Deep Neural Networks for Breast Cancer Classification

Rhys Healy

Research School of Computer Science
Australian National University
Acton ACT 2601 Australia
u5564974@anu.edu.au

27th May 2018

Abstract. This paper details the use of deep neural networks for a binary classification problem on the Breast Cancer Wisconsin (Diagnostic) data set supplied on the UCI Machine Learning Data Set Repository and extends on previous work that approached the same problem with shallow artificial networks. An extensive combination of standard deep learning approaches for evaluation, data pre-processing, network structure and training protocols were applied to the model to derive effective results in terms of accuracy and recall. These approaches were then extended by training on variably noisy data sets, a technique outlined in a cited research paper, with the approach slightly degrading results due to the statistical inefficiency of distorting the data's underlying distribution. Overall, results of 97.64% accuracy and 98.73% recall were marginally superior to a result using more elementary techniques but not as effective as the aforementioned previous shallow neural network or that of one adapting evolutionary neural networks for the same classification task on the same data set.

1 Introduction

In order to develop an understanding of deep neural networks and PyTorch, the Breast Cancer Wisconsin (Diagnostic) data set was selected from the UCI Machine Learning Data Set Repository to facilitate a machine learning binary classification problem. The paper endeavours to extend the previous work of Healy in *Artificial Neural Networks for Breast Cancer Classification* (*ANNBCC*) through a similar process to discern if deep learning is an effective means of analysing the problem [1]. There was a total of 699 instances observing 9 features of breast cancer tumours (radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry and fractal dimension) each coupled with a label as either benign or malignant [2].

This set was chosen particularly for three reasons. Firstly, it is the fifth most popular data set on the repository and hence has a comparatively large number of research papers citing it, giving a greater number of sources for comparisons of results and technical approaches. Secondly, the data set was chosen for the fact that it presents a binary classification problem – one of the most common problems in machine learning – with the notion that technical approaches used in its modelling could potentially generalise and be adapted to a number of other problems. Thirdly, the nature of the importance of correctly classifying cancer proved as further motivation for the choice. The large number of instances allowed for more exhaustive cross validation techniques of various approaches and a standard but not comparatively complex feature dimension yielded scope for more complex approaches to explain and predict variances in the data. The fact that neither the type of data or the number of instances and features were particularly characteristic of those that are common in deep learning applications also allowed for original analysis of the problem and approaches [3].

Feed forward deep neural networks trained with backpropagation in the PyTorch framework were used in collaboration with a supervised training methodology to train and then test the effectiveness of the model in discerning whether each instance of tumour observations was benign (negative) or malignant (positive). Modern deep learning methods were used to extend upon the more elementary implementations and future work outlined in *ANNBCC* with combinations of approaches to data pre-processing, network structure and training procedures considered for a deep neural network. The model was then tested further by adding varying levels of Gaussian noise to the input data in different schemes based on the cited paper *Decrypting Neural Network Data: A GIS Case Study* (*DNNDGCS*) [4].

2 Method

2.1 Evaluation Techniques

As the different approaches to constructing the deep neural network and analysing the data must be justified by evaluation measures, the most appropriate metrics to assess a given model for the classification task must first be derived. The primary evaluation criterion used was accuracy (1) as it is logical to strive for an approach that can correctly categorise every data point. Second to this was recall (2), which denotes the probability that an instance of a malignant tumour

selected at random is classified as such. In the context of the problem, this is an important statistic as for obvious reasons it is preferable to treat a benign tumour as malignant (false positive) as opposed to treating a malignant tumour as benign (false negative).

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{true positives} + \text{true negatives} + \text{false positives} + \text{false negatives}} \tag{1}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{2}$$

To maximise use of the data set to ensure its variances scale to a test approach, *k*-fold cross validation was used. A heuristically chosen value of $k = 10$ allowed an appropriate size of both training and testing data, with randomly selecting each partition meant the model would not implicitly adapt to specific nuances in fixed subsets of the data. Higher values of *k* and "leave-one-out" methods were not used as they often become too time complex with little differentiation in explaining power, whilst also reducing the number of combinations of random partitions. The results from all ten folds were then collated and the metrics outlined above evaluated in aggregation – random partitioning meant no benefit would come from computing the measures for each individual fold – as was a confusion matrix to better illustrate the testing classification tendencies. While it was not recorded in results, the time complexity of approaches was also restricted to approximately 120 seconds of computation to ensure relative succinctness of feedback but still allow some scope for algorithmic complexity.

