INNS COM00029H Open Assessment

Y3843100

1 [20 marks] Discussion of architectures.

The dataset contains fetal cardiotocograms (CTGs) from 2126 patients each of which has 21 different recorded features (input variables). The CTGs have been annotated by three expert obstetricians creating two categories of classes.[1] One is a 10 tuple with respect to the fetal heart rate FHR patterns and the other is a three tuple regarding fetal state. This gives us two supervised classification problems with respectively 10 and 3 distinct classes. Neural Network architectures that can handle classification problems need to have appropriate activation functions and the ability to specify the targets (outputs) as a finite set of discrete classes. We will discuss four different conceptual models with respect to their ability to be configured for classifying our CTG dataset.

1.1 Perceptron

The *perceptron* is a basic network structure in which our output class y is determined by a weighted sum of our inputs X that is evaluated against some hard limit (threshold or activation function) $y = H(\sum_i^X x_i w_i)$. Both the advantages and disadvantages of the perceptron are in its simplicity. On one hand we have intuitive behaviour in the fact that the perceptron finds a line that bipartitions our data space, but on the other we are limited only to linearly separable classes. That limits us to only binary classification or at best *one-vs-many*.

1.2 Multi-Layer Perceptron MLP networks

The shortcomings of the single perceptron are addressed by its orchestrated counterpart, the multi-layer perceptron *MLP*. The three main differences as highlighted by Haykin[2] are: neuron activation functions are differentiable (sigmoid functions are often chosen), unlike the hard limits we had before; between our input and output layers we construct one or more *hidden layers* containing one or more neurons; the input, output and hidden neurons are highly connected. By composing neurons together we can learn more complex patterns at the expense of more complicated learning rules. The benefit of MLPs is that that they can approximate virtually any function provided there is enough data. The disadvantages of using MLPs come from the fact that they are prone to overfitting on data and they do not necessarily have a simple intuitive meaning as the perceptron classifier. We can theoretically use MLPs for our two classification problems as we can specify the number of output neurons to be three and ten respectively. The current task limits us to only use the data provided and we cannot acquire arbitrarily many new datapoints which rises the issue of what network structure would be viable to capture the properties of our limited dataset.

1.3 Radial Basis Function RBF networks

RBF networks are a single hidden layer MLP where Euclidean distance between the inputs and some point in space associated with the neuron's centre is computed instead of linear activation function. More specifically, the

[1] Ayres de-Campos Bernardes Garrido Marques-de Sa Pereira-Leite. *UCI Machine Learning Repository*. 2000. URL: `https://archive.ics.uci.edu/ml/datasets/cardiotocography`.

[2] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. 2nd. USA: Prentice Hall PTR, 1998. ISBN: 0132733501.

hidden layers calculate a radially-symmetric function (usually a Gaussian) from the inputs $f_i(x) = \exp\left(-\frac{|x-c_i|^2}{2\sigma^2}\right)$ where c_i is the centre for neuron i. The outputs are the weighted sums of the different basis functions in the hidden layer. RBF networks classify new data points by associating them to the closest c_i. The benefit of this architecture is that it is often faster to train compared to MLPs, but a substantial drawback is that they struggle with generalising outside of the margins of the training data. But an argument can be made that for our specific task, if we get new patient data that has low response for all of our current classes, that is a potential indicator of an anomaly that would require further investigation.

1.4 Deep Learning: Convolutional Neural Networks CNN

A Deep Learning Neural Network DLNN is in essence a MLP with more than 2 hidden layers. The architectural space of DLNNs is exponentially larger than that of the shallow MLPs and one of their main advantages is that they can compactly represent a wider range of functions. Examples can include but are not limited to modelling sequenced data with recurrent neural networks, doing complex clustering (unsupervised classification) with autoencoders, automatically improving accuracy to previously solved problems with semi-supervised adversarial neural networks and doing complicated pattern recognition with convolutional neural networks CNNs. As our problems are indeed pattern recognition tasks, we will focus on CNNs. CNNs operate best on raw, unformatted feature spaces in which they learn kernel functions to discern different features of the data. The main advantage is that the features learned are space invariant, meaning that they are considered equally likely no matter where in the feature space they manifest. As the data we have is collected from different patients, some common patterns might appear in different locations. If that is the case, regular MLPs would potentially struggle to discern the same pattern between different patients. A substantial drawback of CNNs and other DLNNs is that their performance is highly dependent on the availability of data. Our case of 2126 datapoints is too small to get a convincing interpretation of the results afterwards if we use a DLNN.

