Letter Recognition Using Convolutional Neural Networks

Mrityunjay Mishra
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712
mrityunjay2000.mishra@utexas.edu

Mihir Suvarna
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712
mihirsuvarna@utexas.edu

Daniel Sialm
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712
daniel.sialm@utexas.edu

Abstract

The task of letter recognition, a sub-task of text recognition, is growing in popularity due to emerging applications in robotics, computer vision, and much more. While improvement in this field has been made in the recent years, there is still room for more. Many of the previously developed models have been complex networks with multiple levels. This paper presents a simple, shallow convolutional neural network for the task of letter recogntion. This work also explores how to better regularize and increase the accuracy of the model while keeping it as simple as possible. It is found that even a simple, one-layer convolutional neural network can be very effective at letter recognition given that a good dataset is used for training the model.

1 Introduction

Text recognition, also known as optical character recognition (OCR), is the task of recognizing and extracting text from different media such as photos, documents, and letters. Emerging applications today have increased the demand for OCR software. For instance, scanners extract relevant texts from documents to relay back to the user, robots use cameras to read signs in their environment to better aid navigation, and mail sorters read handwritten envelopes to redirect mail to get to its destination.

Letter recognition, a sub-task of text recognition, is the task of recognizing the features of individual letters to extract them. It has been popularized due to the growing importance of text recognition and also because of upcoming applications that require precise character recognition, one character at a time. Consider modern smartphones, which convert handwritten characters of the user to text instantaneously. Such applications require efficient and effective letter recognition models, and while much improvement has been made in recent years, there remains room for improvement. Therefore, this paper aims to create a robust letter recognition model using a convolutional neural network (CNN).

2 Background

Many OCR software have been built in the past. Perhaps one of the most well known ones is Tesseract OCR, originally developed by HP and now owned and maintained by Google. Tesseract

OCR combines many techniques, such as line finding, baseline fitting, fixed pitch detection, and chopping to perform text recognition using an adaptive classifier[3]. Understanding the context a letter is in, such as different letters around it, will provide better recognition; when trying to identify the last letter in a sequence involving "goin-", the model will more likely predict the final letter to be a 'g' than a 'y'. The model proposed in this paper does not perform such complex analyses of words and simply focuses on one letter at a time to classify it. Additionally, the model in this paper does not look for specific features in each letter; rather, it learns the features relevant for classification on its own. This work also uses a CNN rather than an adaptive classifier.

Text recognition can also be conducted in natural scenes - a common example is detecting the text of a signpost. [5] explains many image processing techniques that can be used for text recognition in such scenarios, such as background removal and rectification. While this work uses some image processing techniques for better feature extraction from the dataset, the aim of this paper is not to do letter recognition in natural settings. Instead, work is only interested in classifying handwritten letters.

The work of Perwej and Chaturvedi [6] provides much inspiration for the model developed in this paper. Perwej and Chaturvedi developed a fully connected network with one hidden layer that took in digitized letters as input and then tried to classify them. They processed the image of handwritten letters first to digitize them into binary, pixel images so that their network could learn the features better. Thereafter, they fed the digitized image into their network for learning. This was an effective approach and inspired processing the acquired dataset into grayscale images; however, the input is not converted to a fully binary image. Additionally, this paper uses a CNN instead of a fully connected network.

The idea of using a CNN was inspired by Wojna et. al. [7]. Wojna et. al. used CNNs, RNNs, and a novel attention mechanism for the task of text recognition. Additionally, they showed that deeper networks were not always better for text recognition as very complicated, spatially invariant features (found in deeper networks) actually hurt performance and accuracy. Consequently, this paper was inspired to use a simple CNN for letter recognition. Using a CNN would decrease the number of parameters and speed up learning. Furthermore, convolving inputs would decrease over-fitting by entwining the different features of a letter together rather than learning individual pixel values. The model in this paper still differs significantly than the one used by Wojna et. al. and it is used for letter recognition specifically, not for the general task of text recognition.

3 Methods

3.1 Data collection

23 students from the University of Texas at Austin were asked to hand write the alphabet 10 times. Both upper and lower case letters were collected. About half of the students were asked to write the alphabet on paper while the other half were asked to write the alphabet on an iPad. This variation (the medium the data was collected on) was introduced to better represent the fact that tablets have become a common writing instrument today. For the model to be truly robust, it should be able to identify handwritten letters on both paper and a tablet.

