Handwritten Character Recognition using Deep Convolutional Neural Network

Hasan Al-Shaikh, Md Kawser Bepary, Shuvagata Saha
ECE Department, University of Florida, Gainesville, FL-32611
hasanalshaikh@ufl.edu, mdkawser.bepary@ufl.edu, sh.saha@ufl.edu

***Abstract*—Handwritten character, digit, or symbol classification is one of the most studied research areas in image processing and machine learning. Automatic recognition of handwritten symbols from captured images in adverse practical scenarios is significant. This work deals with such a task where 10 different handwritten symbols are detected. We propose to classify these symbols with the help of a Deep Convolutional Neural Network (DCNN). We constructed an appropriate preprocessing pipeline based on empirical observations on the given dataset. With our adopted methodology, we were able to obtain 98.07 % classification accuracy on the test set which we had set aside from the given dataset at the outset of experimentation. A detailed description of our experimentation methodology to set the network architecture and preprocessing pipeline is also presented in this report. We noticed, on average, a 2-3% boost in accuracy with application of data augmentation techniques.**

***Index Terms*—Deep Learning, Machine Learning, Image Processing, Symbol Recognition**

I. Introduction

Since the early 1990s, symbol recognition has gotten a lot of attention as a research topic. As a result, a variety of strategies have emerged as answers to this challenge. Because of its robustness against visual blurring, shifting, and scaling distortions, Deep Neural Networks (DNN) have grown more popular in recent years for symbol/letter recognition. In this project, we have developed a deep learning pipeline to classify images of 10 handwritten symbols. Each student from the class contributed to the dataset by creating and recording at least 10 copies of each symbol. We were provided with the training dataset along with training labels in the form of two Numpy (.npy) files. Some of the images were mislabeled which were rectified through manual inspection in the first check point of the project. In the second phase, we used the rectified label file obtained from the earlier checkpoint and the training images to train the model. At the outset, we set aside 80% of the given dataset as training data, 10% as validation set, and 10% as makeshift test data. The preprocessing pipeline for training we used involved a simple scaling and some morphological transformations for data augmentation such as rotation, erosion, horizontal and vertical shifting, and zooming. The preprocessing pipeline, in its entirety, was applied only on the training dataset. For the test dataset, we only applied the appropriate rescaling.

Multiple DNN methodologies have been proposed in pattern recognition applications, identifying symbols, letters, or miscellaneous drawings. Yuan [1] presented a methodology that starts with preprocessing the images. The preprocessing step is followed by segmentation, and a LeNet-5 based CNN was used for segmentation initially. Then, character recognition is achieved offline with a 92.2% success rate. Wang [2] et al. presented a Convolutional Neural Network (CNN) combined with automatic feature extraction for text recognition from images. Automatic feature extraction is an unsupervised algorithm.

A relaxation CNN (R-CNN) and alternately trained relaxation CNN (ATR-CNN) based handwritten Chinese character recognition method is presented in [3]. The first network could learn more quickly than typical CNNs while the second CNN could be trained on a subset of layers to quicken training and improve accuracy up to 96.06%. A more generalized model applicable across different languages and characters is presented by Bai in [4]. In this methodology, the characters share the hidden layers, and based on the target language, the final layer is trained. This method of training CNN is termed as shared Hidden Layer CNN (SHL-CNN). This method was applied to both English and Chinese characters and exhibited significant improvement on traditional CNNs. A perspective of Image recognition has been discussed in Liu [5] reviewing the challenges and applications of CNNs. Zhang et al. [6] integrated a watcher, a CNN encoder and a parser/decoder (RNN) to form the Watch, Attend and Parse (WAP) methodology. The encoder encodes the images to high-level features which are then transformed to output sequences.

We utilized a relatively straightforward CNN which takes scaled preprocessed images of dimension 150×150 and gradually convolves it to a depth of 512. A flattening operation followed by Fully Connected (FC) layers are used to perform the classification. Given the relatively small dataset, it was evident that data augmentation techniques would be required to construct a larger dataset to help the model generalize better. We experimented by training the network with and without augmentation. The augmentation process greatly boosted the network performance as we had intuitively surmised.

The rest of this report is divided into three sections. We describe the implementation in detail in section II and then present the experimental findings for several configuration of the proposed CNN based architecture in section III. Finally, in section IV, we end our study by speculating on why the proposed network failed to accurately anticipate specific photos.