2.1 Initial Approach

Data Pre-Processing. A number of pre-processing steps for the data were used in order to increase the accuracy and recall of the model, in addition to reducing the complexity of the neural network. On first inspection of the data set, there were sixteen instances with missing values for certain features which were removed. As 683 is a prime number, three more instances were removed in order to facilitate a range of possible values for *k* in cross validation and reduce the data set size to 680 with 236 malignant and 397 benign observations. Alteration of the data itself was then tested on each feature vector $\boldsymbol{v}$, first normalising – dividing each feature vector by its Euclidean norm (3) – and then standardising – dividing the difference between each instance of a feature and its mean by its standard deviation (4).

$$\boldsymbol{v} = \frac{\boldsymbol{v}}{||\boldsymbol{v}||} \tag{3}$$

$$\boldsymbol{v} = \frac{\boldsymbol{v} - \mu}{\sigma} \tag{4}$$

Subsequent analysis was applied to reduce the data set. This was deemed less important than in *ANNBCC* where the use of other reductions such as linear discriminant analysis were theorised, as it can counteract the use or scope of convolutional and pooling layers; LDA projects the data onto a single feature and makes these important aspects of deep learning redundant. As such the only dimensionality reduction considered was principal component analysis (PCA), whereby the centred input data *X*, was projected onto a lower dimension by simultaneously maximising the variance in data explained by and minimising the error in the reduction (5,6,7). By adapting Numpy's singular value decomposition function (5), the significance of each dimensional principle component in the data could be derived and then and projected onto a dimension where all features where uncorrelated. The standard method of taking angles between respective vectors was considered more elementary and less rigorous than the above methods of reduction and was hence not considered.

$$X = USV^T \tag{5}$$

$$C = \frac{1}{n} V \mathrm{S}^2 \mathrm{V}^{\mathrm{T}} \tag{6}$$

$$Z = X_k \mathrm{V} \tag{7}$$

Remark. (5) demonstrates the covariance matrix $C = \frac{1}{n} X^T X$ where the data in *X* has been centered, being used to generate eigenvalues and eigenvectors through singular value decomposition in (6) to then project onto a new dimension, *Z*, in (7).

Network Structure. As discerning the appropriateness of deep learning for the problem was deemed the primary focus of the research, alterations to structural concepts involving convolutional layers emphasised. Using the LeNet convolutional neural network as the base for this structure and any variations applied to it due to its simplicity and prevalence, the network was tested with a varying number of convolutional channels and kernel sizes, in addition to maximum, average and no pooling for these layers [5]. The number of convolutional layers was kept at two – any less and it would be difficult to classify the implementation as deep learning, and any more would reduce the size of the

already small feature dimension too much and create an overcomplicated network for the domain based on the hierarchical nature of the data discerned in *ANNBCC*. In order to minimise complexity of the task and, changes to kernel sizes remained constant for both convolutional layers, and given the small dimension of input features, pooling layers were kept to a kernel size of two with a stride of one such that they were overlapping. The convolutional network then fed into the fully connected neural network that was implemented in *ANNBCC* and was not altered throughout the process given its success in the problem domain. This network had two hidden layers – the first with thirty hidden neurons and the second with ten [1]. Because the research was analysing a binary classification problem, only one output neuron was necessary – having one for each output class only serves to add unneeded complexity to the network – and the number of input neurons to the fully connected layer was dictated by the topology of the convolutional layers. The final structural approach to be altered was the activation functions used by the network. Given its prevalence in modern deep learning, only the rectified linear unit (9) activation function in addition to it variations such as the LeakyReLU and RReLU were implemented for the convolutional layers, however only the sigmoid function was used on the output layer. This was to restrict the output within the range of 0 and 1 to create a pseudo-probability that the given instance is a malignant tumour. Creating a Bayesian network to determine the true posterior probability was unnecessary as it was only required to determine which side of 50% probability an instance was of being malignant – classified as 1 if greater and 0 if not.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

$$ReLU(x) = \begin{cases} x, x \geq 0 \\ 0, \text{otherwise} \end{cases} \tag{9}$$

