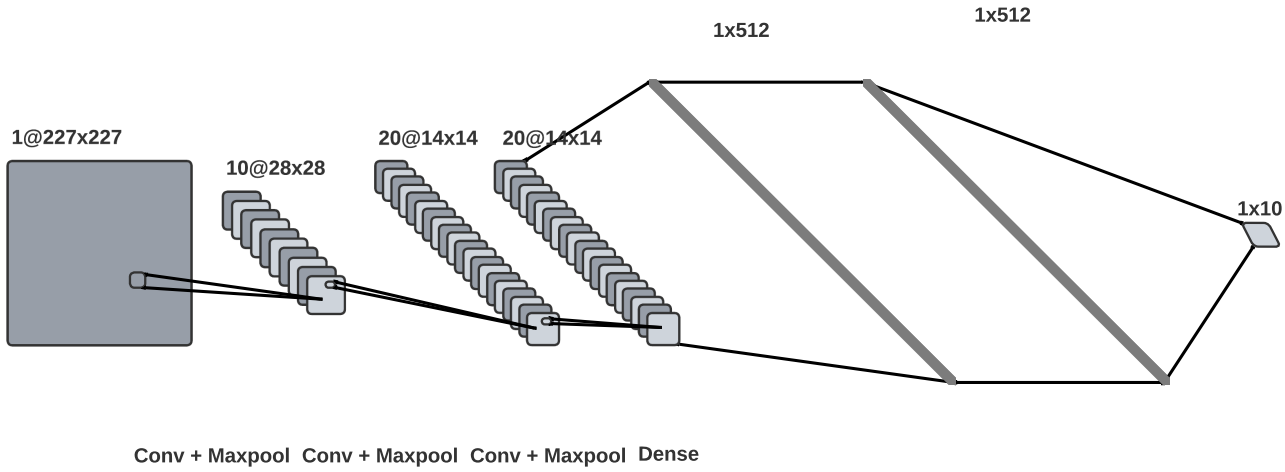

Fig. 2. This is the proposed CNN Architecture that we implement in the study.

nearest neighbors between two sets of data. The unsupervised algorithms within $sklearn.knearestneighbours$ are used for this part. This k-Nearest Neighbour algorithm calculates the distance metric we set, such as Euclidean, between any test and training data points. With this, we then initialize an integer value of K and let the algorithm iterate from 1 to a value n based on the size of the training data. The top nearest neighbors are then selected and the most frequented class values are returned.

Since SVM is a binary classifier, we used a one to one approach that generated $(n * n - 1)/2$ binary classifiers for multi-class classification. The classifier implemented came from the $scikit$ library and was used to fit the PHOG features. The output label is the class with the maximum number of predictions from the binary classification.

The CNN was trained on a custom model with a gray scale image of binary size 227x227. The image was resized to be $(227x227)$. The first layer is similar to Alexnet with $10@11x11$ with 4 strides. The next consecutive layers are with $20@5x5$ and $40@5x5$ filters respectively. For each CNN layer, ReLu activation function was used along with batch normalization and max pooling of size 2x2. The final vector from CNN is flattened to a 1960 1D feature vector which is passed through a 2 layer DNN architecture with 1024 nodes in each layer for extracting class specific features. Each hidden layer has dropout rate of 50% to avoid any kind of over-fitting. The output layer contains 10 nodes with softmax layer. The argmax is taken as 1 encoded vector output. Categorical Cross Entropy loss, generally used for classification was used to optimize the above model.

3) Evaluation: The evaluation process for Naive Bayes consisted of comparing the results of the prediction to the intended labelled outcome. The evaluation code would iterate through the validation set, make a prediction, and evaluate the classification as to whether or not the sample landed in the correct class. The group then used the average mean squared error to obtain a valid benchmark in determining how effective Naive Bayes was with making predictions for the dataset.

To evaluate our kNN model we test the dataset by selecting different values of k. It is seen that the best results are obtained at k=1. For evaluation we use the F-1 score metric. With the F1 scores compared to accuracy provided by the other models we decide how competitive kNN is for this dataset.

The evaluation techniques used for SVM and CNN are explained in the previous section as it goes into more detail exactly how the pipeline will make use of the Categorical Cross Entropy loss function to optimize the models.

III. Experiments

A. Naive Bayes

The overall implementation of Naive-Bayes came with its challenges. The high dimensionality of the priors yielded different results than expected. Bias was multiplied to an identity matrix and then added to the covariance to correct the large number of features, however the results for the bias would usually yield zero considering the dimensionality of the data would lend for the numbers to be small enough to round down to zero. After some adjustements in the bias, the posterior probabilities would return numbers that were out of range for what one should see in this study. Overall, Naive Bayes does not seem to properly account for non-binary classifiers as there are a total of 10 classes, means, and covariances to account for. It is clear that the data is too high dimensional for the model to handle.