In the choice of an architecture it is important to look at the data itself. Our data does not appear to be linearly separable and our task is to discriminate between all the distinct classes, therefore we rule out the perceptron as a viable architecture. MLP and RBF are theoretically viable for both of our classification problems. They both allow for supervised multiclass classification problems and they can solve non-linearly separable problems. For the sake of simplicity and focusing on other parts of the model, we assert that the low response property of the RBF network as a useful indicator for anomalies is not appropriate for our task. We would like to create a model that generalises to similar structures even if they have never been observed. As mentioned in section 1.4,utilising a CNN for that would also incur concerns over overfitting due to lack of data. Furthermore if the data we had was raw, high dimensional scans and not summarised features as it is now, we would have been able to benefit from the space invariance. In the current form of the data, this architectural complexity is not justifiable and therefore we will not be considering CNNs. The severity of the disadvantages for MLP models can be lessened by paying close attention to the meaning of the data and incrementally explore different network structures, applying Occam's razor as a prerequisite. Because of this reasoning, we will empirically compare different MLP networks for our two classification problems.

2 [40 marks] Creation and application of neural networks.

A crucial observation for our classification problems is to note that classifying biological anomalies is naturally going to mean our classes are unbalanced. This is the case for both the FHR patterns and the fetal state problems (fig. 1). To alleviate this, we use a simple oversampling method. Specifically we randomly select underrepresented cases and replicate them until they match the most common class. This method does not give us any new knowledge about the problems, but by scaling the data we take a step towards unbiased estimations when looking at the misclassification rates. Furthermore we one-hot encode our class labels for use with MATLAB's `patternnet`.

