Handwritten Character Recognition Based on CNN

Zhiyun Ling, *M.S. of CISE, University of Florida*
Wanyu Dong, *M.S. of ECE, University of Florida*

***Abstract*—This paper is aiming at training a classifier for a 9-class dataset(with one for unknown). Data augmentation and parameter comparison were applied and tested during the model selection. Final model is based on a two-CNN-layer network. During the accuracy examination on a unseen test set, the accuracy achieves 98% on 2-class data, 98% accuracy on 8-class data, and 95% for 9-class.**

***Index Terms*—Handwritten Recognition, CNN.**

I. Introduction

WITH neural network, we can train models by fitting them with large amount of data, and use the model as an inference tool in almost every field, such as medical image segmentation and natural language processing. Apart from optical character recognition (OCR) which targets machine printed text, handwritten character recognition (HCR) has many more challenges. Before neural networks, character recognition often follow a pattern of data collecting, data augmentation, segmentation, feature extraction, model training and validation. [1]

In 1998, LeCun et al open the gate for CNN (Convoluted Neural Networks) by applying gradient-based learning method to document recognition field. [2] Later in 2012, AlexNet brought deep CNN in popularity among image classification practitioners by winning 2012 ImageNet. [3] Coming to 2015, Google DeepMind team proposed Spatial Transformer Networks which enhances resistance to image transformation such as scaling and rotation, by simply introduces them on the input.[3] With refreshing state-of-art method and volume-increasing dataset plural in laboratory and competition, some considered HCR an almost solved problem for many mainstream languages. [4], [5] Yet, it holds true under the assumption of the presence of gigantic dataset and computing power, as deeper networks requires factorial increasing parameter space to represent. In other words, it is based on large scale dataset, which takes effortless work and time to collect.

In this paper, we constructed two CNN based classifiers, for 2-class, 8-class and 9-class on a unique course dataset.

As in Fig.1, the dataset contains "a,b,c,d,h,i,j,k", totally 8 kinds characters (we refer to handwritten lowercase English characters) written by over 100 students taking EEL5840 course at UF. The additional 9th-class represents the character other than the 8 characters as *unknown*. Unknown class can be experienced by deep learning engineering while applying networks to real-world applications. [6]

Wanyu Dong was with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, 32608 USA.

Zhiyun Ling was with the Department of Computer and Information Science and Engineering, University of Florida Gainesville, FL, 32608 USA.

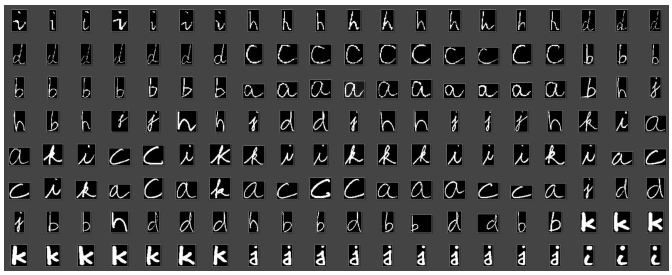

Fig. 1. Visualization of Characters in Dataset

In the meantime, data cleaning and augmentation with spatial transformation was incorporated with input during training and testing period. Output thresholding is chosen for deciding the unknown class in final model.

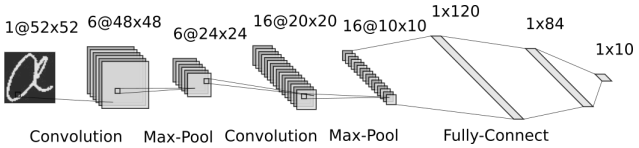

Fig. 2. CNN Model for 2-class / 8-class Classifier

II. Implementation

A. Convolutional Neural Networks

Convolutional Neural Networks(CNN) is one of the artificial neural networks proposed by LeCun et.al in 1998. [2] One of the most common use of CNN is image classification, which means that given input images, CNN outputs class labels for all of them. Typically, a CNN includes convolutional layers, pooling layers and fully connected layers, as in Fig.3.