Training Protocols. Numerous regimes for training the network by backpropagation were also implemented. Stochastic gradient descent and Adam algorithms were considered for optimisation functions using varying values for learning rates. The only loss function used was the binary cross entropy function (10), the most appropriate for binary classification and coupling fittingly with the sigmoidal output, but was computed with logits before applying the sigmoid in order to make use of the "log-sum-exp trick" to maximise speed of calculation [6]. Altering the number of epochs in addition to the size of mini batches within these epochs were explored while considering restrictions of time complexity. Applying dropout to the output of the final convolutional layer with different probabilities for regularisation and to reduce neuron co-adaptation was the final alteration to training used.

$$E(\boldsymbol{x}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}) \tag{10}$$

Citation Influenced Approach. The initial model chosen was then adapted to incorporate concepts outlined in the referenced *DNNDGCS*. This outlined an approach to add random noise to training data, and altering the magnitude of distortion applied at different stages of training [4]. As the paper did little to elucidate how exactly it distorted the data or how it did so to a given magnitude, liberty was taken to construct an appropriate method. For a given percentage randomness, all feature vectors of training data in the concerned epoch had added Gaussian noise distributed with mean zero and a standard deviation the product of the standard deviation of that vector multiplied by the given percentage. Two approaches were then used for training with the noise. Firstly, distortion was alternated between 0% and 10%, 20% or 40% every 50 epochs of training, each in separate test cases. Secondly, an initial distortion of 40% was applied for 100 epochs, and decremented for every 100 epochs after until a final distortion of 0% by the last 100 epochs. This was then applied for an initial distortion of 20% with decrements every 200 epochs. These two approaches while not identical to those outlined in *DNNDGCS* are very similar and their analysis provided similar insights into the problem.

3 Results and Discussion

3.1 Comparative Representation

As a large number of factors were considered in the methodologies outlined above, presenting results for all combinations of approaches proved unnecessarily complex and verbose. Instead, a range of possible alterations given in respective approaches were evaluated with respect to the final model holding all other approaches constant on a single 10-fold cross-validated test run. Note that the reported metrics are non-deterministic due to the stochastic nature in which the weights of the neural network are initialised and updated and how data is partitioned.

3.2 Data-Pre-processing

Feature Alteration.

Table 1. Effects of feature alteration on the final model.

	Unaltered Data	**Normalised Data**	**Standardised Data**
Accuracy	96.32%	96.18%	97.64%
Recall	94.92%	94.92%	98.73%

It is evident in Table 1 that while normalisation had a positive effect on both metrics that standardisation had the most significant influence. This is because when features are on a different scale, different values will update certain weights differently through backpropagation – having standardised values for each of these mean the whole network will be updated in a more consistent manner [7].

Dimensionality Reduction.

Table 2. Eigenvalues corresponding to the principal components of input features derived by PCA.

Principal Component	**1st**	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**
Eigenvalue	182.34	58.53	54.09	46.33	43.14	40.33	34.85	32.85	23.42

Table 3. Effects of dimensionality reduction with regards to final model.

Reduction	**Accuracy**	**Recall**
Unreduced Data	97.64%	98.73%
5 Principal Components	95.97%	96.61%
6 Principal Components	96.08%	95.33%
7 Principal Components	96.27%	96.17%
8 Principal Components	96.17%	94.15%
9 Principal Components	96.27%	96.61%

Remark. While the use of the nine principal components is not a reduction in the sense that it does not actually alter the feature dimension of the data set, as it projects the data onto the vector space of linearly independent eigenvectors of the decomposition it serves to decorrelate the data. Furthermore, based on the size of convolutions, the input data could not be reduced below five features when no form pooling was used or at all when it was.

By observing Table 3 it can be noted that, as expected due to the comparatively small feature size, there are no obviously low eigenvalues – that is each feature explains a non-negligible amount of variation in the data. However, it must also be noted that the first eigenvalue is far greater than three times the magnitude of any other, suggesting that it the first principal component has significant explanatory power. The results in Table 3 demonstrate that contrary to the findings in *ANNBCC* where PCA was used to reduce the feature size to one to overcome the curse of dimensionality – where high dimensional data can suffer from the of anomalies that are not as present in lower dimensions – a decrease in principal components used in fact renders a negative result [1]. Even without reducing the data but decorrelating the feature vectors, the results the standardised data are superior. These two observations can be explained by the fact that convolutions function to discern high level patterns in the data based on their correlations – reducing and decorrelating the data nullifies this correlation and the complexity of that the pattern the convolution can infer from the data [3].