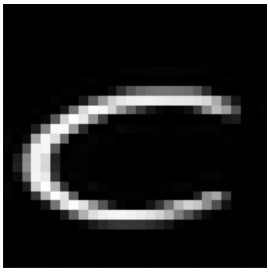

Figure 1: 28 x 28 image of a handwritten 'c'

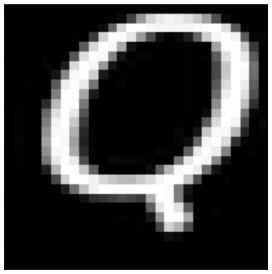

Figure 2: 28 x 28 image of a digital text 'Q

3.2 Data processing

The collected data was processed and converted to ttf files by using Calligraphr [1]. Calligraphr is an online platform that converts handwritten alphabets into custom fonts. It processes the input image (handwritten letters) through various image processing techniques (the specifics of which are not disclosed by Calligraphr) that converts the input letters into purely black and white ttf files. While not specifically disclosed, it can be seen that Calligraphr likely cleans the image using filtering, smoothing, rectification, and other techniques to produce clean ttf files. Such image processing is important in the task of letter recognition - it removes noise, thus allowing for better feature extraction which helps any model learn better. Because Calligraphr did this automatically, it was used for processing the collected data.

Once the ttf files were generated, they were converted into png images (one image per letter) and resized into 28 x 28 images with a black background and white letter (the colors in the original ttf are inverted). Individual pngs were generated because the purpose of the model in this paper is to classify one letter at a time, which cannot easily be done using a ttf file. The images were resized to be of dimensions 28 x 28 because this is the industry standard and also allows a network to learn faster (since the image is smaller). Finally, the colors of the image were inverted before being converted into torch Tensor of size 28 x 28. This inversion occurred so that areas where no pencil marks exist would be represented by a floating value of 0.0 while areas with the "most" pencil makrs would be represented by a floating value of 1.0.

3.3 Augmenting the data with EMNIST

The EMNIST [2] dataset from Torchvision [4] was used to augment the collected data. EMNIST is an extension of MNIST which contains handwritten letters and numbers. Only the letters from the EMNIST dataset were used. A subset of these letters were randomly picked and combined with the collected dataset to provide a more robust dataset that did not bias the learning of the model towards the handwriting of the few people from whom the data was collected. For a more in-depth explanation for such augmentation, refer to the discussion in the *Preliminary results* section.

3.4 Augmenting the data with digital fonts

The dataset was further augmented with digital fonts for computational experiment 1a. About one hundred fonts were used. Each font's ttf or otf file was processed using the same method the collected dataset was, producing png images of dimensions 28 x 28 with a black background and white letter. This super-augmented dataset was not used for any other experiment besides computational experiment 1a.

3.5 Building the convolutional neural network

The architecture of the CNN model built can be seen in Figure 1. The CNN was purposefully kept small (i.e. shallow) because [7] found that deeper networks were not always better for text recognition. The first layer of the model was an 8-channel convolutional layer with 5 x 5 kernels (no padding) with a step size of 1. The size of the kernel was chosen to be 5 x 5 since this size is quite common with 28 x 28 images.

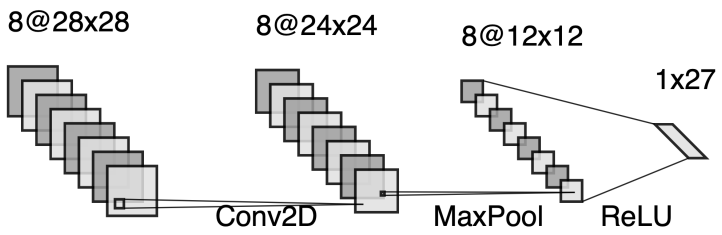

Figure 3: Architecture of the basic CNN used

The number of channels was chosen arbitrarily - it was reasonably assumed that 8 channels would keep the network simple enough while also capturing a reasonable amount of complexity and features from the data. The second layer was a max pool layer with size 2 x 2 kernels. Max pool was used because it is proven to be useful in CNNs to prevent over-fitting as well as decrease computational cost. A kernel size of 2 x 2 was used to prevent loss of too much information (the images in question were small to start with so it would be ideal to preserve as much as possible to better classify the letters). The resulting feature maps were flattened and fed through a ReLU nonlinearity in the final two layers to give probabilities of being to the class of each letter. Note that the final step gives 27 probabilities - this was to account for the *N/A* class if the image in question was predicted to not be a letter at all. It is also important to note that both upper and lower case letters should be classified under the same class.