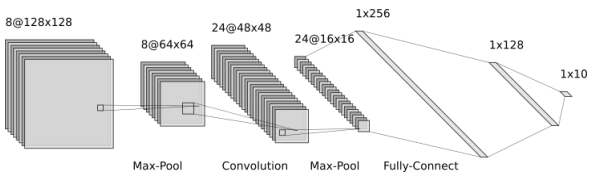

Fig. 3. A typical CNN diagram

1) Convolutional layer: Based on convolution operation, which is common in computer vision, this layer can extract features from input image. Every 2D image is a matrix, and the kernel filter, usually a 3x3 or 5x5 matrix, performs convolution with the input. After the kernel passes all the data point on an image, the output tends to be smaller, which preserves low

level features. The length of output matrix can be calculated as:

$$W_{out} = \frac{W_{in} - F + 2P}{S} + 1 \quad (1)$$

where W_{in}, F, P, S represent the width of input matrix, filter, padding and stride respectively. [7]

2) Pooling layer: This layer can reduce the matrix size by combining multiple data points into one. Typically, a 2x2 maxpooling is chosen in most cases. It works by sliding a 2x2 kernel throughout the matrix with a stride of 2, and choose the max value as the output. The output matrix will become one-fourth of input matrix at size since width and height is halved. Its a down-sampling method and proved to work well in preserving information and reduce data size. [7]

3) Fully connected layer: This layer is the basic neural layers, which connects every neuron from input to all neurons in the output, as the last three layers in Fig3. And it is a flatten operation, which turns input into a vector. Early ANN are composed by this kind of layers. But it has two many parameters, compared to the layer with shared parameters. In practice, several fully connected layer are present at the end of a CNN. The first fully connected layer in a CNN, takes the inputs from the feature analysis and applies weights to predict the correct label, while the last fully connected layer gives the final probabilities for each label. [8]

B. Dataset visualization and augmentation

As mentioned above, the dataset contains characters written by different students. More importantly, these data are scanned, preprocessed and uploaded individually. Proven by later experiments, these data varies in not only writing style but also in size, position, stroke, and background noise. To gain a clear view about what the data looks like, and how the data is present, we starts by visualizing the whole dataset and identify outlier or anomaly data points which could harm the model. On visualization of data,as in Fig.1, we observed that character images are mostly clean and identifiable with human eyes, while some are "dirty" data, which falls in below catagories, and spatial transformation are applied accordingly.

1) Unidentifiable data: As in Fig.4, some of data are totally white and no information is present, while others are full of white surroundings possibly due to overexposure , which is a common problem in real-world data collecting. Since these outlier are hard to identify even by human, we clean data by discarding this group of outlier, otherwise it may impose harm to accuracy.

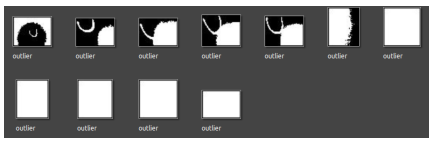

Fig. 4. Visualization of Characters in Dataset - Unidentifiable image

2) Low resolution: As in Fig.5, several data are in low resolution compared to most images, and can be challenging for our model. These images are possibly generated in a down-sampling manner. Another similar feature in data is the stroke, as in Fig.5. Different people write in different pen so the stroke varies significantly. To improve model resistance to low resolution data, we incorporates random down-sampling to our input dataset.

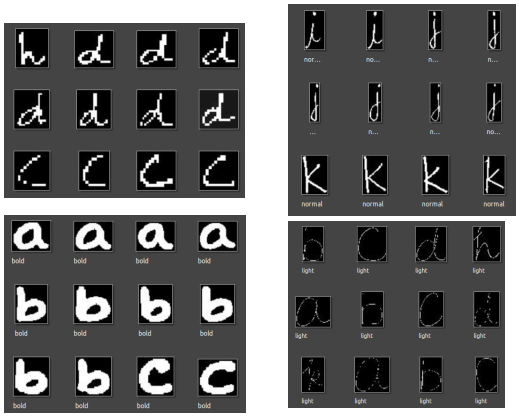

Fig. 5. Visualization of Characters in Dataset - Less Resolution and Different Stroke- light, bold, normal

3) Affine transformation: As in Fig.6, quite a mount of data are inherently transformed and can impact model significantly. These affine transformation includes rotation, translation, and scaling, which is very normal for unconditioned data. These noise can justify the robustness of a model. To tackle that, we applied spatial transformation method by adding affine transformation to input images. Rotation, scaling, and translation are applied randomly.

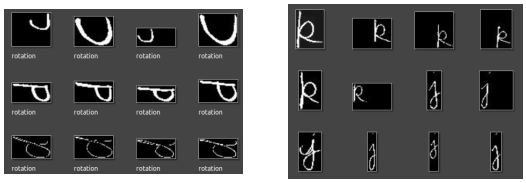

Fig. 6. Visualization of Characters in Dataset - Affine Transformation

4) Writing style: As handwriting is highly personalized, the shape for the same label written by different people varies. As is showed in Fig.7, different styles can be confusing even with human eyes. For instance, some handwritten "i" are similar to "j", and "b" are quite different. Even handwriting by same people can be confusing. So, it's important for our model to generalize over the dataset, rather than remember one style well. L2 Regularization is incorporated to prevent over-fitting.