3.2 Network Structure

Convolutional Layers.

Table 4. Accuracies of two convolutional layers of differing number of channels.

			2nd Layer			
1st Layer	**5**	**10**	***20***	**40**	**60**	**80**
5	95.59%	96.03%	94.71%	95.88%	96.76%	95.88%
10	-	96.47%	97.01%	97.64%	96.86%	96.47%
15	-	-	96.47%	96.23%	97.08%	96.17%
20	-	-	96.32%	95.73%	96.47%	96.17%
40	-	-	-	95.73%	96.03%	95.88%
60	-	-	-	-	96.32%	96.47%

Remark. The results were only computed based on channel combinations where the inequality channels (1st Layer) ≤ channels (2nd Layer) held, inferred from the structure of LeNet [5].

It is apparent by observing the data in Table 4 that there is little distinction between many of the results for combinations of neurons, while being marginally favourable towards those in Table 4 with mid ranges of both 1st and 2nd layer neuron numbers. This is expected given that the second layer with more channels can explain complex and higher order patterns within the data, but the marginality of the differences suggests that there are few complex tendencies in the observation data. As such, the best performing combination of 10 channels in the first layer and 40 in the second was chosen – the network was complex enough while not overly large and exhibited a simple numerical structure.

Kernel Size.

Table 5. Results with different convolutional kernel sizes.

Kernel Size	Accuracy	Recall
2	97.05%	96.61%
3	97.64%	98.73%
4	96.47%	97.46%
5	95.88%	94.92%

Remark. Kernel sizes of greater than 3 could only be computed without pooling.

Table 5 demonstrates similar results for varying kernel sizes yet a slight peak in evaluation metrics for a kernel size of 3. A kernel size too high does not segregate enough of the data to infer patterns from specific regions, and a kernel size too low does not convolve over enough features to derive patterns that their combinations generate. Hence with a feature dimension of 9, a one-dimensional kernel of size 3 was a balanced approach, as evidenced by the results.

Pooling.

Table 6. Results with different pooling functions applied to convolutional layers.

Pooling	Accuracy	Recall
None	96.18%	95.76%
Max	97.64%	98.73%
Average	96.61%	95.76%

It is evident by the results of Table 6 that there is a marginal benefit in utilising max pooling over either average or no pooling. The preference over average pooling is not unexpected as it is common in many deep learning domains to deliver superior results due to that fact it is more sensitive to the existence of a pattern as opposed to the mean value of it. The lesser results of a network structure without pooling demonstrates the effect of the curse of dimensionality. While the convolutional layers decrease the feature size, because of the number of filters applied the number of input neurons to the fully connected layers was still large. Here the pooling reduces the dimension of this input, and the reduced data can improve convergence speed and in turn result quality as it makes the required neural structure and optimisation algorithms simpler. If a more complicated optimiser and network were implemented this difference would most likely not be the case [3].

Activation Functions.

Table 7. Results of different activation functions on convolutional layers.

Activation Function	Accuracy	Recall
ReLU	97.64%	98.73%
LeakyReLU	96.98%	97.92%
RReLU	96.47%%	95.33%

It is evident that the ReLU function performed marginally better when used on the convolutional layers as opposed to activations involving its variations. This in general is due to its simplicity, the fact it more closely represents a biological neuron that either fires or remains dormant, a non-vanishing gradient and sparsity through the network [2]. The fact that there is little differentiation can best be explained because of a combination of aspects in the way the network is structured and trained that negate the minor drawbacks of the alternate approaches. LeakyReLU and RReLU are advantageous over the normal ReLU function in large networks where layers and neurons can remain dormant due to null gradients [3]. Though the network is relatively deep, because it is not as broad in terms of features as many deep learning models, this does not present as much of an issue and instead can prevent regularisation.

3.2 Training Protocols

Optimisation Algorithms.

Table 8. Results with different optimisation functions and learning rates.

