Handwritten Character Recognition using Deep Learning

Upoma Das, *UFID:79808214*, Pantha Protim Sarker, *UFID: 36703333*, Md Latifur Rahman, *UFID: 29884063*, Mohammad Bin Monjil, *UFID: 50782093*
Department of ECE, *University of Florida, Gainesville, FL-32611*

***Abstract*—A convolution neural network (CNN)-based handwritten symbol identification system is implemented in this study. A state-of-the-art CNN structure named ResNet-18 is employed as the final training model. The training dataset which contains 6,720 photographs of handwritten symbols includes 10 different symbols. The dataset contains a wide variety of handwritten symbols that are difficult to detect due to background noises and anomalies. Data augmentation and transfer learning have been the main focus for this implementation. Our method developed a large number of augmented images that approximated the variances in real data using various data augmentation techniques. The final trained model was able to detect all ten classes with a validation accuracy of 99.48%.**

Index Terms*—CNN, ResNet-18, feature extraction, augmentations*

I. Introduction

Handwritten character and numeral recognition has grabbed the interest of computer vision and image processing researchers. The main reason for this is that this program is used in a wide range of applications, including automated mail sorting, automated bank check processing, automatic automobile plate recognition, and digitisation of old historical handwritten documents, among others [1] [2] [3]. Moreover, in the field of pattern and graphics identification, symbol recognition is extremely important. The interpretation of domain-specific graphical notations on documents – such as maps, musical notes, traffic signs, and so on – is an exciting application of symbol recognition. [4]. Existing statistical techniques to automated symbol identification confront a number of difficulties, including choosing efficient pre/post-processing methods, identifying acceptable features, and extracting those features using appropriate statistical models. [5] [6] [7]. Recent developments in machine learning and artificial intelligence have made great progress in building a more efficient system for symbol identification by overcoming these current hurdles. Due to the different graphical properties of different domain-dependent symbols, as well as non-uniformity in the image acquisition environment, developing a neural network that works efficiently in every scenario is dependent on the quality of the collected data, model structure, and parameters chosen. All of these factors are heavily influenced by the application. Although deep learning-based symbol identification techniques are more exact and versatile than traditional computer vision and statistical algorithms, they require a lot of computational power and a large representative data set. But with advent of integrated circuit technology, computational power is abundant and with many data generating tools/sensors being available and the connectivity provided by the internet, the world is generating massive amount of data every year. This steered the pattern recognition paradigm and the current trend is to train large neural network models with lots of data which provides automatic feature extraction and accuracy way higher than traditional algorithms.

The purpose of the study is to develop a system that could efficiently categorize ten different handwritten symbols. To accomplish this, we first chose a representative handwritten symbol data set that adhered to the standards. The collected handwritten characters were then successfully recognized with 99.48% percent validation accuracy using a convolutional neural network-based approach.

The project report includes the following sections: implementation, experiments and conclusion. Section II presents an overview of the implementation details. Section III describes data set analysis, data augmentation, training. The result analysis and algorithm performance are then shown this in section too. Section IV contains important takeaways as well as obstacles. Finally, the study offers some recommendations for future improvements.

II. Implementation

The objective of this project was to develop an end to end machine learning algorithm that can classify handwritten characters into ten classes. The project was divided six steps, as outlined in Fig. 1. The work started with data collection. In this project, data were handwritten characters on physical paper which include a, b, c, d, e, f, g, h, $, and #. The training dataset consists of a total of 6720 grey-scale images of 300x300 pixels each. These data were then split into train and validation data with a ratio of 80%/20%. The next step involved a deep observation of the nature of the data itself. After careful observation of sample images from the data set, we used data augmentation to increase the size of our training data set. Next, we selected and prepared our model for the unknown testing data. For model selection, we used Resnet architecture [8] for the classification task. The model is trained with both original and augmented images during the training phase.

The core idea of residual network (*ResNet*) is that each added layer should contain the identity function as one of

*All the authors equally contributed to this project.

