Pattern Reduced Neural Networks for Breast Cancer Diagnosis

James Richardson
u5825738@anu.edu.au

Research School of Computer Science, Australian National University, Canberra, Australia

Abstract. This paper will examine the use of simple neural network techniques, both a basic single layer network and a basic convolutional neural network, on a classic classification data-set - Wisconsin Diagnostic Breast Cancer (WDBC). After establishing a baseline training on all but the testing holdout, pattern reduction was performed to examine the impact on the performance of the models. No significant gains in performance measures were found, however, $> 95\%$ accuracy was achieved with as little as 6% of the original training set on the single layer network and with as little as 7% on the convolutional neural network.

Keywords: Neural networks · Classification · Medical diagnosis · Pattern reduction

1 Introduction

The data-set for this paper, Wisconsin Diagnostic Breast Cancer (WDBC)[1], was chosen for a number of reasons. Firstly, it contains a large enough number of examples (569) and a large number of features (30) on which to train. Secondly, the goal of the data-set is meaningful - the potential for statistical techniques to revolutionise medicine is vast and, as such, it is interesting to be able to explore the possibilities. Finally, the data-set is reasonably old (1995) which allows comparing older classification techniques to modern neural networks.

The problem being modelled is that of cancer diagnosis - taking a set of observations about the cell nuclei in a biopsy of a breast mass and using those observations to decide if the given mass is malignant (cancerous) or benign (not cancerous).

This paper will focus on emulating the results of Gedeon and Bowden and their paper Heuristic Pattern Reduction[2], and then apply the pattern reduction technique on a more modern deep learning technique. That paper suggests that similar, if not better results can be gathered from only training on a subset of the total available training data. The theory presented by the paper is that when using a higher number of training patterns, neural networks are more likely to over fit to accommodate less common patterns which are not representative of the whole set. The potential for similar or better accuracy when training on a smaller set is also attractive from a training time point of view as training a neural network is linear in the number of patterns.

This paper will apply pattern reduction to a number of degrees and asses the performance on a single layer forward feed network and then on a convolutional neural network.

There are a number of papers which have applied a wide variety of techniques to this data-set, but they mainly examine novel learning techniques, not necessarily pure performance on the data-set as high levels of performance are quite easy to achieve (it took this author less than 30 minutes to build a multi-layered forward feed network capable of above 99% accuracy and, for example, an evolutionary artificial neural networks approach for breast cancer diagnosis[5][6][7] reports accuracy above 98%). As such, this paper will focus on the speed and robustness of training the network with pattern reduction within a set number of epochs instead of raw accuracy numbers.

2 Method

2.1 Preprocessing

Very little preprocessing was performed on the original data-set, as all features were already real numbers. If, for example, categorical features were included, then greater time would have needed to be spent on making those understandable to the model. The raw real numbers from the data-set were then passed through Scikit-learn's[3] standardisation preprocessor which takes each of the features and forces them into a standard normal distribution with zero mean and unit variance. This standardisation process is important as it allows the neural network model to more rapidly learn what values are going to have more or less significance depending on where they fall on the distribution. For example, if a given feature had a mean of 10, then the network will have to spend a significant amount of time learning that that feature having a value of 10 is not significant. However, if we process the data such that that mean is brought to zero, then the model will automatically discount data close to the mean as it will have a value close to zero.

Importantly, the preprocessing will not have an effect on the results of the pattern reduction, as the pattern reduction is entirely independent of and agnostic to any preprocessing technique.

2.2 Basic Forward-Feed Neural Model

A simple neural network was built to test the abilities of pattern reduction with 10 hidden units and a sigmoid used for the activation function. The network used SGD as the optimiser, cross entropy as the loss function and Pytorch[4] for the implementation. These particular settings were chosen as they represent one of the most simplistic possible neural networks, allowing the pattern reduction technique to be more easily displayed. They also mirror the settings used in the original Heuristic Pattern Reduction paper.