	Learning Rate	
Optimiser	***0.01***	***0.1***
Stochastic Gradient Descent	96.32% (95.34%)	97.64% (98.73%)
Adam	96.62% (96.61%)	75.74% (36.02%)

Remark. The data in the table represents: “Accuracy (Recall)”.

Table 8 demonstrates minimal difference between approaches for choice of optimiser and learning rate, but marginally better performance of stochastic gradient descent with a faster learning rate. Given the data was of a smaller dimension

than is common in deep learning, it is logical that a greater step size would generate superior results, allowing faster convergence on a reduced search through local minima.

Training Length and Batches.

Table 9. Results from training with different batch sizes.

Batches	**Accuracy**	**Recall**
1	96.18%	94.92%
6	97.64%	98.73%
18	97.05%	97.46%

Table 10. Results from training for a different number of epochs.

Epochs	**Accuracy**	**Recall**
500	96.62%	96.19%
1000	96.7%	96.84%
1500	97.35%	97.46%
2000	97.64%	98.73%
2500	97.5%	97.88%

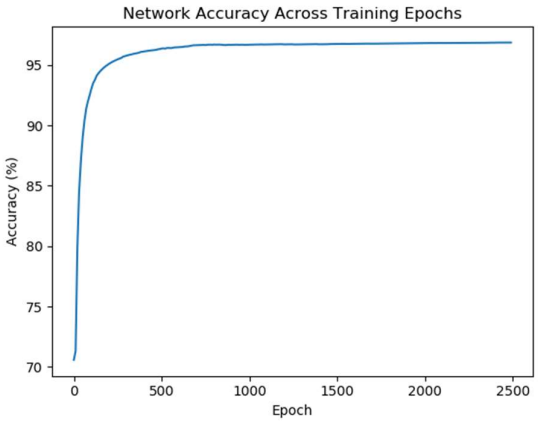

Figure 1. Network accuracy based on validation data across training epochs.

Due to mini batches, the greater step size and convergence with a combination of optimisation and dimensionality approaches, the network demonstrates both a fast and relatively smooth increase in accuracy to a peak at approximately 2000 epochs where overtraining begins to very marginally degrade results. This is a pleasing result and contrary to a common case in machine learning where there are numerous local minima in accuracies across training time [3]. As such, it was deemed that for the simplistic model, it would not be necessary to implore more complex heuristic techniques for early stopping to maximise the chance of stopping training at an optimal point.

Drop Out.

Table 11. Results from apply dropout with different probability to the convolutional layer.

Dropout	**Accuracy**	**Recall**
0	96.32%	96.19%
0.1	96.32%	96.19%
0.2	97.64%	98.73%
0.3	96.76%	96.61%
0.4	95.88%	95.34%

Remark. The dropout indexes represent the dropout percentage being applied.

Based on Table 9, it is evident that the addition of dropout to the model makes a marginal difference in its ability to cope with unbalanced data and regularise the model. Applying this after the last convolutional layer where the number of input neurons into the fully connected layers was far greater than the feature dimension outputted by the convolutions meant a reduction in neuron co-adaptation when the number of neurons increased in the network and the connective weights were attempting to learn different tendencies within the data. It was also less likely to overtrain and lean towards classifying data as the more dominant class of benign and thus improve recall.

3.4 Citation Influenced Model

Table 12. Result of alternating noise for 50 epoch blocks.

Alternate distortion	**Accuracy**	**Recall**
10%	96.76%	95.76%
20%	96.47%	95.76%
40%	96.18%	94.92%

Remark. The distortion was alternated with undistorted data on the odd block of 50 epochs.

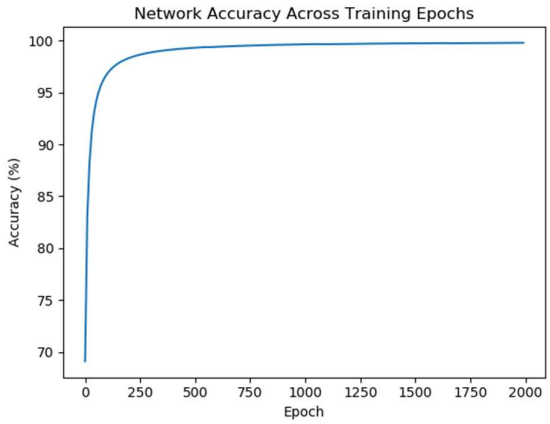

Figure 2. Accuracy across epochs with alternating 20% distortion

Table 13. Result of decrementing noise in epoch blocks.

