Deep Dropout Artificial Neural Networks For Recognising Digits And Characters in Natural Images

Erik Barrow[1] and Chrisina Jayne[1] and Mark Eastwood[1]

Coventry University, Coventry, West Midlands, UK

Abstract. Recognising images using computers is a traditionally hard problem in computing, and one that becomes particularly difficult when these images are from the real world due to the large variations in them. This paper investigates the problem of recognising digits and characters in natural images using a deep neural network approach. The experiments explore the utilisation of a recently introduced dropout method which reduces overfitting. A number of different configuration networks are trained. It is found that the majority of networks give better accuracy when trained using the dropout method. This indicates that dropout is an effective method to improve training of deep neural networks on the application of recognising natural images of digits and characters.

Keywords: Character Recognition, Natural Images, Artificial Neural Network, Deep Learning, Dropout Network

1 Introduction

This paper investigates the application of Deep Neural Networks approach [1] and the Dropout Neural Networks method [2] to recognise and classify both digits and characters from a set of natural images [3].

Character recognition is a difficult problem to solve, as many characters can be similar to each other. With the addition of natural images the problem becomes harder, as many variances in the text can occur, such as font, colour, backgrounds, lighting, angle, and texture [3].

Neural networks have been previously applied to the MNIST dataset [4] for recognition of handwritten digits using a Deep Neural Network [4], producing an 0.35% error rate. This is a low error rate, however all the images in the dataset have been pre-processed.

Deep Neural Networks [1] have been shown to be an effective machine learning technique allowing for more complex features to be learnt. Deep Neural Networks refer to Neural Network Architectures with a large number of hidden layers [1]. The breakthrough paper published in 2006 by Hinton et al. [1] introduced an unsupervised fast, greedy learning algorithm that finds a fairly good set of parameters quickly in deep networks with millions of parameters and many hidden layers. This discovery enabled the pre-training of deep supervised multi-layer neural networks using the Restricted Boltzmann Machine

(RBM) generative model for each layer [1], [5]. The pre-training serves as an initialization of the neural network which is then fine-tuned with respect to a supervised criteria as usual. The unsupervised pre-training helps with initialization of the net into a more favorable region of the weight space [6] compared to the random initialization. Several studies have show that this leads to better generalization results [6], [7], [8].

More recently Neural Networks utilising the so called dropout method [2] have been shown to provide a better accuracy in classification problems, by reducing overfitting of the data and preventing a network becoming stuck in a local minima. Dropout is a method applied to neural networks that involves dropping neurons during training to cause other neurons to learn new features and prevent overfitting. Dropout based learning has previously been used successfully on the ImageNet dataset [9] of natural images. In this previous application [10] a Deep Convolutional Neural Network was used with dropout, and gained 37% error on identifying the object in the image, and 17% error in identifying the image within the top 5 predictions.

The purpose of this paper is to investigate the application of Deep Neural Networks with Dropout on the dataset of natural characters and to find out whether it can help solve the overfitting problem in this context. The results of the paper provide insight into the feasibility of using the dropout method for this problem, and provide recommendations for its practical application.

Motivation for this work comes from the various applications of deep neural networks and the dropout method to data sets of natural images [10] and handwritten digits [11], as well as the need for ways to improve accuracy of reading text in natural images.

This paper is organised into 5 sections. Section 1 introduces the paper, Section 2 describes deep neural networks and the dropout method, and Section 3 covers the data Set and pre-processing used. Experiments and Results are presented in Section 4, and Section 5 Concludes.

2 Methodology

The methodology consists of Restricted Boltzmann Machines for pre-training of the network weights, and a Deep Neural Network trained with the dropout method for the classifier.

2.1 Restricted Boltzmann Machines

The first part of the neural network for this problem is a Restricted Boltzmann Machine (RBM). The RBM is used to pre-train the weights of a neural network to provide better results during later training of deep architectures [1].

A boltzmann machine is a system of random variables $\mathbf{v}, \mathbf{h}$ whose joint probability can be described by an energy function as follows:

$$P(\mathbf{v}, \mathbf{h}) = \frac{\exp(-E(\mathbf{v}, \mathbf{h}))}{Z} \tag{1}$$

where Z is a normalisation factor

$$Z = \sum_{\mathbf{h}} \sum_{\mathbf{v}} \exp(-E(\mathbf{v}, \mathbf{h}))$$

.

The RBM used here has an energy function:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i v_i b_i - \sum_j h_j c_j - \sum_{ij} v_i W_{ij} h_j \tag{2}$$

RBM is trained by forward propagating the input from one layer to the next. The output is then passed back to the layer that gave the inputs, essentially forward propagating in the reverse direction. The returned results represent a reconstruction of what the network thinks was the input into the system. This altered predicted input is then fed forward again and compared to the original inputs results to produce an error value [12].