II. Implementation

In the suggested technique, the input picture is preprocessed before being fed into a neural network for feature extraction in an end-to-end process. As proposed network architecture, we used a hybrid of CNN and FCN. Data augmentation techniques based on manual observation and visualization of the dataset is also applied to combat overfitting. Section II-A and Section II-B present the data preprocessing and data augmentation approaches respectively. Next, Section II-C describes the proposed network architecture, Section II-D describes the objective function and the training scheme is elaborated on in Section II-E.

A. Data Preprocessing Pipeline

At first, two preprocessing methods are applied to training and test data. Although it is expected that the input image will be a grayscale image with the range of pixel values 0 to 255, there are some issues with brightness, clarity and background of the image. The first preprocessing step involves scaling the image to a smaller size so that the training process does not take too long. In our experiments, the original 300×300 pixel images was resized to 150×150 pixel. Resizing the images may result in spatial information being lost which is the reason we refrained from scaling into too small an image.

B. Data Augmentation

The fundamental reason for performing data augmentation is to supplement the training data with additional observations. In actuality, the acquired images from the students may have a variety of non-idealities, including noise, poor lighting, shadows, and haziness. The training data is not large enough to compensate and represent all of these variations adequately. One particularity, for example, of the constructed training dataset for this project is the existence of ‘grid lines’ as background to the images. This stemmed from students writing the handwritten alphabets on ruled paper. The characteristics of the images motivated us to apply in total 9 different data augmentation techniques: erosion, dilation, rotation, vertical flip, left/right shift, scaling, up/downshift, etc. We also superimposed horizontal and vertical black lines on the image to emulate ruled paper.

Figure 1 shows images after applying all these augmentations on a training image sample. Five augmentation methods among the nine are randomly applied for each image in the training set such that rotation and grid line addition must be applied.

C. Proposed Network Architecture

In this work, as neural network, we propose a hybrid of CNN and FCN architectures. The proposed network utilizes what we refer to as ‘layer clusters’. Each CNN ‘layer cluster’ has a conv2D layer (as defined by the Keras API) followed by max pooling, batch normalization and dropout layers. Each FCN layer cluster has a dense layer followed by batch normalization and dropout layers. Between the CNN and FCN layer clusters is a flatten layer which flattens the $3 \times 3 \times 256$ ‘images’ into 1D data. The max-pooling layer with a stride value of 2 is used at the initial part of the network to reduce the dimension of the feature maps as well as to help in overfitting. The

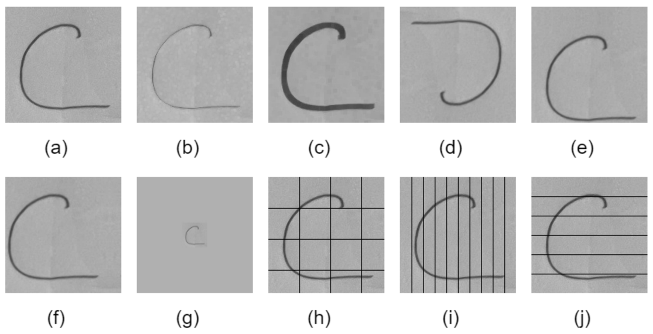

Fig. 1: Different data augmentations used in the work: (a) Original image, (b) erosion, (c) dilation, (d) rotation, (e) vertical shift, (f) horizontal shift, (g) zooming, (h) grid-lines, (i) vertical grid, (j) horizontal grid.

dropout layer also helps in eliminating overfitting and reducing computational burden by randomly dropping a percentage of the connections. The purpose of batch normalization is to normalize the features learnt after the use of each activation function. The activation function used in all of the layers is ReLU except the final Dense layer which has the Softmax activation function. Since the network is subjected to complex problems, ReLU is used to bring the non-linearity. The batch normalization layer also helps to combat exploding gradient problem which might otherwise be accrued from the use of ReLU.

We arrived at this network structure through random search of model hyperparameters such as learning rate, kernel dimensions, dropout rate, model depth and width through experimentation. Some of the significant results from this experimentation can be found in section III.

D. Objective Function

Categorical cross-entropy loss is chosen as objective function. This objective function L is given by

$$L = -\sum_{c=1}^{N} (l_c \times log(\widehat{l_c})) \tag{1}$$

Here, l_c and $\widehat{l_c}$ denote the ground truth and probability of the class prediction for an image. N indicates the number of class.

E. Training Scheme

The mini-batch optimization technique is utilized for training, with the batch size set to 32. For the optimization purpose, Adam Stochastic Optimization algorithm [7] is chosen, with learning rate, decay values for first and second moments are set to 10^{-4}, 0.9, and 0.999, respectively. The Nvidia RTX GPU is used for training purposes.

