Convolution Neural Network to Solve Letter Recognition Problem

Amelia Amelia

Email: u5431566@anu.edu.au

Research School of Computer Science
Australian National University
108 North Rd, Acton, ACT 2601, Australia

Abstract

In this paper, we extend the concept of Shared Weights to Convolutional Neural Network in classifying the 26 letters in English alphabets. We found that the CNN model performed significantly better than the ordinary network which uses Shared Weights concept. The CNN model developed in this paper achieved 92% of accuracy while the Shared Weights concept only managed to achieve 74.29% of accuracy. The paper also experimented on the effect of number of output channels and kernels in the CNN model. We came to the conclusion that the model performed better once we managed to find the optimum number of kernels. We also concluded that although higher number of output channels results better performance, further consideration is needed in order to decide if slight improvement in performance is worth the computation cost. The paper also drew comparisons with other papers which used the same EMNIST data set and discussed the difference in models built and their performance gap which mainly caused by the difference in structure of the network and technique used.

Keywords:
Weight sharing, letter recognition, two-layer network, convolutional neural network

1. Introduction

Letter recognition using a neural network is one of the most widely experimented topics in Computer Science. Many research papers acknowledge the vast interest researchers have in letter recognition (Guyon, et al., 1991). Modelling in a neural network also helps researchers to obtain new knowledge about design principles for letter recognition which is important for future research (Fukushima, 1992).

The amount of interest in this topic motivated us to build a two-layer neural network to classify black-and-white rectangular pixel images as one of the 26 letters in the English alphabets. In our previous paper, we contributed by discussing and comparing two different methods in solving a letter recognition problem. The first method used is a simple two-layer neural network. The second method used is the Shared Weight concept introduced by Gedeon (1998). In previous experiment, we concluded that Shared Weight does not necessarily performs better than Two-Layer Network as one of the consequences of the method is that it is hard to find the right compression-decompression function.

In this paper, we tried to extend the concept of Shared Weight (Gedeon, 1998) to Deep Learning by building a simple Convolutional Neural Network to classify the 26 letters in the English alphabets. Convolutional Neural Network (CNN) has shown much success in achieving translation invariance for image processing tasks. The success is largely related to the weights sharing, local connectivity and max-pooling in the CNN architecture which help to reduce the number of parameters in the whole system which results in a more efficient computation in the classification process (Abdel-Hamid, et al., 2012). This paper will discuss on how to implement a simple CNN on an image data set of English alphabets and the result of the implementation. The paper will further discuss about the result of experimenting with the number of kernels and channels in the network. Lastly, it will draw comparison between the experiment we conducted and other experiments which used the same data set as us.

2. Methods

2.1. Data set

In the previous experiment, we used the Letter Recognition data set available at UC Irvine Machine Learning Repository (Slate, 1991). The data set consists of 20,000 black-and-white pixel images of the 26 letters in the English alphabets. The distinct stimuli of the images were converted into 16 primitive numerical attributes which were written in the form of integers.

This time, we are using a real-life image data set of handwritten letters. This data set is called EMNIST data set (Cohen, et al., 2017). It is an extension of the MNIST data set which is widely used for image classification problem. EMNIST data set contains more complicated data such has handwritten English alphabets (used in this paper) and a combination of both alphabets and numbers whereas MNIST only has numerical data which is only suitable for digit classification (10 classes). The purpose of the model that we built is to classify the images in the EMNIST data set into the correct letter, hence classifying the images into 26 different categories

2.2. Activation Function

In our previous work, we used the Shared Weight concept introduced by Gedeon (1998). This method was expected to have better results compared to the usual two-layer neural network as by using the Shared Weight concept, we would be able to reduce the space between network weight configurations. A Shared Weight Network works by making the weights from input layer to hidden layer and the weights from hidden layer to output layer to be identical. As a consequence of the identical weights, in theory, the compression function then must be directly invertible. This means that the network should perform better than standard networks as the network is able to find 'the inverse function' instead of producing the approximate of 'the inverse function'. In our experiment, we implemented the Shared Weight concept using Rectified Linear Units (ReLu) activation function. ReLu Activation Function has been quite widely used in recent deep learning networks. Research has shown that ReLu Activation Function is suitable to train large networks as it performs faster training and allows for better generalization (Zeiler, et al., 2013).