The RBM minimises the energy function via approximate gradient descent. We use the enhanced gradient presented in [13]. This training is done one layer at a time. So the first layer will have all the input data run though it over a number of epochs and will then move onto training the next layer, using the outputs of the previous trained layer as the input.

For the energy function given in equation 2, the weight and bias gradients have the form:

$$\nabla_e w_{ij} = cov_d(v_i, h_j) - cov_m(v_i, h_j)$$

$$\nabla_e b_i = < v_i >_d - < v_i >_m - \sum_i < h_j >_{dm} \nabla_e w_{ij}$$

$$\nabla_e c_j = < h_j >_d - < h_j >_m - \sum_i < v_i >_{dm} \nabla_e w_{ij}$$

where $< \cdot >_d$, $< \cdot >_m$ denote averages over the data/model distributions respectively, and $< \cdot >_{dm} = \frac{1}{2} < \cdot >_d + \frac{1}{2} < \cdot >_m$. These values are multiplied by the learning rate λ to provide the parameter updates.

After training of the stacked RBMs, the weights are then used to initialise the Deep Neural Networks that will be used to further train the weights in a supervised manner.

2.2 Deep Neural Networks

A Deep Neural Network is a network that has many hidden layers (typically more than one hidden layer) which utilises initial weights pre-trained with the RBM method. To train the Deep neural network the traditional backpropagation algorithm is used [14]. The inputs are multiplied by their associated weight matrix and then summed before being put through a sigmoid function. The result of the sigmoid activation function is then the output for our neuron that can be passed to the next layer as an input for the following layer [15].

When the input has forward propagated as far as the output layer the results of the prediction are compared against a list of targets for that input. The difference between the target and the actual input becomes an error value which is used to tweak the weights for that neuron. The error of that neuron is then passed back through back propagation as a factor for the neurons on the previous layer to take into account and update its own weights based on the error of all the neurons it was attached to.

This training is performed over a number of inputs and then repeated over a large number of epochs, stopping when the minimum has been found, or a maximum amount of epochs has been reached where more training would provide little improvement.

2.3 Dropout Method

Dropout is a relatively new method used in training neural networks and deep learning [2] [10]. The idea of dropout is that we randomly select a number of neurons on each layer (excluding the output layer) to be switched off for one epoch. At the end of that epoch the neurons and their associated weights are switched back on, and another set of neurons are selected to be switched off for the next epoch.

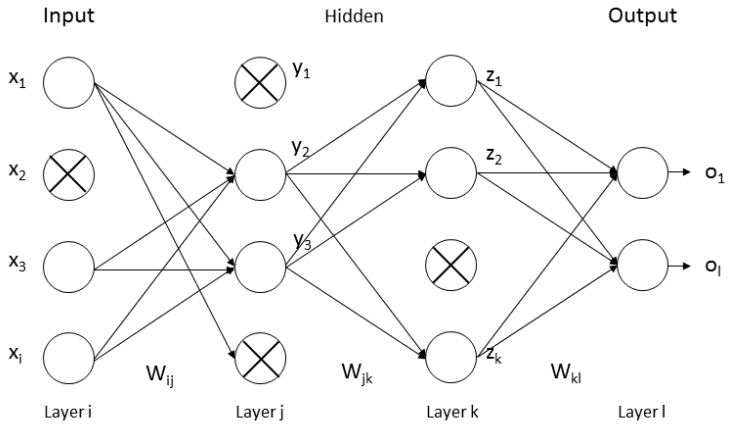

Fig. 1. A Network using Dropout during training

The idea of this is that turning off neurons that have been trained to recognise a certain feature will force a neuron that has learnt nothing or is slow in learning to speed up and try and replace the neurons that have been taken away. Doing this a number of times over a large amount of epochs helps to prevent over fitting of the data and getting stuck in a local minimum.

In a practical application, existing code can be edited to incorporate the dropout method. One of the simplest ways to do this is to select the neurons randomly at the beginning of each epoch, and then after taking a backup of all the weights set the weights associated with the off neurons to zero. This prevents the neurons that are turned off from having an impact on the training and also prevents them from being trained. The advantage of doing dropout this way is that you do not need to re-size and reshape the matrix to accommodate the change in neurons, although not reducing the size of weight matrix could effect speed on a large data set.