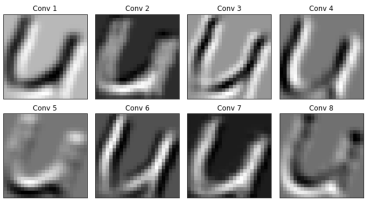

Figure 4: Conv. Layers Output

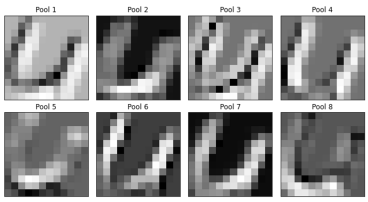

Figure 5: Pooling Layers Output

4 Experimental results

4.1 Preliminary results with the CNN

This CNN performed quite poorly when run on just the collected dataset of 23 people's handwriting: the model yielded an accuracy of around 50%. However, such poor performance is not surprising. The model was not able to train well because it over-fit to the handwriting of the few people it received as input and became heavily biased since each person wrote the alphabet ten times. This can be verified through the increasing validation loss despite decreasing training loss. Consequently, the model did not generalize well and gave poor results when run on the test set (comprised only of the collected dataset). It should be noted that there was no data leakage between the training, validation, and test sets. While an accuracy of around 50% is not the worst (in fact, is is about twelve times better than randomly guessing), it can clearly be increased if a better dataset with more letter variety is provided.

Table 1: Performance through Different Datasets

Augmenting the Dataset	
Accuracy w/o Augmented Dataset	Accuracy w/ Augmented Dataset
53.85%	84.03%

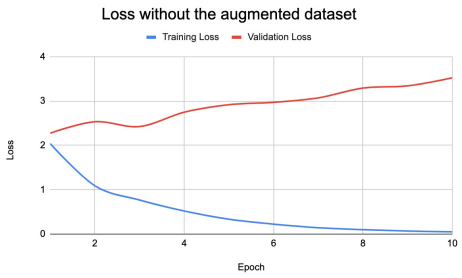

Figure 6: Loss over Nepochs – Original

Figure 7: Loss over Nepochs – Augmented

Therefore, the model was then trained using the augmented dataset that included data from EMNIST. Note that the training, validation, and test sets were all augmented with random data from EMNIST. There was an immediate improvement: the new accuracy increased to about 85%. This clearly shows the importance of carefully constructing the dataset: because EMNIST harbors handwriting from many different individuals (thus being more representative of the variety of handwriting that are possible), it allowed the CNN to learn and generalize well. This also shows that the simple CNN constructed is actually quite good at recognizing handwritten letters.

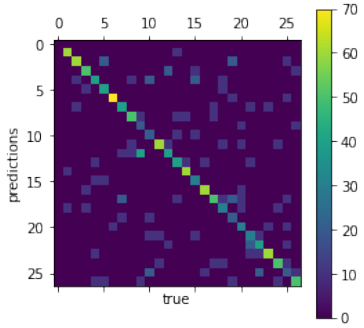

Figure 8: Contingency Matrix – Original

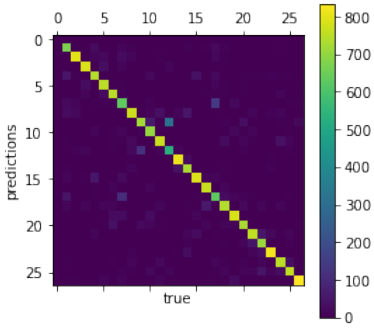

Figure 9: Contingency Matrix – Augmented

4.2 Computational experiments

To further test and improve upon the model, two computational experiments were conducted. The first tries to better regularize the CNN. The second tries to improve the accuracy and the regularization of the CNN by changing the architecture of the CNN.

Table 2: Performance through Regularization

Regularization	
Method	Accuracy
Data Augmentation	78.54%
Dropout	82.15%
Avg. Pooling	82.79%
Early Stopping	84.53%

4.2.1 Experiment 1a: regularization through further data augmentation

It was hypothesized that augmenting the dataset further with pictures of digital fonts would further regularize the model. 138 different fonts were used to generate 28 x 28 png images (with a black bacground and white letter) of each letter. This new set of data was used to augment the current dataset. Each of the training, validation, and test sets were augmented using different fonts to avoid data leakage. The accuracy of the newly trained model did not improve and in fact decreased by about 2%. This may be attributed to the fact that only a small number of fonts were used. There are more than 36000 fonts available and only a few were used to augment the dataset. It is believed that using a larger variety of fonts will better regularize the network and improve accuracy. Furthermore, digital fonts have very distinct features such as serifs and corner rounding that do not necessarily exist with handwritten letters.