In this experiment, we still used ReLu as the activation function in our CNN as well as Log Softmax Activation Function which was used in our two-layer neural network. Softmax Activation Function is standard for 1 of K classification problems (Bishop, 1995). It is ideal for multiclass classification problems as the function ensures that every output produced by the network is all between zero to one and that they all sum to one on every timestep which means the function produces probabilities of the observed object being a certain object at a given frame (Graves & Schmidhuber, 2005).

2.3. Structure of the CNN

In order to reduce under-fitting of the network and to make sure that the CNN we built can increase the network's feature learnability while keeping the CNN as simple as possible, we built a CNN with two convolution layers followed by a dropout and then two fully connected layers. The reason we used dropout is to reduce overfitting. Dropout is a technique which randomly drop units along with their connections from neural network during training, hence, prevents units from adapting too much to the data set. This results in better regularization and improved performance on supervised learning (Srivastava, et al., 2014). We also used Max-pooling, which is a form of non-linear down sampling in the Pooling Layer in order to reduce overfitting and reduce the number of parameters and computation in the network.

2.4. Number of Channels and Kernels

The output channels represent the number of features we want to identify. We experimented by using 10, 20, 30, 40 and 50 output channels for the first convolution layer and 20, 40, 60, 80 and 100 output channels for the second convolution layer. The purpose of this experiment is to investigate whether learning as many features as possible always improves the performance of the network.

Kernel is a set of shared weights and biases. We experimented with the kernel size to investigate whether the more details we catch (the smaller the kernel size is), the better the performance of the network will be. We experimented with kernel size of 3, 5 and 7.

3. Results and Discussion

3.1. Experimenting with output channels and kernel

Below is a table of results of a CNN with 10 output channels in Layer 1 and 20 output channels in Layer 2 with different kernel sizes (Table 1).

Table 1

Kernel size	Average Loss	Accuracy
3	0.4073	87%
5	0.4004	87%
7	0.5009	84%

As we can see from the table above, the accuracy improves a little bit when we reduce the size of the kernels to 5 from 7. The results indicate that the smaller the kernel size which means the more details we try to capture, the better the performance of the network is. However, the improvement will stop once it reaches the maximum point and after that, using smaller kernel size will not be useful as we can see from the performance result of the CNN network with kernel size 3 which is roughly the same as the performance of the network with kernel size 5. From this experiment, it seems that we can conclude the optimum size of kernel in this case is 5.

Below is a table of results of a CNN with kernel size of 5 and different output channels (Table 2).

Table 2

Experiment No.	No. of output channels in Layer 1	No. of output channels in Layer 2	Average Loss	Accuracy
1	10	20	0.4004	87%
2	20	40	0.2755	91%
3	30	60	0.2513	91%
4	40	80	0.2437	92%
5	50	100	0.2289	92%

The results indicate that the more features we set the network to learn, the better the performance of the network is. However, as we can see from the table above, while there is quite significant improvement from Experiment 1 to Experiment 2 as the average loss decreases by almost 50% and the accuracy increases to 91%, there is not much improvement from Experiment 2 to Experiment 5. We may conclude that considering the computation cost,

increasing the number of output channels for a very small improvement such as above may not be worthwhile.

3.2. Results comparison with previous work

In our previous work, when implementing the Shared Weight concept, our network's testing accuracy was 74.29%, just slightly higher than a regular Two-Layer Network's accuracy of 71.19%. This result made us to conclude that Shared Weights does not necessarily improve the network's performance. However, by extending the Shared Weight concept to Convolutional Neural Network, the performance of the network is significantly better as the accuracy ranges between 87% to 92%. This significant improvement in performance may be caused by the structure of the CNN which allows the network to learn more efficiently about the distinct features of each alphabet. The network benefits from things such as Max-pooling and Dropout which help to reduce overfitting or underfitting problems while reducing the number of free parameters and computation cost. Whereas, the success of the Shared Weight concept in the previous work depends on finding the right compression-decompression function which proved to be difficult.