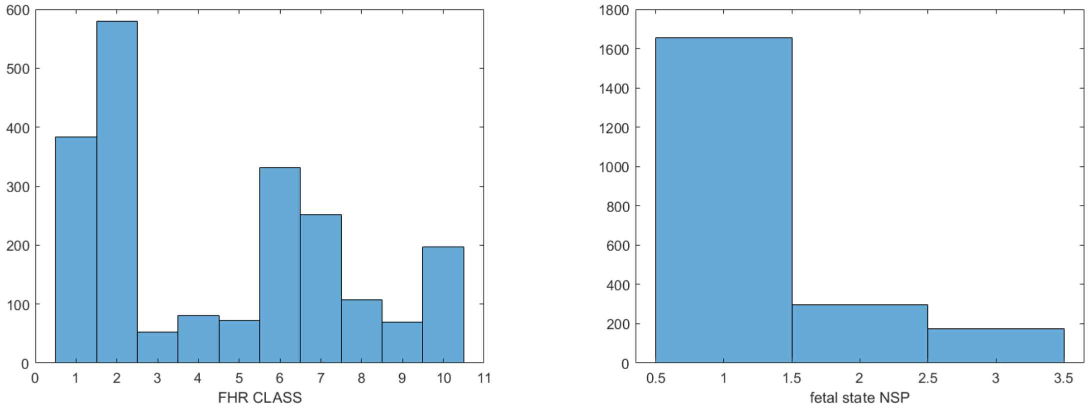


Figure 1: class distributions

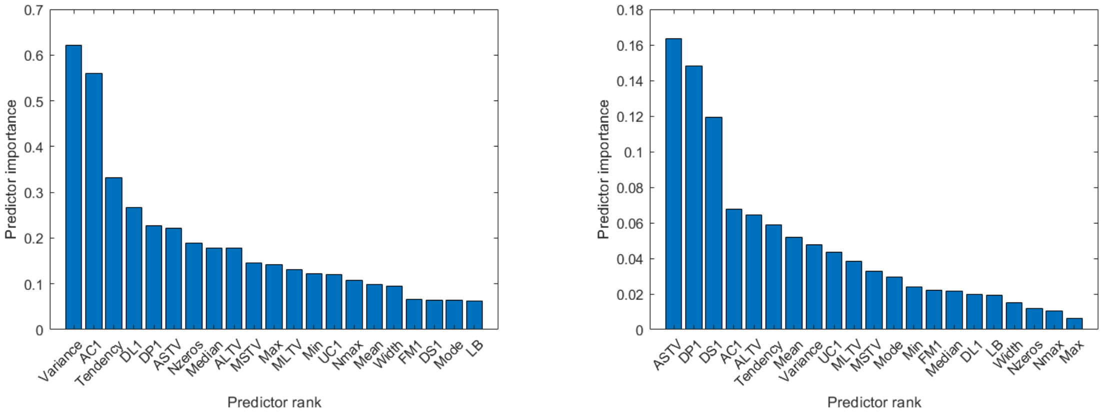
Noise reduction by removal of high frequency noise and FHR spikes has been done by the original researchers, but an interaction plot of the features reveals further preprocessing is required 3a. We observe the mean, mode and median are highly correlated. To reduce the complexity of the model we will keep only one. The choice between the three statistics is done via a feature importance analysis (fig. 2). A pairwise comparison of the mutual information of each feature and the classes of the two problems $I(X,Y)$ in search of redundancy[3] is conducted. The comparison reveals that the median and mean respectively for the FHR and NSP tasks have the highest predictive capabilities. It is unsurprising to note that the features with highest predictive capabilities are the variance, tendency and the ones related to heart rate accelerations and decelerations. Subsequently we address the skewed distributions and non regularised parameters by looking into normalising the data. An empirical test reviews that $(\mu = 0, \sigma^2 = 1)$ regularisation surpasses $[-1;1]$ normalisation performance. Moreover, the regularisation reveals that the FHR change parameters $\{AC, FM, UC, DL, DS, DP\}$ are also highly correlated (fig 3b features 2-7). In a bid to reduce complexity, we repeat the process we did for choosing the statistics features. The final 14 tuple feature subsets used for the two models are as follows:

- FHR: {LB, AC, ASTV, MSTV, ALTV,MLTV, Width, Min, Max, Nmax, Nzeros, Median, Var, Tendency}
- NSP: {LB, DP, ASTV, MSTV, ALTV,MLTV, Width, Min, Max, Nmax, Nzeros, Mean, Var, Tendency}

We disregard training algorithms that compute the Jacobian as they require the network to use a mean-squared error *MSE* loss function, which is not appropriate for our tasks (elaboration on choice of loss functions can be found in section 3). An empirical comparison of different training algorithms for a network with a single hidden layer with $N = \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$ neurons can be seen in figure 4. It appears that the family of conjugate gradient backpropagation *CGB* (traincgb, traincgp and traincgf) achieve the best trade-off between performance and training time. At a glance, the results of CGB with Powell-Beale restarts (traincgb) appears to be the most consistent (least noisy) and therefore we will chose it as a training algorithm. The initialisation parameters for traincgb are selected via a Nguyen-Widrow[4] algorithm, which evenly distributes the weights and

[3] MathWorks. *Rank features for classification using minimum redundancy maximum relevance (MRMR) algorithm*. Online; Retrieved March 24, 2020. 2020. URL: `https://uk.mathworks.com/help/stats/fscmrmr.html`.