The network was trained with 2000 epoch over training set with a learning rate of 0.01 and then tested against a 30% hold out set. The separation of the train and test sets was stratified to ensure a representative testing set.

2.3 Convolutional Model

A basic convolutional network was constructed to test whether or not the pattern reduction principle extends to deep learning techniques. It has a basic structure compared to most convolutional neural networks, with two one dimensional convolutional layers with 1 by 5 kernels followed by two fully connected layers. The first of the fully connected layers uses a relu activation function and the second is linear. The network used SGD as the optimiser, cross entropy as the loss function and Pytorch[4] for the implementation. Once again, these settings were chosen to represent one of the most simplistic possible convolutional neural networks to allow the effect of the pattern reduction to be as clear as possible.

The network was trained with 2000 epoch over training set with a learning rate of 0.01 and then tested against a 30% hold out set. The separation of the train and test sets was stratified to ensure a representative testing set.

2.4 Reduction

In Heuristic Pattern Reduction[2] the method chosen for selecting which patterns to remove involves training a classifier on the whole dataset, a computationally expensive task. In this paper, we will use stratification as the heuristic. Taking stratified subsets should perform better compared to selecting at random when looking at smaller subsets as we will ensure that there are still a representative number of examples of both classes and can be easily performed from looking at the raw dataset.

2.5 Evaluation Metrics

To evaluate performance as the number of patterns are reduced three metrics will be used over the hold out test set.

- Cross entropy loss
- Accuracy
- Balanced accuracy as define as,

$$BACC = \left(\frac{TruePositive}{Positive} + \frac{TrueNegative}{Negative}\right)/2$$

The loss and accuracy of the model are both standard measures of model performance as the loss describes how far from the ground true each prediction is and the accuracy describing the overall quality of predictions. Including both is important as an increase in loss does not require a decrease in accuracy. A model may be 100% accurate, but only be placing just enough probability on the selection to get it over the line. This may or may not be preferable to a model which is less accurate, but more certain about the choices being made and, as such, both are included.

The addition of balanced accuracy to the more traditional metrics is to better evaluate unbalanced data sets where the number of patterns in each class in not equal. For example, if only 5% of the total population are true, then guessing all false would have an accuracy of 95% but a balanced accuracy of 0.5.

3 Results and Discussion

3.1 Results

Below are the raw results of the experiment. First for the single layer network.

Table 1: Results of reducing the number of patterns in the training set

% of training set used (number of patterns)	Cross entropy loss	Accuracy	Balanced accuracy
100 (398)	0.0980	0.9708	0.9609
90 (358)	0.0976	0.9766	0.9688
80 (318)	0.0987	0.9708	0.9609
70 (278)	0.0973	0.9708	0.9609
60 (238)	0.0984	0.9708	0.9609
50 (199)	0.0973	0.9708	0.9609
40 (159)	0.0986	0.9708	0.9609
30 (119)	0.0988	0.9649	0.9531
20 (79)	0.0950	0.9766	0.9688
10 (39)	0.1050	0.9766	0.9750

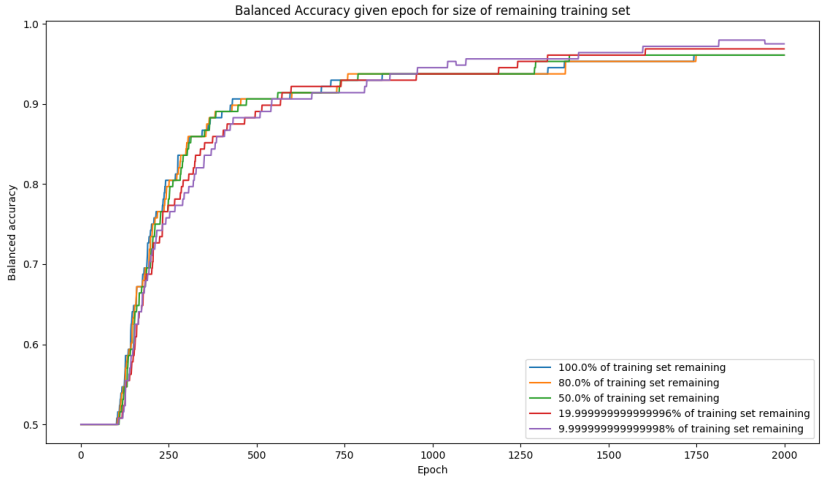

Fig. 1: The balanced accuracy of a few example sizes of training set against the number of training epoch. Only a few sizes were selected to improve readability - the full data for this graph is contained in Table 1.