It is also necessary to modify the inputs of each node after training for the final prediction by multiplying the input by the proportion of nodes retained during training, so that the input of each node is similar to that of a network with a percentage of nodes switched off. If $\mathbf{x}^{(l)}$ are the node outputs at layer l, and p_l is the proportion of nodes retained during dropout for layer l, the node output during prediction is:

$$\mathbf{x}^{(l)} = sigmoid(p_l \mathbf{W}^{(l)} \mathbf{x}^{(l)} + \mathbf{b})$$

.

3 Data Set description

The Chars74K image data set [3] is used to conduct experiments on the effectiveness of the dropout deep neural networks. The dataset consists of 74 thousand images in 64 classes (0-9, A-Z, a-z), with images obtained from three sources (natural images, hand drawn images, and synthesised characters). For the purpose of this application we use only the natural image part of this dataset, which provides an accurate simulation of the application of deep neural networks in a real world environment.

Fig. 2. A sample of images from the data set

The dataset in its original state is unsuitable for the application of a neural network due to the varying sizes of images and the wide variety of colours that

introduce complexity into the recognition process as well as more features to be trained upon.

To prepare the dataset all the images were processed by first Gray scaling the image, and then resizing the image on its largest axis to 50 pixels. The smaller axis was then padded with pixels to match the average colour of the border pixels on one side to bring the image up to 50x50 pixels. This provides 2500 features to train our network upon.

The final step of preparation was to flip the colours of the Grey scale images so that the backgrounds were black and the number white. On a small number of instances the processed characters were too similar to the background and either became blanked out entirely in some cases, or in other cases where the background and text are the same colour the characters were not switched to white, as can be seen in Figure 3.

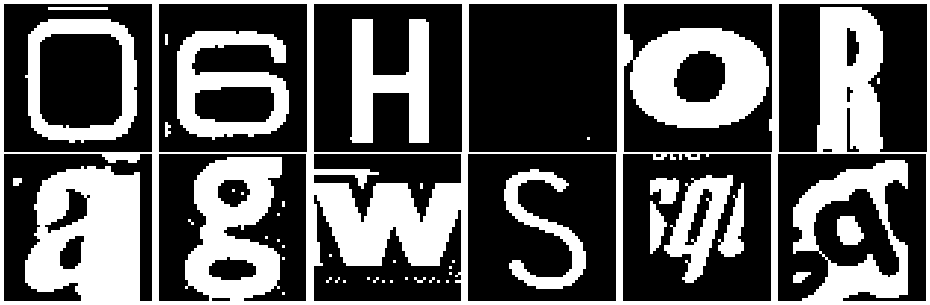

Fig. 3. A sample of images after processing

These variations in the processed dataset were minimal to the correctly processed images and should provide minimal impact to the training of the network as it is.

The dataset was split into three sub sets. A training set of 5705, a validation set of 1000 for use during training, and a test set of 1000 for testing after the training had finished.

4 Application And Results

The architecture of the Deep Neural Network used in our application is an input layer of 2500 neurons, at least 3 hidden layers, and an output layer of 62 neurons for all the classes. The network is initialised with weights produced by training Restricted Boltzmann machines on the hidden layers, leaving the output layer weights as randomly initialised.

Both Deep Neural Networks with and without the use of dropout were trained with the same architecture to provide a benchmark for the dropout method and prevent any bias.

The RBMs are trained over 10,000 iterations on each layer (stack), to initialise the weights for use with the Deep Neural Networks with and without dropout method.

Table 1. Test Results with 10,000 Iterations

Architecture	Error With Dropout	Error Without Dropout
1000,500,500	45.7	46.9
1500,1000,500	46.4	47.3
1500,1000,500,250	46.6	49.1
500,500,500	45.4	45.3
500,500,1000	43.7	45.8
1500,1000,500,1000	46.6	46.0
500,500,200	45.7	48.3
800,500,500,200	45.4	47.6
500,500,500,500	42.9	44.6
800,500,500,500,250	44.1	46.8

A number of different configurations can be seen in results Table 1. These same configurations where used to create the 10 different networks. Each configuration has 1 network with and 1 network without dropout, both being initialised with the same weights. The dropout networks all used the same drop limit of a maximum of 20% on the input layer and 30% on the hidden layers.

Table 1 shows that 8 out of 10 dropout based networks give better results over a cap of 10,000 iterations, with an average of 1.9875% increase in accuracy over the best 8, and a 1.52% increase over all 10. Where the results are worse for the networks with the dropout the difference is no more than 0.6 percent. The best trained dropout network gained an error of 42.9% and the best network without only achieved 44.6%, which is a increase of 1.7% in accuracy.