3.3. Results comparison with other papers

There are other papers that use deep learning and the EMNIST data set in order to classify the 26 letters in the English alphabets. Meany & Arola (2018) used a three-layer CNN which feeds into an LSTM layer. Just like our model, the model they developed also used Max-pooling and Dropout and achieved 77% accuracy whereas out model achieved 92% accuracy. This is unexpected because the model they developed should have performed better considering the depth of their model and the additional LSTM layer. This could be caused by them trying to classify word images instead of characters.

Other than building a network with more depth, another way to improve the performance of a Deep Convolutional Neural Network (DCNN) is by implementing CUDA to reduce computation time an achieve high accuracy (Singh, et al., 2017). By implementing CUDA, Singh, et al. (2017) managed to achieve 99.62% of accuracy with the computation time of 8 minutes and 56 seconds using 5 layered DCNN on GPU with 30000 iterations. This implies that in order to significantly improve our model, it seems that we need to implement new technique such as the one used by Singh, et al. (2017).

4. Conclusion and Future Work

In conclusion, we found that extending the concept of Shared Weights to Convolutional Neural Network improve the performance of the network significantly. This could be caused by the network benefitting from the structure of the CNN such as Max-pooling and Dropout which help to reduce overfitting or underfitting problems and reduce the number of free parameters and computation cost. Further experiment to finetune the CNN found that an optimum kernel size will improve the performance of the network. We also found that the larger the number of output channels which means the more features we set the network to learn, the better the performance of the network will be. However, we may want to consider if the slight increase in performance is worth the extra computation cost.

For future work, we may want to observe other parameters that can impact the performance of CNN. An experiment on these parameters, such as the stride and padding of the network may allow us to further finetune our model.

5. References

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H. & Penn, G., 2012. *Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition.* Kyoto, IEEE.

Bishop, C., 1995. *Neural networks for pattern recognition.* New York: Oxford University Press, Inc.

Cohen, G., Afshar, S., Tapson, J. & van Schaik, A., 2017. *EMNIST: an extension of MNIST to handwritten letters.* [Online]
Available at: https://arxiv.org/abs/1702.05373
[Accessed 20 May 2018].

Fukushima, K., 1992. Character recognition with neural networks. *Neurocomputing,* 4(5), pp. 221-233.

Gedeon, T. D., 1998. *Stochastic bidirectional training.* San Diego, IEEE.

Graves, A. & Schmidhuber, J., 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks,* 18(5-6), pp. 602-610.

Guyon, I. et al., 1991. Design of a neural network character recognizer for a touch terminal. *Pattern Recognition,* 24(2), pp. 105-119.

Klami, A., 2003. *Regularized Discriminative Clustering,* Helsinki: Helsinki University of Technology.

Meany, C. & Arola, M., 2018. *Optical Character Recognition via Deep Learning.* [Online]
Available at: http://cs230.stanford.edu/files_winter_2018/projects/6910235.pdf
[Accessed 29 May 2018].

Peltonen, J., Klami, A. & Kaski, S., 2004. Improved learning of Riemannian metrics for exploratory analysis. *Neural Networks,* 17(8-9), pp. 1087-1100.

Singh, S., Paul, A. & Arun, M., 2017. *Parallelization of digit recognition system using Deep Convolutional Neural Network on CUDA.* Chennai, IEEE.

Slate, D. J., 1991. *UCI Machine Learning Repository: Letter Recognition Data Set.* [Online]
Available at: http://archive.ics.uci.edu/ml/datasets/Letter+Recognition
[Accessed 16 April 2018].

Srivastava, N. et al., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research,* 15(1), pp. 1929-1958.

Zeiler, M. D. et al., 2013. *On rectified linear units for speech processing.* Vancouver, IEEE.