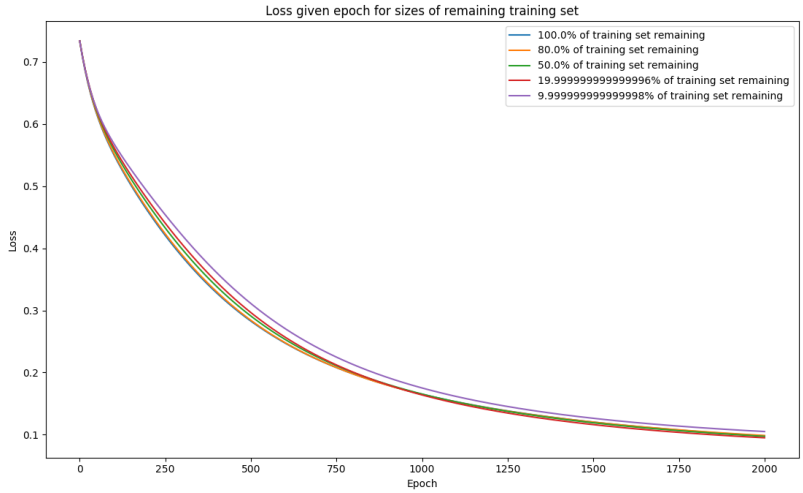

Fig. 2: The cross entropy loss of a few example sizes of training set against the number of training epoch. Only a few sizes were selected to improve readability - the full data for this graph is contained in Table 1.

As you can see, there is very little degradation on this data-set, even when reducing the size of the training set to 39 examples. As such, a finer break down of fewer than 39 patterns is also provided.

Table 2: Results of reducing the number of patterns in the training set

% of training set used (number of patterns)	Cross entropy loss	Accuracy	Balanced accuracy
10 (39)	0.1050	0.9766	0.9750
9 (35)	0.1052	0.9649	0.9594
8 (31)	0.1054	0.9766	0.9750
7 (27)	0.1022	0.9649	0.9594
6 (23)	0.1096	0.9649	0.9625
5 (19)	0.1213	0.9474	0.9297
4 (15)	0.1112	0.9474	0.9328
3 (11)	0.1624	0.9415	0.9250
2 (7)	0.1830	0.9181	0.8938
1 (3)	0.4161	0.8246	0.7688