4.2.2 Experiment 1b: regularization through dropout

Adding a dropout layer before the fully connected layer is a common strategy to better regularize a network and reduce the over-fitting problem. However, it was hypothesized that using dropout would actually decrease the performance of the model in this work because the input images were quite small to begin with. Dropout would take away key information that the network would need

to accurately classify the letters. The hypothesis did come to be true and the accuracy of the newly trained network dropped by about 2%.

4.2.3 Experiment 1c: regularization through average pooling

Average pooling is also another common strategy that can be used instead of max pooling. This is because the convolved kernels from average pooling are usually smoother compared to the convolved kernels of a max pooling layer which can be more effective in many situations. However, because the input image is quite small, it was hypothesized that using average pooling would not impact the accuracy of the model significantly. This was indeed the case: the accuracy effectively stayed the same. Looking at the convolved kernels shows that the kernels from a CNN with an average pooling layer are a little smoother. However, they are not too dissimilar than the ones from the CNN with max pooling, which explains why using average pooling did not cause a major impact.

4.2.4 Experiment 1d: regularization through early stopping

Early stopping is a common strategy for regularizing networks. The base model was trained across 10 epochs. To test early stopping, the model was trained across 50 epochs and would stop training once validation loss did not decrease significantly over a few epochs. It was hypothesized that early stopping would improve accuracy marginally. However, accuracy did not seems to improve by much. This is likely because the early stopping network finished training after just 12 epochs which is not very different from the 10 epochs used to train all other models.

4.2.5 Experiment 2a: double convolutional layer CNN

Visualizing the learned kernels from basic model (a single layer convolutional network) shows that the kernels seem to be learning the different types of edges that letters have. For example, Kernel seems to be learning vertical edges while Kernel seems to be learning horizontal edges. This suggests that adding a second convolutional layer could help the network learn more precise features per letter and increase the overall accuracy. The double convolutional layer network was built on top of the single convolutional layer network; it added a second convolution layer, going from 8 to 64 channels with a 5 x 5 kernel size and padding 1, followed by a max pool layer and another ReLU activation, before finally going through a flatten and linear layers in the end. The number of channels were increased by a factor of 8 so that the network could learn considerably more features in the second convolution layer and to stay consistent with the first convolutional layer.

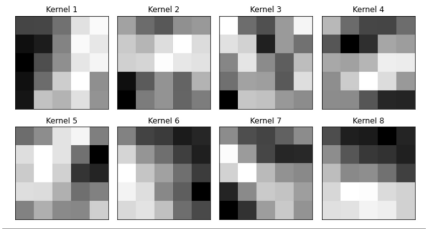

Figure 10: First Conv. Layer Kernels

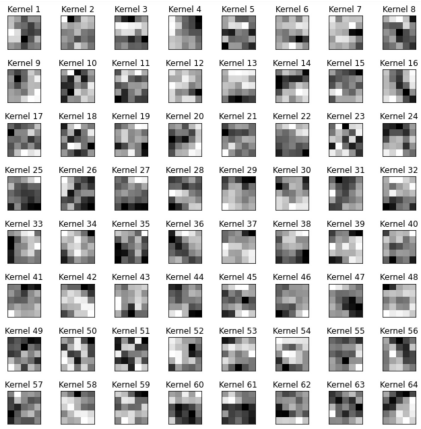

Figure 11: Second Conv. Layer Kernels

It was hypothesized that overall accuracy of the new network will increase, and it indeed increased by about 5% to get a total accuracy of 89.92%. Visualizing the kernels from the second convolution layer shows that the network is learning more precise features. It seems that it may be learning more precise edges per letter, but that is much harder to tell. Nonetheless, this shows that making the network a bit deeper does improve the accuracy, as has been the case in previous networks (for other tasks) as well.

4.2.6 Experiment 2b: changing the kernel size

To better learn how the base model was learning, an optimized input model (built on top of the base CNN) was built. The optimized input model took in an input of all zeros and then optimized this input to reduce the loss (or error) to a specific letter (the letter 'a', for example). The following optimized inputs were produced.