III. Experiments

Exhaustive experiments are performed to ascertain the viability of the proposed methodology in handwritten symbol classification. Section III-A, Section III-B and Section III-C

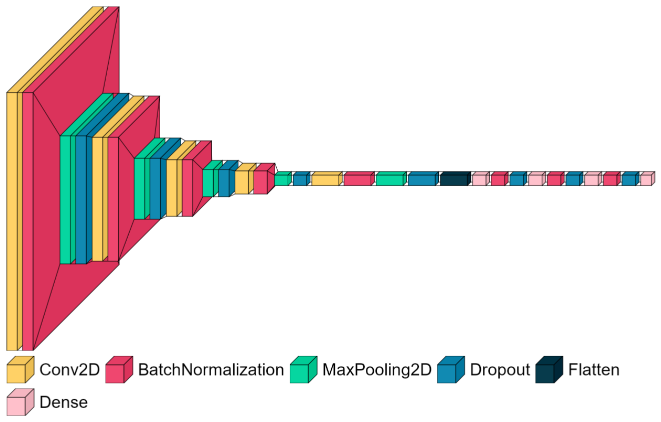

Fig. 2: Proposed network architecture for detecting handwritten characters. 5 CNN layers followed by, 3 FCN layers with a dropout of 0.2

respectively provide detail descriptions of performance metrics, dataset and different configurations of the proposed CNN that we experimented with in this work. Finally, Section III-D provides the results of experiments and also analyzes them.

A. Evaluation Metrics

In this study, the suggested method's performance and that of competing techniques are assessed in terms of accuracy. Accuracy, in particular, as used in our evaluation, draws attention to instances of true positives and negatives. The classification performance is visualized using the confusion matrix. The computational complexity of the model is also estimated in order to analyze and compare the performance of different networks. The overall amount of trainable and untrainable parameters is used to estimate the computational cost of a network architecture.

B. Dataset

After data collection and curation were done by the students of EEL5840, we augmented the dataset using the method discussed in section II. The dilation, erosion, scaling, vertical flip, shearing, rotation, left/right shift, up/downshift, grid line addition techniques were applied solely on the 80% of the images that were kept as training data. In this way, we create an augmented training dataset of size 32256 samples. We kept a portion (10%) of the provided training data additional data as validation data which is only used to monitor the performance of the training and verify whether the model is overfitting or not. Since the main test dataset is not released, we consider the remaining 10% as test dataset.

C. Comparing Methods

The structure of the proposed network and its associated hyperparameters are arrived at by random search and comparing validation performance. We have highlighted six of these experimentation constructs and their outcomes in table I. As we obtained greater than 95% accuracy on the validation set with a simple CNN model, we refrained from using more complex and involved pretrained CNN models (such as: VGG, ResNet etc.). From the comparison shown in the table, it is generally seen that increasing the no. of CNN layer clusters boosted accuracy. Increasing dropout in general also improved accuracy which is consistent with our intuition that more dropout, up to a certain, extent would help the model generalize better. The increase in depth of FCN layers helped increase the accuracy as well.

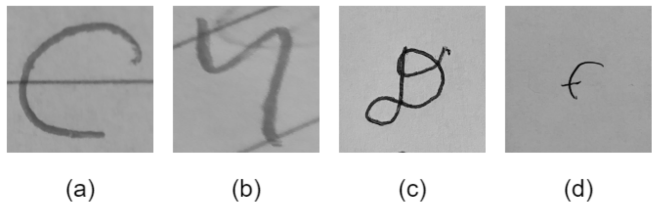

Fig. 3: Four of the images that were mislabeled by the network.

D. Results

The performance of the proposed approach is compared to that of other networks with similar architectures. Table I provides the comparison in terms of accuracy and the computational complexity of the network. From the table, it can be noted that the highest accuracy is found for the model with 5 CNN, 3 FCN layer clusters which also has the highest computational complexity and dropout.

Figure 4 shows the confusion matrix obtained exhibits bright cells along the diagonal which is expected since the accuracy is quite close to 100%. We note from the confusion matrix that most common incidence of mislabeling was that of ‘e’ being mislabeled as ‘c’ and ‘f’ being mislabeled as ‘e’. This can be explained if we take a closer look at the images whose labels were not predicted correctly by network. Figure 3 illustrates four such images. Figure 3(a) and 3(d) are representative of the most common misclassifying incidences. We note that in the first case, due to specific cropping of the image and blurring of the ruled paper, the ruled marking running straight through the middle of the image makes the ‘c’ look like an ‘e’. In the case of the second image, due to the small size of the handwritten letter and curved top portion, the ‘f’ appears to be an ’e’ to the network. Figure 3(b) and 3(c) are instances where due to the unique handwriting of the student, the shape does not match with any other instance of the same class.