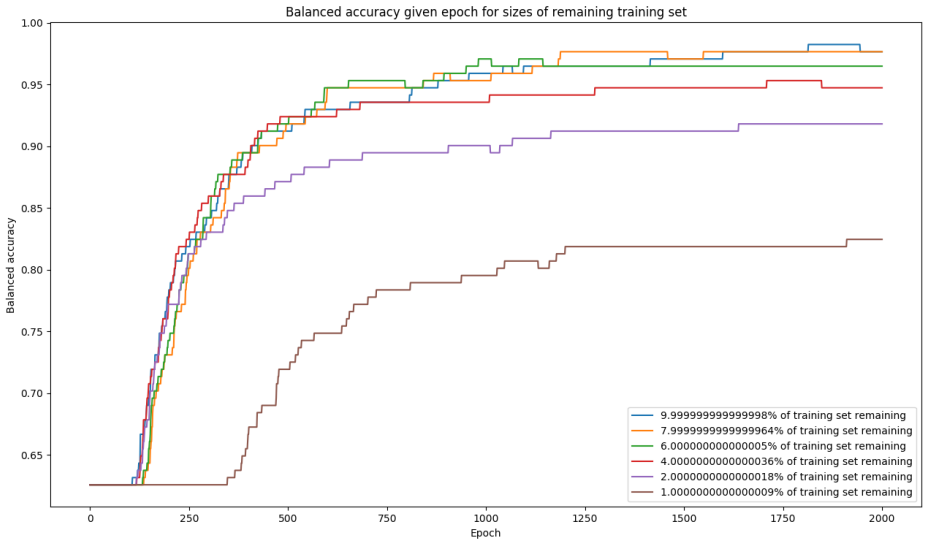

Fig. 3: The balanced accuracy of a few example sizes of training set against the number of training epoch. Only a few sizes were selected to improve readability - the full data for this graph is contained in Table 2.

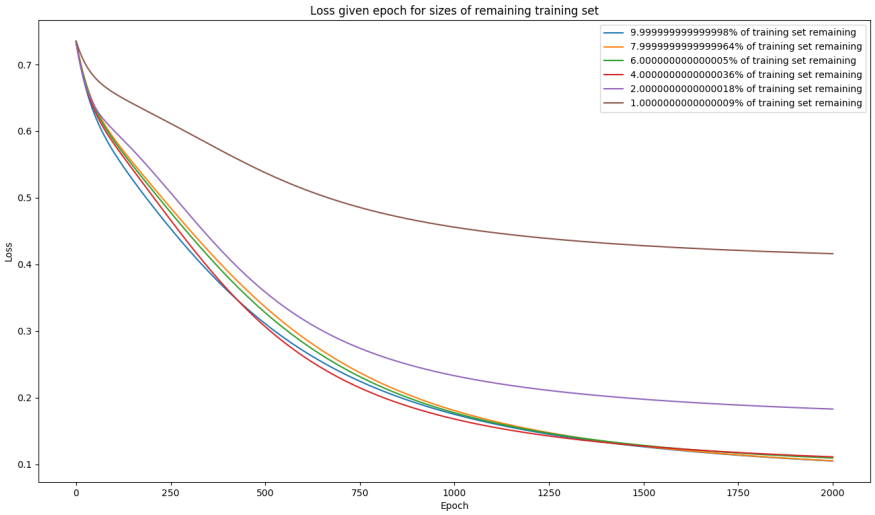

Fig. 4: The cross entropy loss of a few example sizes of training set against the number of training epoch. Only a few sizes were selected to improve readability - the full data for this graph is contained in Table 2.

Second, for the convolutional neural network.

Table 3: Results of reducing the number of patterns in the training set

% of training set used (number of patterns)	Cross entropy loss	Accuracy	Balanced accuracy
100 (398)	0.0851	0.9708	0.9641
90 (358)	0.0734	0.9649	0.9563
80 (318)	0.0823	0.9649	0.9594
70 (278)	0.0825	0.9825	0.9797
60 (238)	0.0968	0.9532	0.9406
50 (199)	0.0790	0.9649	0.9531
40 (159)	0.0911	0.9766	0.9719
30 (119)	0.1428	0.9532	0.9375
20 (79)	0.1274	0.9591	0.9547
10 (39)	0.1270	0.9649	0.9594