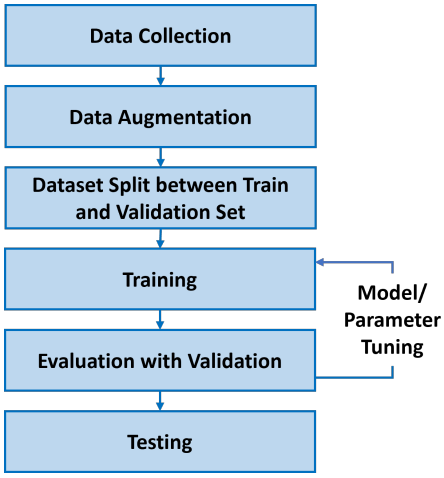

Fig. 1: Step by step implementation.

its elements. This is easily done by a a residual block. Let's assume the input as x. The learning outcome of mapping, $f(x)$ will be fed to the activation function. If the identity mapping, $f(x)=x$ is the desired underlying mapping, which is the case for complex deep learning model, the residual mapping, $f(x)-x$ is much easier to learn. The Fig. 2 illustrates the residual block of *ResNet*, where the solid line carrying the layer input to the addition operator is called a residual connection. With residual blocks, inputs can forward propagate faster through the residual connections across layers. These residual blocks overcome the vanishing gradients problem associated with other deep Neural networks (e.g., *AlexNet*, or *VGG*). in theory, deeper network should improve the training and test accuracy. But in reality, with the network depth increasing, accuracy gets saturated and then degrades rapidly. *ResNet* overcomes these limitations with the skip connections [8]. This is the reason for choosing *ResNet* among all the deep learning algorithms.

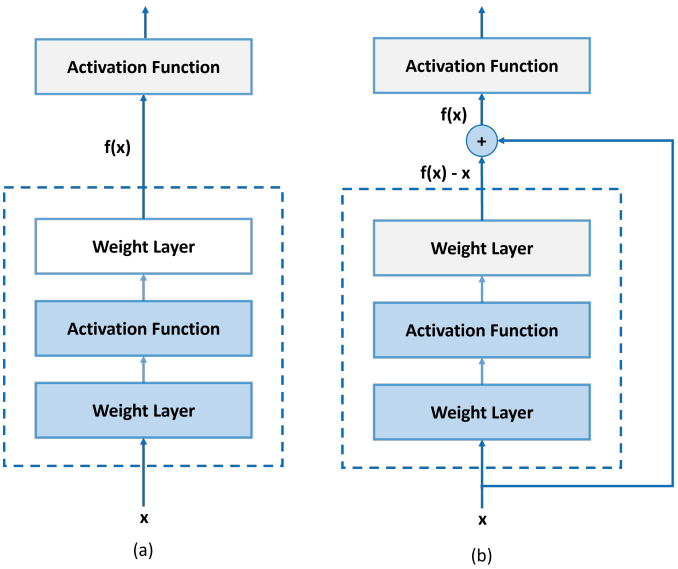

Fig. 2: (a) A regular block. (b) Residual block.

ResNet follows *VGG*'s [9] full convolutional layer design. The residual block has two convolutional layers with the same number of output channels. There are 4 convolutional layers in each module. Each convolutional layer is followed by a batch normalization layer and a ReLU activation function. The number of channels in the first module is the same as the number of input channels. Together with the first *7x7* convolutional layer and the final fully-connected layer, there are 18 layers in total.

We leveraged transfer learning and introduced our training with pretrained weights for *ResNet* model. The pretrained weights helped the models to perceive a initial knowledge of basic feature extraction and converge faster. The pretrained weights were obtained from training over the Imagenet [10] dataset that contains 1000 classes. The pretrained weight was capable of extracting basic features from our images. For our dataset with 10 classes, we replaced the last maxpool layer with an adaptive concatenated pooling layer, and the original fully connected layers with 1000 outputs with a new fully connected layers with 10 outputs. Our experimental model architecture is shown in Fig. 3.

III. Experiments

A. Data Set Analysis

To find the nature of the data we started our analysis by careful inspection of the individual data samples. We observed several issues with our dataset. We found the data to be noisy, faded, zoomed-in, zoomed-out, inverted, blacked-out, and distorted images. Even some images had background shadows, and imprint of other symbols in the background. Fig. 4(a) lists some examples of our data set. If we consider a real life test scenario, the test images could have same types of distortions, and noise. So to maintain generalization in the training dataset, we chose not to apply image preprocessing like denoising. Because, if the model trains with denoised image and sees a noisy image during testing, the model would not perform well. So we applied several data augmentations methods over these training data sets to increase our training data size. To get the true sense of the train data, we also observed the class distribution in the training dataset. From Fig. 4(b) we cam see that classes in the training data are almost in balance. So the model will not be biased towards any particular class during training.

B. Data Augmentation

Data augmentation is an important part of our experiment. The training images of our dataset were taken in a limited set of conditions. But, our target application may exist in a variety of conditions, such as different orientation, location, scale, brightness etc. We account for these situations by training our neural network with additional synthetically modified data which is called data augmentation. The applied random augmentations over our training dataset include crop, horizontal flips, and rotations. These augmentations help the model to be generalized and perform well over unseen data and prevent overfitting.