Initial distortion	**Accuracy**	**Recall**
40%	96.91%	95.33%
20%	96.62%	95.76%

Remark. The initial distortion of 40% was decremented over 100 epoch blocks to 0%, and the 20% over 200.

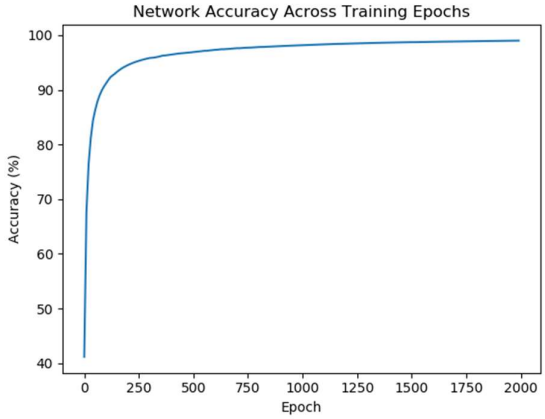

Figure 3. Accuracy across epochs with distortion decrementing from an initial 20%.

The results demonstrate similar yet slightly less significant results to those derived without distorted data, only differing marginally between the alternate and decremental approaches, in addition to a smooth and similar speed of convergence. The greater decrease in recall than the previous 98.73% is a notable observation – the added distortion was intended to regularise the model and decrease overfitting but instead had the opposite result, with a greater tendency to classify an instance as the dominant class of benign and hence increasing the number of false negatives. The deterioration of evaluation metrics is likely indicative of the statistical inefficiency of adding Gaussian noise to a data set that is most likely of a different and far more complex distribution. The noise distorts the distribution of the input data making it more difficult for the model to predict the underlying distribution and hence classify data instances correctly [3]. This is underpinned by the results in Table 12, where the results of alternating with less noisy data are indicative of the deep network better learning the distribution of the initial data.

3.5 Final Model

Evaluation. Based on the above statistics the final model incorporated the following approaches: reduction onto one dimension using PCA, two convolutional layers with a kernel size of 3 using max pooling – the first with 10 channels, the second with 10 – both with a ReLU activation function, stochastic gradient descent as an optimiser with a learning rate of 0.1, training for 2000 epochs and a dropout on the last convolutional layer with a probability of 0.2. The fully connected layers were of the same structure as that in *ANNBCC* with two hidden layers consisting of 30 and 10 neurons, respectively. In many cases, the distinction between choices in these approaches were small when considering an alteration of only one of the implementational aspects, however their aggregation had a more significant effect. Though the results in 3.4 demonstrate only little decay in evaluation metrics with the addition of Gaussian noise to the input data in various distortions and schemes emulating the reference literature, due to the decrease in recall by approximately 3% and the statistical inefficiencies underlying the approach, it was deemed that it would serve no benefit in being applied

Comparison.

The outcome of this final model – 97.64% accuracy and 98.73% recall – can then be compared to previous work on the same classification problem for the data set both using 10-fold cross validation in their evaluations. Wolberg in his 1994 paper used more elementary methods such as decision trees and separation through linear programming techniques similar to that of LDA to achieve an accuracy of 97% [8]. More recently, Abbas in 2002 implored evolutionary and memetic pareto artificial neural networks to attain a similar result to this research of approximately 98% accuracy [9]. His methods used only a single hidden layer with less than ten neurons whilst concentrating more significantly on the evolutionary algorithms and differential vector optimisation. The work in *ANNBCC* using a larger artificial neural network than Abbas but without deep learning and the use of PCA for the reduction of features onto one dimension managed 98.73% accuracy and 98.72% recall [1]. This suggests that while performing soundly on the task and not differing greatly from other results, that deep learning is not the most appropriate approach for the classification task. This is consistent with literature where convolutional and pooling networks are better for spatial problems where the data within convolutions are better correlated. This was not a case of a spatial problem, with the features and their ordering not correlated in a way that maximised the effectiveness of convolutional functionality.