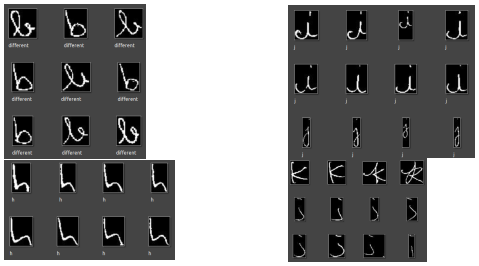

Fig. 7. Visualization of Characters in Dataset - Writing Style

Additionally, images are not in the same shape, which means they cannot be converted to numpy array and tensor

directly. Since the longest dimension, either width or height, is 50 pixels, we first padded the images into 52x52, and then normalized them into [-1,1] range.

C. Model Configuration

For both the subset containing only 'a' and 'b' characters and the whole dataset, a model is created, as is in Fig.2.

Inputs are 52x52 sized images with batch size at 4 and 64 for 2-class and 8-class classifiers respectively. Since images are all gray-scaled, input tensor is in the shape of batchx1x52x52. A convolutional layer with 6 neurons, paired with a 2x2 maxpooling layer, takes in the input, and outputs a batchx6x24x24 tensor. Following are another convolutional layer with 16 neurons and 2x2 maxpooling layer, rendering a batchx16x10x10 output. Three fully connected layers in the end take in features and flatten them into 10 output. Loss function is cross entropy, so no need to convert class label to one-hot encoding. Activation function is ReLU. All layers have stride of 1 and no padding. To prevent over-fitting, L2 norm is used. Regularization method is dropout, which outperforms L2 penalty during experiments. The optimizer is SGD (Stochastic Gradient Descent), with learning rate at 0.001 and momentum at 0.9.

III. Experiments

A. 2-class classifier

We start by creating a basic CNN model with data augmentation. Originally, dataset is composed of 8-class data. First, the subset which only contains 'a' and 'b' characters is extracted. Total data number is 1600. Then we train and test our model. In order to do that, we split subset into test set and train set at a ratio 8:2. Test set is kept unknown from model until testing. Only resizing is performed at the input dataset. Fig.8 shows the loss curve of the 2-class classifier on a 50-epoch training (left), and the accuracy on random selected data from whole dataset. We can see from the accuracy curve,

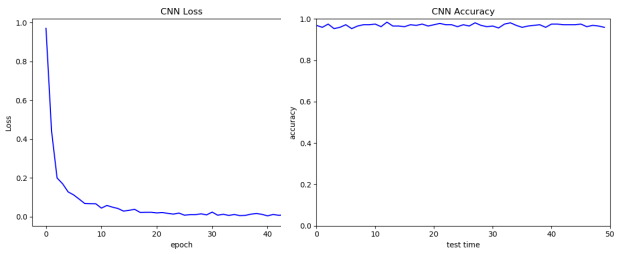

Fig. 8. For 2-class Classifier without Data Augmentation, Loss(left), Accuracy(right)

it remains stable around 95%. So, the result is reliable.

B. Data Augmentation

Then we implement data augmentation method and verify those images are randomly transformed. Results are shown in Fig.9. Now, we can train our model with data augmentation enabled, resulting in Fig.10.

It's interesting that with the subset, data augmentation didn't help with accuracy. Actually, it slightly increases the loss and

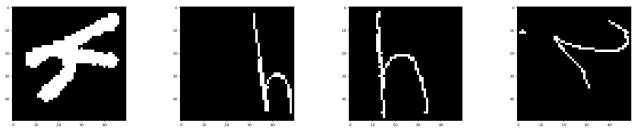

Fig. 9. Random Down-sample, Translation, Scaling, Rotation on Data

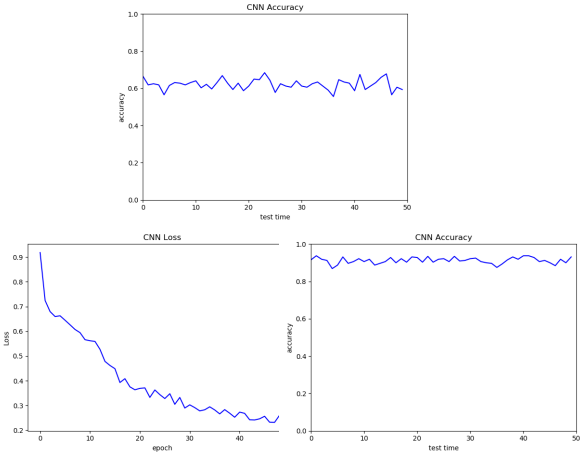

Fig. 10. For 2-class Classifier. Accuracy without Augmentation(upper); With Data Augmentation, Loss(left), Accuracy(right)

hamper the accuracy, possibly due to the size of the dataset is so small. When we apply transformation to test data and trying to find the problem, the result as in Fig.10