Table II summarizes a separate experimentation where the impact of preprocessing and augmentation is analyzed. In this case, the proposed architecture is trained for two distinct scenarios. The first case involves the model being trained on non-augmented training data. In another case, augmentation is used, but data is not preprocessed. In that case, the proposed model achieved 95.68% accuracy on the makeshift test set. The preprocessing pipeline, in general, provided a 2-3% boost in accuracy metrics across all tested configurations as can be inferred from the table It illustrates a strong positive correlation between the data augmentation and the model's performance. It happens because extensive data augmentation helps the model learn the unseen and additional data, which

TABLE I: Comparison of performance with other comparing methods in terms of accuracy and computational complexity

Model	Model Description	Accuracy (%)	Computational Complexity
Model-1	5× CNN + 2× FCN + Dense with Dropout = 0.1	95.83	2.82M
Model-2	4× CNN + 2× FCN + Dense with Dropout = 0.1	95.09	4.65M
Model-3	3× CNN + 2× FCN + Dense with Dropout = 0.1	92.56	525K
Model-4	5× CNN + 3× FCN + Dense with Dropout = 0.1	96.43	2.89M
Model-5	5× CNN + 2× FCN + Dense with Dropout = 0.2	96.13	2.82M
Proposed	5× CNN + 3× FCN + Dense with Dropout = 0.2	98.17	2.89M

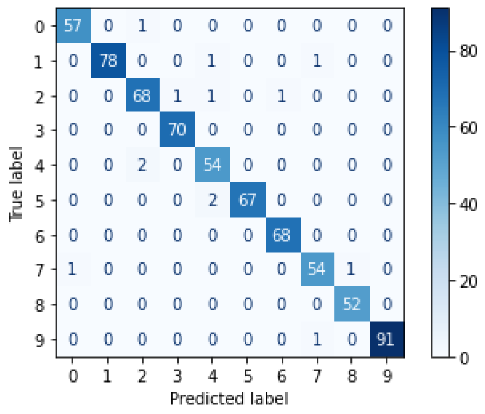

Fig. 4: Confusion matrix for the proposed method

TABLE II: Impact of data preprocessing and augmentation for the proposed method

Preprocessing	Augmentation	Accuracy (%)
Present	Present	98.07
Present	Absent	96.04
Absent	Absent	95.68

will ultimately help recognize symbols in difficult practical scenarios.

IV. Conclusion

Classification of handwritten symbols or characters has historically been a focus of researchers in image processing and machine learning. We successfully classify ten different handwritten symbols from photographs in this work. The dataset was created collaboratively by participants in the EEL5840 course. Prior to implementing deep neural networks, our suggested technique comprises data preparation and data augmentation. We used nine distinct data augmentation techniques to add fresh observations to the training dataset. Our proposed methodology includes data preprocessing and data augmentation before applying deep neural networks. We applied 9 different data augmentations to include new observations in the training dataset. We used a deep CNN to be trained our augmented and preprocessed data. On a makeshift test set, it achieves 98.07% classification accuracy. Our detailed experiments show how data preprocessing, and augmentation significantly improve the proposed methodology's performance. We also explained why the network might have failed to correctly classify some samples in the dataset. We plan to implement more explainable neural networks and other advanced image processing techniques in the future.

References

[1] A. Yuan, G. Bai, P. Yang, Y. Guo, and X. Zhao, "Handwritten english word recognition based on convolutional neural networks," in *2012 international conference on frontiers in handwriting recognition*. IEEE, 2012, pp. 207–212.

[2] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, 2012, pp. 3304–3308.

[3] C. Wu, W. Fan, Y. He, J. Sun, and S. Naoi, "Handwritten character recognition by alternately trained relaxation convolutional neural network," in *2014 14th International Conference on Frontiers in Handwriting Recognition*, 2014, pp. 291–296.

[4] J. Bai, Z. Chen, B. Feng, and B. Xu, "Image character recognition using deep convolutional neural network learned from different languages," in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 2560–2564.

[5] Q. Liu, N. Zhang, W. Yang, S. Wang, Z. Cui, X. Chen, and L. Chen, "A review of image recognition with deep convolutional neural network," in *Intelligent Computing Theories and Application*, D.-S. Huang, V. Bevilacqua, P. Premaratne, and P. Gupta, Eds. Cham: Springer International Publishing, 2017, pp. 69–80.

[6] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, and L. Dai, "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," *Pattern Recognition*, vol. 71, pp. 196–206, 2017.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.