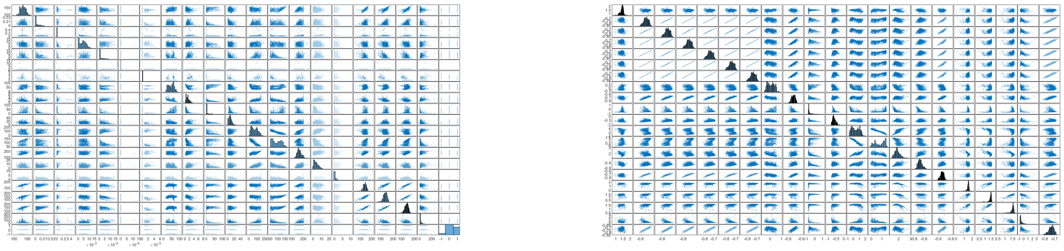
[4] MathWorks. *initnw (Nguyen-Widrow layer initialization function)*. Online; Retrieved March 23, 2020. 2020. URL: `https://uk.mathworks.com/help/deeplearning/ref/initnw.html`.



(a) FHR

(b) NSP

Figure 2: fetal state NSP feature importance



(a) original

(b) standardised

Figure 3: feature interaction matrices

biases for the active regions across neurons. CGB does not use a fixed learning rate, but calculates the learning step at each iteration. All the initialisation parameters can be seen in table 1.

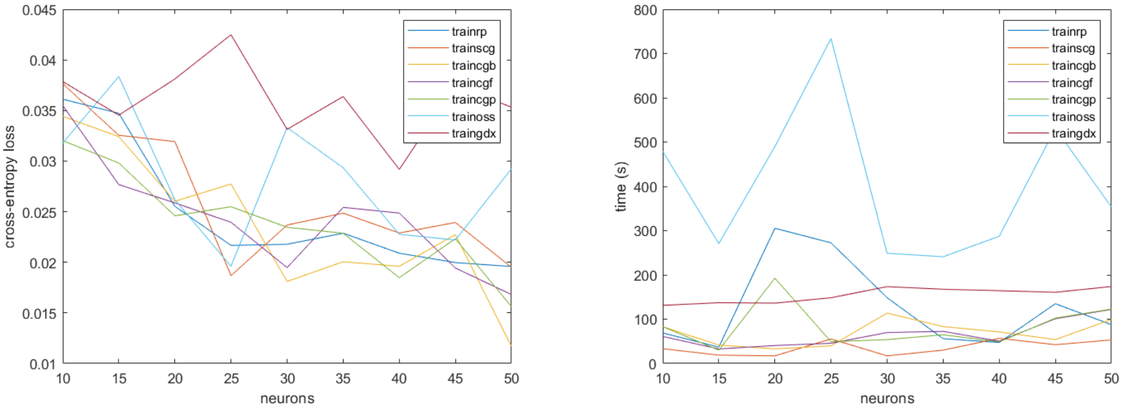


Figure 4: CE loss and training time for different training algorithms

Input layer The input layer for both problems consisted of the reduced feature set (excluding correlated) for a total of 14 neurons.

Hidden layer Achieving the optimal trade-off between network complexity and performance was done by incrementally evolving the MLP structure. Each network was ran with one hidden layer having $1, \ldots, 100$ neurons and was evaluated against the misclassification rate and the loss function (figs 5 and 6). As it can be

max epochs	loss goal	min gradient	α	β	δ	γ	min step	max step
1000	0	1e-10	0.001	0.1	0.001	0.1	1e-06	100

Table 1: Initialisation parameters for CGB

observed from the line plots, the training process is very noisy. To introduce consistency and reduce the potential of overfitting, we find the model with the least neurons that is within a threshold of the best model. The procedure is as follows:

```
$best\_model \leftarrow find(min(all\_models.loss))$
$threshold \leftarrow best\_model.loss + 0.01$
$rc\_model \leftarrow min(find(all\_models.loss < threshold))$
```

The threshold value is chosen on an uninformed assumption that a 1% error is permissible in this medical context. A better threshold value could be obtained from the client. No experimentation with additional hidden layers was conducted due to the acceptable performance of a single hidden layer network and the concern of overfitting. All models have been trained by cross-validation with a ratio $\{0.7, 0.15, 0.15\}$ respectively for the train, test and validation datasets. Metrics were obtained against the test set as it is the only truly unbiased estimator.[5] As each datapoint is from a separate patient,[6] we randomly assign single entries to the three sets as opposed to doing block randomisation. Discussion of the performance can be seen in the results section 3. The final models have 34 and 46 neurons in the hidden layer respectively for the FHR and NSP.