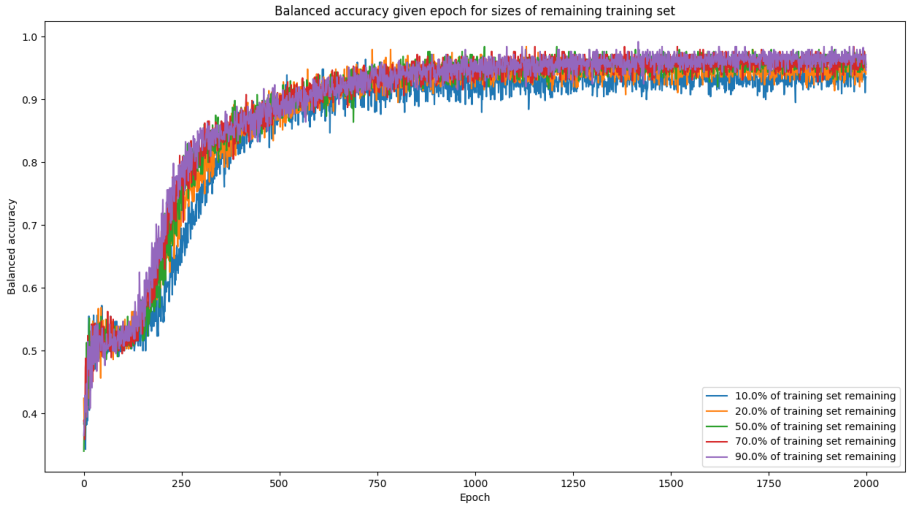

Fig. 5: The balanced accuracy of a few example sizes of training set against the number of training epoch. Only a few sizes were selected to improve readability - the full data for this graph is contained in Table 3.

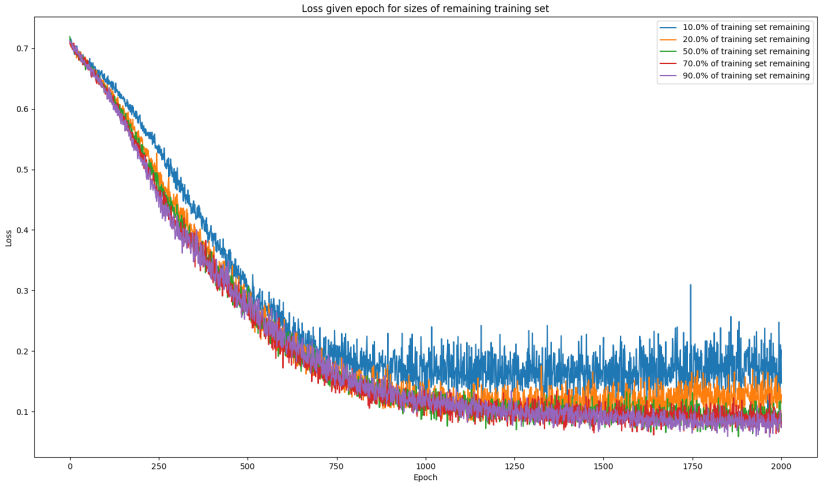

Fig. 6: The cross entropy loss of a few example sizes of training set against the number of training epoch. Only a few sizes were selected to improve readability - the full data for this graph is contained in Table 3.

Once again, there is very little degradation on this data-set, even when reducing the size of the training set to 39 examples. As such, a finer break down of fewer than 39 patterns is also provided.

Table 4: Results of reducing the number of patterns in the training set

% of training set used (number of patterns)	Cross entropy loss	Accuracy	Balanced accuracy
10 (39)	0.1470	0.9649	0.9594
9 (35)	0.1546	0.9474	0.9391
8 (31)	0.1950	0.9474	0.9360
7 (27)	0.1838	0.9532	0.9438
6 (23)	0.1629	0.9415	0.9344
5 (19)	0.1310	0.9415	0.9407
4 (15)	0.1450	0.9474	0.9454
3 (11)	0.4638	0.8655	0.8423
2 (7)	1.6121	0.7368	0.6610
1 (3)	2.8403	0.6023	0.5158