Occasionally dropout networks can benefit from extra training as it can take up to double the iterations of a non dropout network to converge [10]. Because of this we gave the dropout networks an extra 5,000 iterations to run. Table 2 shows the results after these extra iterations, and we can see that most networks either improved or had already converged before, except one case where the error went up 0.1%, which is negligible. For the two cases that had previously been worse for the dropout, one has now overtaken its competing neural network, where as the other has reduced the gap to 0.1%, suggesting even more training time may eventually lead to the dropout network overtaking the non dropout. We have also reduced ther smallest dropout error to 42.7%, and improved the average accuracy increase to 1.75%.

Figure 4 shows the graph of validation error over time (iterations). From the left graph we can see that the network without dropout converged sooner at around 5000 epochs, whereas the dropout network continued to train past this point to eventually get a lower error. The right graph shows a zoomed image of a section of the graph, and we can see that the error on the normal network is

Table 2. Test Results with 15,000 Iterations

Architecture	Error With Dropout	Error Without Dropout
1000,500,500	45.7	46.7
1500,1000,500	46.4	47.5
1500,1000,500,250	46.4	49.1
500,500,500	44.4	45.3
500,500,1000	43.8	45.6
1500,1000,500,1000	46.0	45.9
500,500,200	45.7	48.3
800,500,500,200	44.8	48.1
500,500,500,500	42.7	44.7
800,500,500,500,250	44.1	46.8

very smooth and likely stuck in a local minima, whereas the dropout network error is a lot more jagged as it jumps in and out of minima. At the time of the non dropout network converging the dropout network had a worse validation error, however over further iterations it overtook the non dropout network.

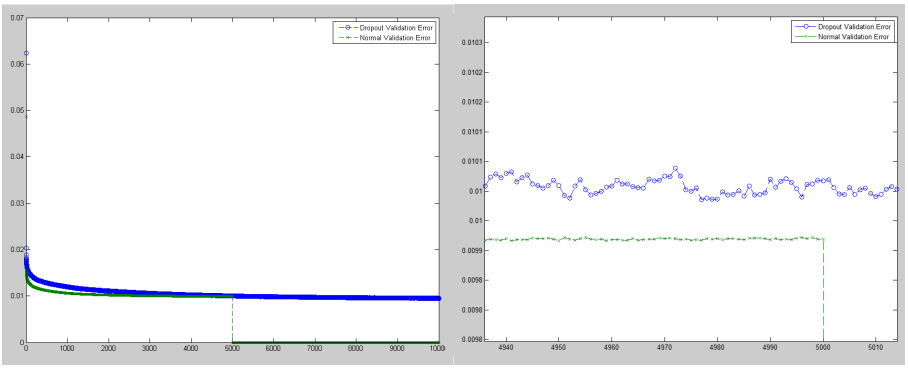

Fig. 4. Validation Error During Training for the [500,500,500,500] hidden layer architecture.

5 Conclusion

The Dropout method on deep neural networks has been shown to be effective at improving the results for image recognition of natural digits and characters. After extra iterations the average increase in accuracy was 1.75%, with the best neural network providing 57.3% accuracy. This indicates that our results are in line with the results reported in [3], which reported 55.26% accuracy achieved with Multiple Kernel Learning experiments. The same subset of images used in

[3] was selected from the dataset, but with 15 training examples per class and a balanced 15 test samples from each class. However our subsets were created using a random sample selection. The accuracy of our networks could be further improved in the future by increasing the maximum amount of iterations allowed and fine tuning the architecture of the neural network.

References

[1] Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7) (July 2006) 1527–1554

[2] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1) (2014) 1929–1958

[3] de Campos, T.E., Babu, B.R., Varma, M.: Character recognition in natural images. In: Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal. (February 2009)

[4] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324

[5] Bengio, Y.: Learning deep architectures for ai. Found. Trends Mach. Learn. **2**(1) (January 2009) 1–127

[6] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., Montral, U.D., Qubec, M.: Greedy layer-wise training of deep networks. In: In NIPS, MIT Press (2007)

[7] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics. (2010)

[8] Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. J. Mach. Learn. Res. **10** (June 2009) 1–40

[9] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) (2015)

[10] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105

[11] Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep, big, simple neural nets for handwritten digit recognition. Neural computation **22**(12) (2010) 3207–3220

[12] Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: Proceedings of the 24th international conference on Machine learning, ACM (2007) 791–798

[13] Cho, K.H., Raiko, T., Ilin, A.: Enhanced gradient for training restricted boltzmann machines. Neural Computation **25:3** (2013) 805–831

[14] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cognitive modeling **5** (1988)

[15] Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. Signal Processing Magazine, IEEE **29**(6) (2012) 82–97