Output layer Our output layers have 10 and 3 neurons respectively for the two problems due to our one-hot encoding of the classes.

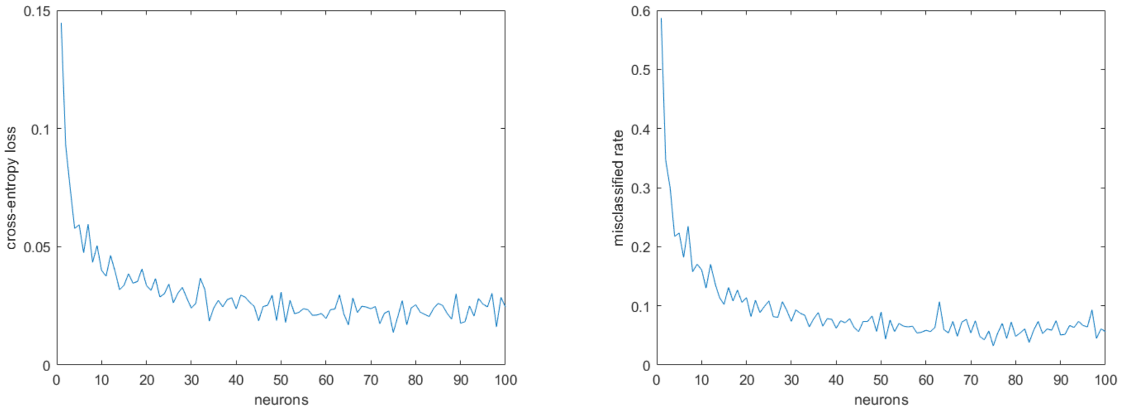


Figure 5: FHR networks performance across different sizes of the hidden layer

3 [20 marks] Results and evaluation

In regards to loss functions, as the decision space for a classification differs from that of a regression problem, we need to be wary of the usefulness of the MSE. We have chosen $crossentropy = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}(T_{i,j}log(X_{i,j}) + (1-T_{i,j})log(1-X_{i,j}))$ as our metric as it penalises neuron output proportionally on how incorrect the prediction

[5] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009, p222.

[6] Zahra Hoodbhoy et al. “Use of machine learning algorithms for prediction of fetal risk using cardiotocographic data”. In: *International Journal of Applied and Basic Medical Research* 9.4 (2019), p. 226.

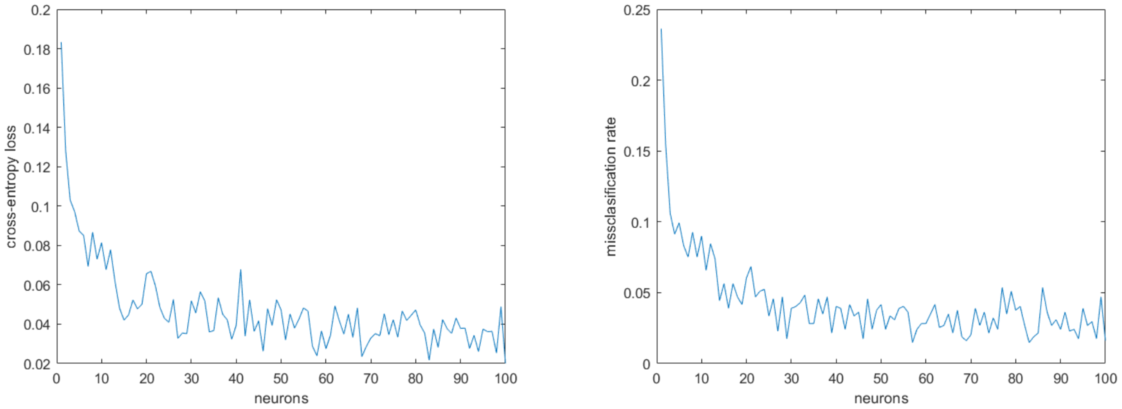


Figure 6: NSP networks performance across different sizes of the hidden layer

is. CE is shown to lead to better multinomial classification models compared to MSE.[7] A prerequisite to use it is the output neurons to have a softmax/sigmoidal activation function, which we conveniently have. To complement it with a metric that reflects our context, we also use the misclassification rates against the test set.