4 Conclusion

4.1 Summary

A feed forward deep neural network trained with backpropagation was used to predict benign and malignant tumours based on nine features from the Breast Cancer Wisconsin (Diagnostic) data set outlined in the UCI Machine Learning Data Set Repository. By comparing different approaches to data pre-processing, deep learning structures and training protocols with respect to the validation metrics of accuracy and recall with 10-fold cross validation, the data was standardised and a network with two convolutional layers utilising pooling to avoid "the curse of dimensionality", feeding into the fully connected layers in *ANNBCC* due to their effectiveness in the problem domain was derived. Many of the final approaches when compared to other possibilities and holding other variations constant had only minimal positive effects on the model, however their aggregation provided a desirable result. An approach based on the cited paper *DNNDGCS* was then implemented, to discern the existence of any benefits from adding varying levels of Gaussian noise to training data. Results degraded slightly for different schemes and magnitudes of distortion influenced training, suggesting the addition of Gaussian noise created statistical inefficiencies by distorting the more complex underlying distribution of training distributions and hence reducing the ability of the model to predict it. As such, there was deemed to be no obvious gain from adopting the approach. The derived results were then compared to two approaches from the literature for the same data set and task, with the implementation performing marginally better than an elementary method using decision trees and linear programming, and marginally poorer than that of a model imploring an evolutionary neural network. It also produced lesser results to those outlined in the previous work of *ANNBCC* that utilised a shallow neural network, suggesting that while deep learning can soundly be applied to the problem it is not the most effective approach for the domain.

4.2 Further Research

Though the previous sections detailed a rigorous approach to numerous approaches for altering the deep neural network for better performance in the classification problem, there are still a number of considerations that could be applied in an attempt to extend the model and improve the performance. In terms of data pre-processing of data, the use of the Chi-Squared feature selection method could be explored to exclude certain feature vectors given that the eigenvalues derived from PCA all demonstrated statistical significance. Non-linear methods for dimensionality reduction such as manifold learning algorithms could also be explored as there is a significant likelihood that the data is far from linear in nature, in addition to using Gaussian noise or other methods to recreate new data to balance benign and malignant instances. The training protocols could be extended by apply pre-training the neural networks and implementing more complex and rigorous approaches for early stopping. Exploring more complicated methods for optimisation suitable for deep learning such as conjugate descent and quasi-Newton methods could improve the convergence to optimal parameters and decrease the effects of the curse of dimensionality. Evolutionary algorithm approaches could be used in order to find the optimum

combination of many of these aspects of implementation for the model. Basing the network structure on deep learning topologies other than LeNet could also be explored.

Creating a Bayesian neural network to compute the true posterior probability of cancer would be a useful extension for this current application as it would be insightful to be able to diagnose a tumour as benign or malignant with a degree of certainty, but would involve a complex computational model for which there is little documentation and not currently implemented in PyTorch. An extension of the citation influence approach could involve using a generative model to emulate the underlying distribution of the data and using that distribution to add noise to the model to negate the statistical inefficiencies resulting from using Gaussian noise to distort training instances. Some of these more complex processes could be made more plausible by allowing longer training time and using the GPU or concurrent threads for processing.

5 References

1. Healy, R. (2018). Artificial Neural Networks for Breast Cancer Classification. Australian National University, Canberra.
2. 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.
3. Bishop, C, (2006). Pattern Recognition and Machine Learning. Springer, New York.
4. Bustos, R. A., & Gedeon, T. D. (1995). Decrypting Neural Network Data: A GIS Case Study. In Artificial Neural Nets and Genetic Algorithms (pp. 231-234). Springer, Vienna.
5. ResearchGate. (2018). https://www.researchgate.net/figure/Architecture-of-LeNet-5_fig3_313808170
6. Pytorch. (2018). http://pytorch.org/docs/stable/nn.html?highlight=loss#torch.nn.BCEWithLogitsLoss.
7. Raschka, S. (2014). http://sebastianraschka.com/Articles/2014_about_feature_scaling.html.
8. W.H. Wolberg, W.N. Street, and O.L. Mangasarian (1994). Machine learning techniques to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77, 163-171
9. Abbass, H. (2002). An evolutionary artificial neural networks approach for breast cancer diagnosis. Artificial Intelligence in Medicine, 25.