B. kNN

While working with this dataset it was seen that although kNN is easy to implement, it does not work well with a high dimensional dataset. The main challenge while using the kNN approach was finding the value of K which works most

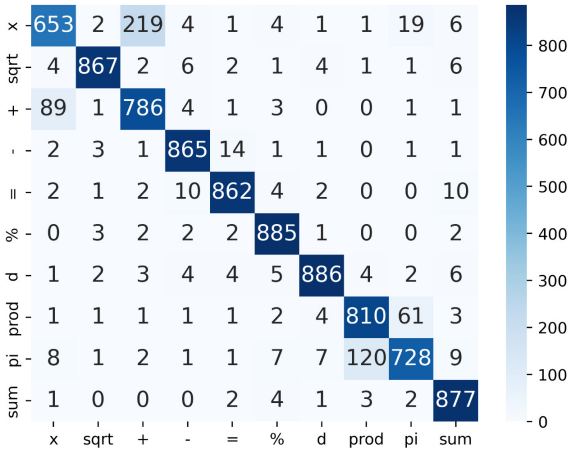

Fig. 3. Confusion Matrix for CNN performance on the entire data set.

effectively for accuracy across a large dataset. This is done by data mining tasks which is a cost sensitive task. Optimal results are obtained when K=1 as in this case the nearest neighbour is considered. The low accuracy of kNN may also be attributed to it being a lazy classifier. It is not trying to optimize any error for the classification but only memorizing data.

C. SVM

The implementation of SVM was able to overcome the curse of dimensionality and fit well to the PHOG features. Despite this, due to sparse representation of the data, the model overfitted which yielded an accuracy of 65% on the validation set.

D. CNN

The CNN implementation allowed us to tackle the issue that more traditional models endured as a result of the poor preprocessing used for PHOG feature extraction.

The CNNs, with weight sharing and attention to local region with kernels, can extract multiple locally sensitive features like edge, orientation and scale. Furthermore, these local features are concatenated in subsequent higher layers which produce higher level abstract filers. Here, we assume that the filters are able to identify different polynomial strokes for symbols. We also assume that the filter is able to find and subdue similar information, like paper and noise, where our manual preprocessing struggled. This conclusion is based on 90% accuracy achieved in both training and validation set. A confusion matrix visualizing this result is presented in figure 3.

None of the models tested in this paper achieved a 100% accuracy as can be seen on the accuracy plot for CNN in Figure 4, our highest performing model. There is ambiguity present between symbols like x and multiplication symbol X. Due to this similarity, both tend to get classified interchangeably. Similar issues exists with symbols pi and $product$. Manually labelling the dataset was also difficult as the symbols

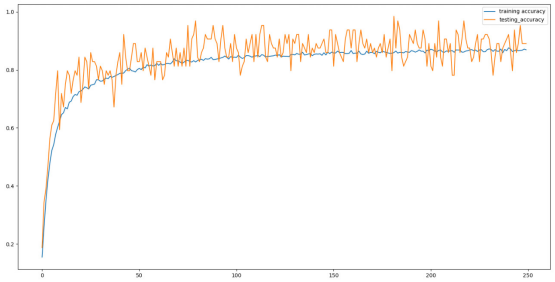

Fig. 4. Accuracy Plot for CNN on training and validation set.

above where not produced in a controlled environment nor under precise guidance.

IV. Conclusion

In this study we analyze different machine learning implementations to recognize handwriting. We have shown here that we can leverage a CNN model to do the work we want. Our study examined four different models to comprehensively determine the effectiveness of each one in classifying handwritten mathematical symbols. The study demonstrates the effectiveness of models like SVM and CNN when compared to more traditional and simpler models like kNN and Naive Bayes. Our study concludes that, from this group, CNN performs best when dealing with data that has poor preprocessing.

References

[1] F. Sultana, A. Sufian, and P. Dutta, "Advancements in image classification using convolutional neural network," 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), IEEE, 2018, pp. 122-129.

[2] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in Computer Vision – ECCV 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[6] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," CoRR, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[7] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," CoRR, vol. abs/1710.09829, 2017. [Online]. Available: http://arxiv.org/abs/1710.09829

[8] N. D. Jimenez, and L. Nguyen, "Recognition of Handwritten Mathematical Symbols with PHOG."

[9] D. Das, D. R. Nayak, R. Dash, and B. Majhi (2019), An empirical evaluation of extreme learning machine: application to handwritten character recognition, Multimedia Tools and Applications, 78(14), 19495-19523.

[10] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," CoRR, vol. abs/1709.01507, 2017. [Online]. Available: http://arxiv.org/abs/1709.01507

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in CVPR09, 2009.