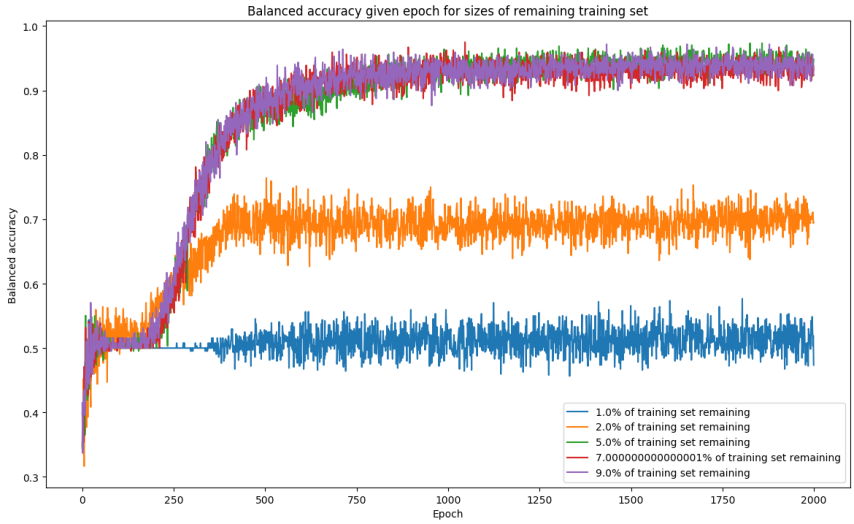

Fig. 7: The balanced accuracy of a few example sizes of training set against the number of training epoch. Only a few sizes were selected to improve readability - the full data for this graph is contained in Table 4.

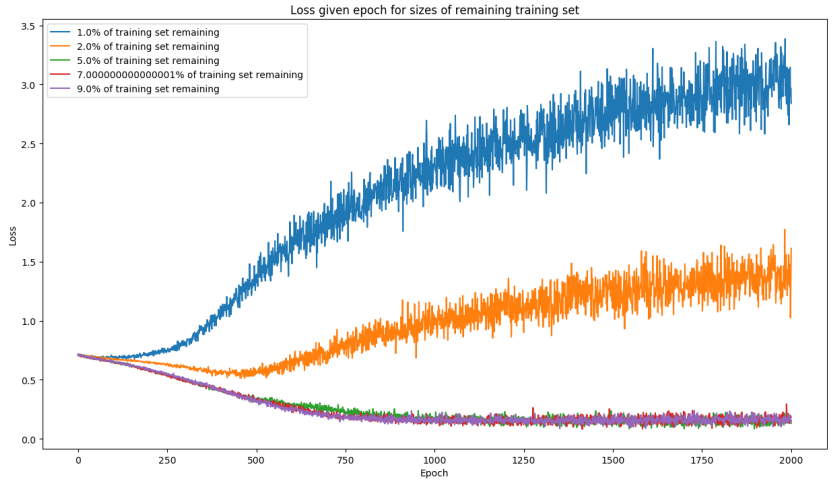

Fig. 8: The cross entropy loss of a few example sizes of training set against the number of training epoch. Only a few sizes were selected to improve readability - the full data for this graph is contained in Table 4.

3.2 Discussion

A number of conclusions are clear from the results above. Firstly, while there was no significant performance increases, there was little to no performance loss even for very small training sets when looking at both the single

layer and convolutional networks. For the single layer network, reasonable results ($> 95\%$ accuracy and balanced accuracy) could still be gathered from as little as 6% (23 patterns) of the original set proving that there was a large amount of redundant data in the full training set. Even when only training on a very small subset of three patterns the network was still able to perform reasonably well (with $> 80\%$ accuracy). For the convolutional network, reasonable results ($> 95\%$ accuracy and balanced accuracy) could still be gathered from as little as 10% (39 patterns) of the original set proving that there was a large amount of redundant data in the full training set. The convolutional network however, was much less able to cope with very small subsets of the original data, as can be seen by the accuracy and balanced accuracy dropping off in a more extreme manner in comparison to the single layer network. It should also be noted that the convolutional network suffers from particularly high variance when training on smaller data-sets, as can be seen most clearly in both Fig 3.1 and Fig 3.1. This shows that the convolutional network is much more sensitive to the learning rate and the optimiser used. For the sake of consistency both the learning rate and optimiser were held consistent for the single layer and convolutional network, but the variance would be significantly minimized if those parameters were tuned. We can also see a stability issue forming on the very low pattern numbers for the convolutional network which in the smallest case saw growing loss over time.