C. 8-class Classifier

Next, we compares how well the model performs on whole dataset and whether data augmentation can help. Total data number is 6400, and train set vs test set is 8:2. Fig.11 shows the loss curve of the 8-class classifier on a 50-epoch training. Then, we add data augmentation to it and set epoch number to 500, because it takes longe to converge. Result in Fig.12 As is shown above, augmentation still doesn't

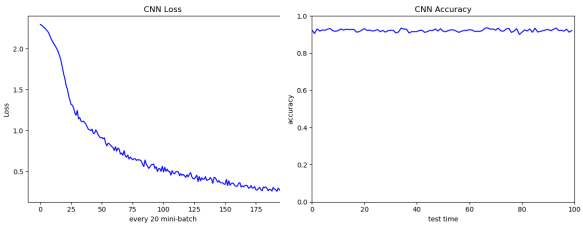

Fig. 11. For 8-class Classifier without Data Augmentation, Loss(left), Accuracy(right)

improve the accuracy of this model, even makes it worse. This is probably due to the simplicity of our CNN model, which only has two convolutional layers, and 6 and 16 neurons in them respectively. So, it's hard for this model to learn a complicated model that generalize over the augmented training data. Another possible reason is for the training epochs. Due to the time constraints, both of our models haven't converged completely.

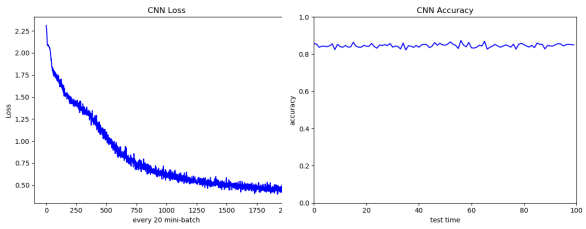

Fig. 12. For 8-class Classifier with Data Augmentation, Loss(left), Accuracy(right)

D. Unknown class

For the 9th-class, unknown, we set thresholding at the output layer. When all probabilities are less than the threshold, we decide this is a outlier for 8 known classes. Otherwise, the corresponding label will be the output.

Finally, we choose the model trained without data augmentation as our final design. After training for 1000 epochs, result is Fig.13. Loss is around 0.4, and accuracy is around 0.98.

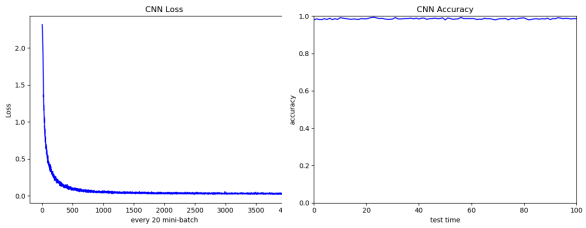

Fig. 13. For 9-class Classifier without Data Augmentation, Loss(left), Accuracy(right)

IV. Conclusion

Above discussion and experiments provides some insight of how a typical character recognition classifier is built. And experimental design are key guidance for this paper. For simple image classification such as 2-class classification, convolutional networks performs well without data augmentation. Even more, due to the simple architecture of a simple network, increasing data complicity can harm model accuracy. For a large image classification task, deeper and more complicated models outperforms simpler one. But it needs extra computing time to learn and converge. It's a trade-off between model complicity and accuracy in real-world applications.

Some extra works we have done for this paper is trying to introduce CapsNet, which is newly published model and outperforms most networks for less data samples. [4] Further study for this paper could be in that direction.

References

[1] A. Priya, S. Mishra, S. Raj, S. Mandal, and S. Datta, "Online and offline character recognition: A survey," in *2016 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2016, pp. 0967–0970.

[2] Y. LeCun, L. e. o. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] V. Jayasundara, S. Jayasekara, H. Jayasekara, J. Rajasegaran, S. Seneviratne, and R. Rodrigo, "Textcaps: Handwritten character recognition with very small datasets," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 254–262.

[5] D. Claudiu Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *arXiv preprint arXiv:1003.0358*, 2010.

[6] A. R. Dhamija, M. Gü nther, and T. E. Boult, "Reducing network agnostophobia," in *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. USA: Curran Associates Inc., 2018, pp. 9175–9186. [Online]. Available: http://dl.acm.org/citation.cfm?id=3327546.3327590

[7] A. Karpathy *et al.*, "Cs231n convolutional neural networks for visual recognition," *Neural networks*, vol. 1, 2016.

[8] T. K. Hazra, R. Sarkar, and A. Kumar, "Handwritten english character recognition using logistic regression and neural network."