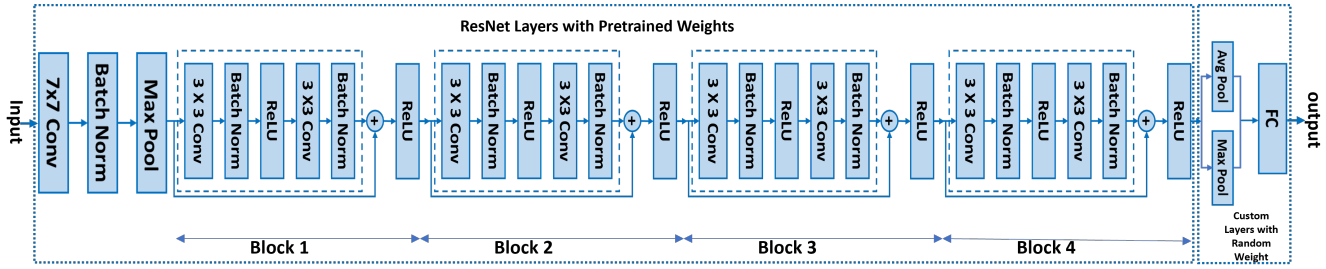

Fig. 3: Implemented *ResNet* model architecture.

Fig. 4: (a) Some sample images of our training data set, (b) Training data distribution.

C. Training

We have used transfer learning and initiated our training with pretrained weights for *ResNet-18, ResNet-34*, and *ResNet-50*. The pretrained weights gave the models a initial knowledge of basic feature extraction and helped it to converge faster. The pretrained weights came from training over the Imagenet [10] dataset that contains 1000 classes. For our dataset with 10 classes, we replaced the last maxpool layer with an adaptive concate pooling layer, and the original fully connected layers with 1000 outputs with a new fully connected layers with 10 outputs. Our new experimental model architecture is shown in Fig. 3. The pretrained weight was capable of extracting basic features from our images.

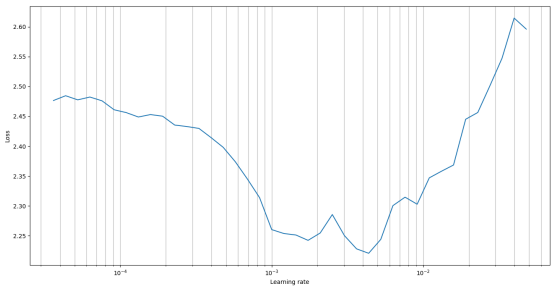

Fig. 5: Finding the most effective Learning rate.

D. Results & Discussion

As mentioned in Section II, we have used transfer learning, and the pretrained weight came from Imagenet dataset [10] , which is a dataset consisting of RGB images while our images are grayscale. So, the pretrained weights required training to learn to extract grayscale features. On the other hand, this training procedure may overfit our data very fast. To mitigate both issues, we have used a learning rate finder that searches for the most efficient learning rate within a range of 10^{-7} to 10. As shown if Fig. 5, the loss was diverged at 4.365×10^{-3} which is our final learning rate. We have trained the custom top layers with this learning rate, and layers with pretrained weight with smaller learning rates ranging from 4.365×10^{-4} to 2.18×10^{-3}. Thus, we have ensured that the pretrained weights change slowly to adapt to grayscale feature extraction. We have optimized the model using *cross entropy* loss.

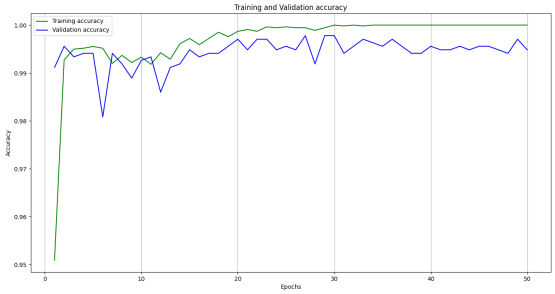

Fig. 6: Training and validation accuracy of our trained *ResNet* model.

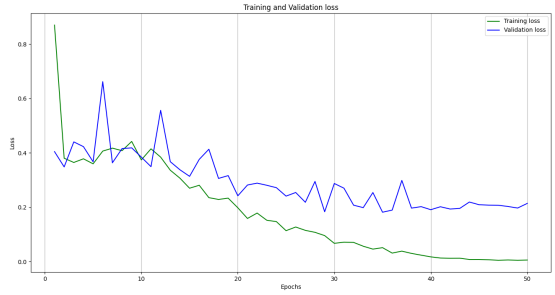

Fig. 7: Training and validation loss of our trained *Resnet* model.