Secondly, the data-set at hand is easily separable. This can be seen from the fact that the single layer network only needed a handful of examples from each class to be able to achieve a good result. That makes this data-set a great example of what can happen when you hand people powerful tools, such as a large neural network. They are likely to use that very powerful tool to attempt to solve the problem when it could have been solved by a much more basic algorithm given how easily separable the data is. On data of the size used in this paper that is not a problem - training only took a few seconds for the full training set - but on a much larger data-set this result shows that it is worth attempting smaller models and training sets before training a large network on thousands of patterns. This is also clear when you compare the results of the convolutional network to those of the single layer net - while in theory the convolutional network is the more powerful tool, it at best performs similarly to the single layer network while being much slower to train due to the complexity of the model.

Finally, as the training time after model building overhead is linear in the size of the training set - significant performance gains were seen in completing the pattern reduction on both the single layer and convolutional models. While no statistics were gathered to numerically support this, it was clear in the production of the results that training time was reduced significantly as the number of patterns reduced.

The problem with the pattern reduction technique is still that it is hard to tell how much of the data-set is required in order to train effectively until attempting to train the network on the smaller data-set. The rough heuristic presented in Heuristic Pattern Reduction[2] still requires the data-set to be trained on the whole data-set, negating the training time benefits. This is particularly problematic when reduction does not increase performance of the model, such as in the results presented here.

The key design choice which has allowed this high accuracy, even down to very low pattern numbers, has been using stratification as the heuristic. This ensured that even when looking at a handful of patterns, there was guaranteed to be a representative number of in and out of class data. It is unclear whether or not stratified splitting was used in Heuristic Pattern Reduction[2], and as such it is beneficial for this paper to note it's importance when restricting training sets.

These results are also consistent with other results reported on this data-set which are usually around 98% [5][6][7], so there has not only been little to no degradation compared to the baseline used in this paper, but also when compared to other papers which have used this data-set too. However, once again, the ease of separation of this data-set sets what is a low benchmark for performance and makes this data-set a perfect candidate for training on a reduced training set. On a more difficult to separate data-set the results would likely be more interesting, displaying either the increase in performance shown in the original paper, or perhaps showing a more rapid degradation when reaching small training sets.

4 Conclusion and Future Work

This paper has shown that pattern reduction can be easily applied to both a single layer neural network model and a convolutional neural network with minimal performance degradation. Good performance was achieved by using a small subset of the original training data while significantly reducing the training time. This corroborates the findings of the original paper and saw performance in line with other papers used on this data-set. It has also shown that stratified reduction of data can be used as a highly effective, and simple to implement heuristic.

In order to verify the benefits of pattern reduction further it would need to be tested on more data-sets - particularly on data-set with varying difficulties of separation as noted in the discussion. Investigation into how to determine the size of the training set from the training data itself would also be extremely interesting and valuable, but is an extremely hard problem to solve. Pattern reduction is also always going to be domain dependent - this means that another important piece of work would be to characterise the sorts of data-sets which would benefit from this process, before the time and effort is spent. This paper used stratified data splitting as the simple heuristic for reducing the dataset - investigating other data based heuristics which do not require training would also be of

great interest. This could take the form of outlier removal or another method based on the distribution of the data itself.

References

1. Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
2. Gedeon, T.D. and Bowden, T.G. (1992) Heuristic Pattern Reduction, International Joint Conf. on Neural Networks, Beijing, vol. 2: 449-453.
3. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
4. Paszke, Adam, Gross, Sam, Chintala, Soumith, Chanan, Gregory, Yang, Edward, DeVito, Zachary, Lin, Zeming, Desmaison, Alban, Antiga, Luca & Lerer, Adam (2017). Automatic differentiation in PyTorch. , , .
5. Hussein A. Abbass. An evolutionary artificial neural networks approach for breast cancer diagnosis. Artificial Intelligence in Medicine, 25. 2002.
6. C. Campbell, N. Cristianini. (1998) Simple Learning Algorithms for Training Support Vector Machines.
7. Kristin P. Bennett and Erin J. Bredensteiner. A Parametric Optimization Method for Machine Learning. INFORMS Journal on Computing, 9. 1997.