It can be seen that each output has a different shape, meaning that the network is definitely identifying different features to differentiate different letters. What the network is learning precisely, however, is harder to tell. However, it can be seen that the inputs are not very smooth - i.e. the features learned by the model are not smooth. This can be verified by looking at the learned kernels of the single layer CNN and also by looking at its convolved images (which are not very smooth either). It is common in image processing to smooth images to produce better results. Thus, the kernel size is altered to see the impact on the overall accuracy. It is hypothesized that increasing the kernel size will increase accuracy and vice versa.

Table 3: Performance with Changing Kernel Size

Changing Kernel Size	
Kernel Size	Accuracy
3 x 3	81.65%
5 x 5	84.03%
8 x 8	84.74%
11 x 11	84.78%

Increasing the kernel size did improve accuracy, but only marginally. However, accuracy decreased with a reduced kernel size, as hypothesized. The produced optimized inputs with bigger kernels are also smoother, indicating that the network is learning more smoothed out features. This demonstrates that a smoother representation of the features helps the model better learn and classify letters.

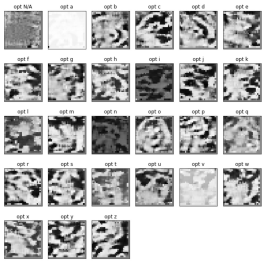

Figure 12: 3x3 Kernel Optimized Inputs

Figure 14: 5x5 Kernel Optimized Inputs

Figure 13: 8x8 Kernel Optimized Inputs

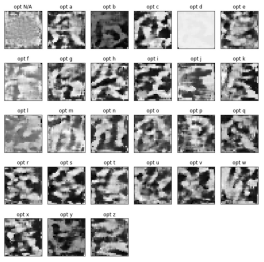

Figure 15: 11x11 Kernel Optimized Inputs

5 Conclusion

This work built a single layer convolutional network for the task of letter recognition, which is growing in importance due to a variety of emerging applications. It was shown that a simple, single-layer CNN was quite good at classifying handwritten letters and even provided a good amount of

regularization and generalization. Still, making the network a little deeper (i.e. adding a second convolution layer) does improve the accuracy as it allows the network to learn more precise features. It was further demonstrated that if the network was able to learn more smoothed out features (i.e. use a larger kernel size), then it was able to learn features better and classify the letter with a higher accuracy. However, it was shown that no matter how good the architecture of the model, the model will not perform well if it was not fed a good, representative, and vast dataset.

In the future, it would be interesting to to see how this model (or a model similar to the model presented in this paper) would perform when there are natural deformations in the input images (e.g. the image is taken from different angles). It would also be interesting to see how the network would perform if noise was added or different levels of intensities were introduced. The network presented in this paper fared quite well in the task of letter recognition, but that does not mean that the best results have been achieved. There are many more opportunities for improvement, and it will be interesting to see how future models evolve to classify letters (and even other characters) with increasing levels of accuracy.

6 Roles and Contributions

While all three members were involved in all the processes (from data collection to writing this paper), each member inevitably ended up focusing on one step of the process a little more than the other. These areas of focus were as follows.

- Mrityunjay Mishra – Data collection, writing the paper.
- Mihir Suvarna – Data pre-processing and cleaning, writing the paper.
- Daniel Sialm – Maintaining the notebook, writing the paper.

References

[1] *Calligraphr*. 2022. URL: `https://www.calligraphr.com/en/`.

[2] J. Tapson et. al. G. Cohen S. Afshar. "EMNIST: an extension of MNIST to handwritten letters". In: The MARCS Institute for Brain, Behaviour and Development, Western Sydney University. Mar. 2017.

[3] R. Smith. "An Overview of the Tesseract OCR Engine". In: Google Inc.

[4] *Torchvision*. 2022. URL: `https://pytorch.org/vision/stable/index.html`.

[5] Y. Zhu et. al. X. Chen L. Jin. "Text Recognition in the Wild: A Survey". In: College of Electronic and Information Engineering, South China University of Technology.

[6] A. Chaturvedi Y. Perwej. "Neural Networks for Handwritten English Alphabet Recognition". In: *International Journal of Computer Applications*. Vol. 20. Apr. 2011.

[7] D. Leet et. al. Z. Wojna A. Gorbant. "Attention-based Extraction of Structured Information from Street View Imagery". In: University College London and Google Inc.