We have trained several models from the *Resnet* family with our dataset. Table. I demonstrates the accuracy and simulation

time for *ResNet-18*, *ResNet-34* and *ResNet-50*. Each of these three models demonstrate similar performance but the simulation time varies for each model. Since the *ResNet-18* model is the smallest one with minimum time complexity, we have chosen *ResNet-18* as our final model for this project. Fig. 6 gives us the training and validation accuracy and Fig. 7 shows the training and validation loss during the training phase. From these two figures, it is evident that both the training and validation accuracy is increasing with the number of epochs. These two graphs were used to determine whether the model is converging or not. As evident in Fig. 7, the Training and validation loss reaches the minima after around 20 epochs.

TABLE I: Performance Comparison for Different Models

MODEL NAME	AVERAGE SIMULATION TIME PER EPOCH (SEC)	VALIDATION ACCURACY
RESNET-18	27	99.48%
RESNET-34	43	99.85%
RESNET-50	47	99.71%

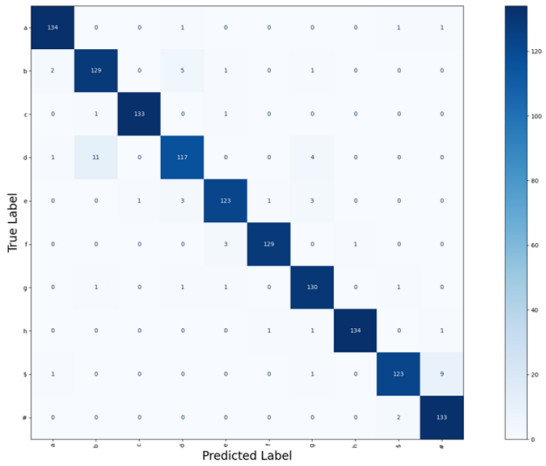

Fig. 8: Test set Confusion Matrix.

We have tested our trained model with 1347 unseen images to evaluate it in terms of accuracy. The test set accuracy was 95.4%. According to the confusion matrix (Fig. 7), our model is misclassifying two specific classes ('b' and 'd'). In most of the misclassification cases, it is predicting either class 'd' as class 'b', or class 'b' as class 'd'. Additionally, In some cases, it is predicting class '$' as class '#'. These are some flaws associated with our model.

IV. CONCLUSION

In this project we have successfully implemented an end to end machine learning system for handwritten character recognition for 10 class. The steps included data collection, labelling, model selection, training and validation closely resembling a real life scenario. We explored several Deep Learning architectures including ResNet-18, ResNet-34 and ResNet-50, the accuracy of these models are comparable but ResNet-18 being least computationally intensive we have selected it as our final model. Using data augmentation we have prevented our model from overfitting and using pre-trained weights trained on ImageNet we reduced our training time considerably by getting convergence within 20 epochs. Our trained model achieved an impressive accuracy of 99.48% on the validation set. Overall, the project experience has been a valuable one which will greatly help us tackle future machine learning problems we face.

REFERENCES

[1] F. Ye and A. G. Bors, "Learning joint latent representations based on information maximization," *Information Sciences*, vol. 567, pp. 216–236, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025521002449

[2] F. Wang, H. Zhu, W. Li, and K. Li, "A hybrid convolution network for serial number recognition on banknotes," *Information Sciences*, vol. 512, pp. 952–963, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519309211

[3] X. Peng, D. Chen, and D. Xu, "Hyperplane-based nonnegative matrix factorization with label information," *Information Sciences*, vol. 493, pp. 1–19, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519303408

[4] J. Lladós, E. Valveny, G. Sánchez, and E. Martí, "Symbol recognition: Current advances and perspectives," in *GREC*, 2001.

[5] J. Lladós and G. Sánchez, "Symbol recognition using graphs," in *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, vol. 2. IEEE, 2003, pp. II–49.

[6] D. Anderson, C. Bailey, and M. Skubic, "Hidden markov model symbol recognition for sketch-based interfaces." in *AAAI Technical Report (6)*, 2004, pp. 15–21.

[7] M. M. Luqman, T. Brouard, and J.-Y. Ramel, "Graphic symbol recognition using graph based signature and bayesian network classifier," in *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 1325–1329.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] S. Tammina, "Transfer learning using vgg-16 with deep convolutional neural network for classifying images," *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 10, pp. 143–150, 2019.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.