Model evaluation: FHR patterns (fig 7) To mitigate the possibility of good results purely due to chance, an experiment was conducted where the selected model was independently trained and evaluated 500 times. The model with the CE loss closest to the sample average was taken as a representative of our predictive capabilities. As can be seen from the error histogram 8, the model is consistent in his predictions with the error being fairly tightly centred around the zero-error margin. Our model achieves high predictive rates of above 85% for all FHR patterns. The model seems to most often confuse the calm or REM states of sleep. This is potentially due to the fact that they were the predominant classes before we oversampled the data. A closer look reveals that the calm sleep state is most often confused with the calm vigilance or the suspect patterns. It can be argued that for this medical task false positives of the suspect pattern are permissible as that would just implore further investigation. Following from this, it is good that the suspect pattern has more false positives (13.8%) compared to false negatives (6.9%). An interesting observation is that half of the false negatives mislabel the suspect as a pathological condition, which also will implore further investigation. Although the oversampling has not miraculously created new informative data entries, the contextualisation of the error assures us that this model could be useful when assessing FHR patterns.

Model evaluation: fetal state NSP (fig 9) We repeat the experiment detailed in previously and obtain a consistent average model for the fetal state prediction task. We observe that the error distribution is satisfactory centred around the zero-error margin (10). Similarly to the previous problem, not classifying a suspect or a pathological case is worse than confusing a normal case for one of the two. Due to this, we will focus on the upper half of the confusion matrix. In general the model classifies pretty accurately with the greatest errors being when it misclassifies a normal case as a problematic one (either suspect or pathological). The more alarming thing is that the model reports 9 out of 270 (3.3%) suspects are classified as normal, which although globally low (1.2%) would still mean that suspects would go undetected. The pathological case (which is arguably the most dangerous to go undetected) has no cases misclassified as normal. Even with the synthetic oversampling, this has to be taken with a grain of salt. We can observe that there is a fairly symmetric relationship between the normal being misclassified as a pathological and vice versa (1 case out 265 and 0 cases out of 210). Therefore

[7] Pavel Golik, Patrick Doetsch, and Hermann Ney. “Cross-entropy vs. squared error training: a theoretical and experimental comparison”. In: *INTERSPEECH*. 2013.

although we cannot conclude with certainty that there is a clear boundary between the pathological and normal, our model is able to convincingly discriminate between them.

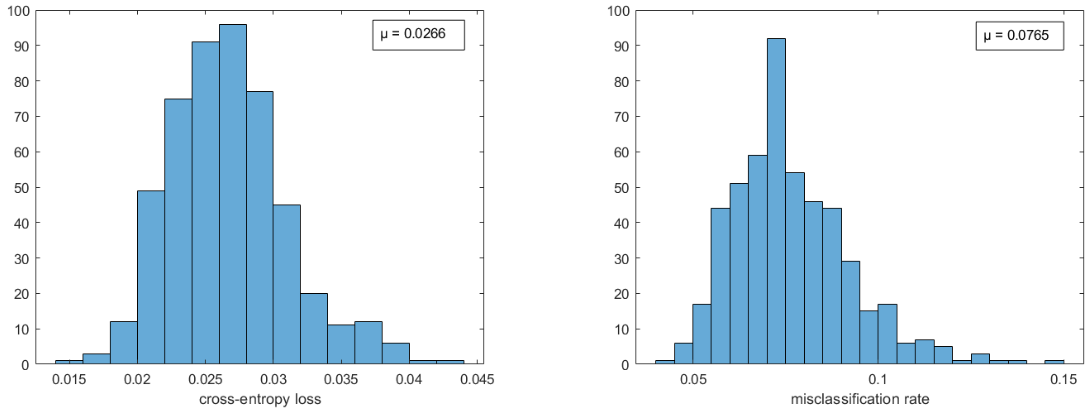
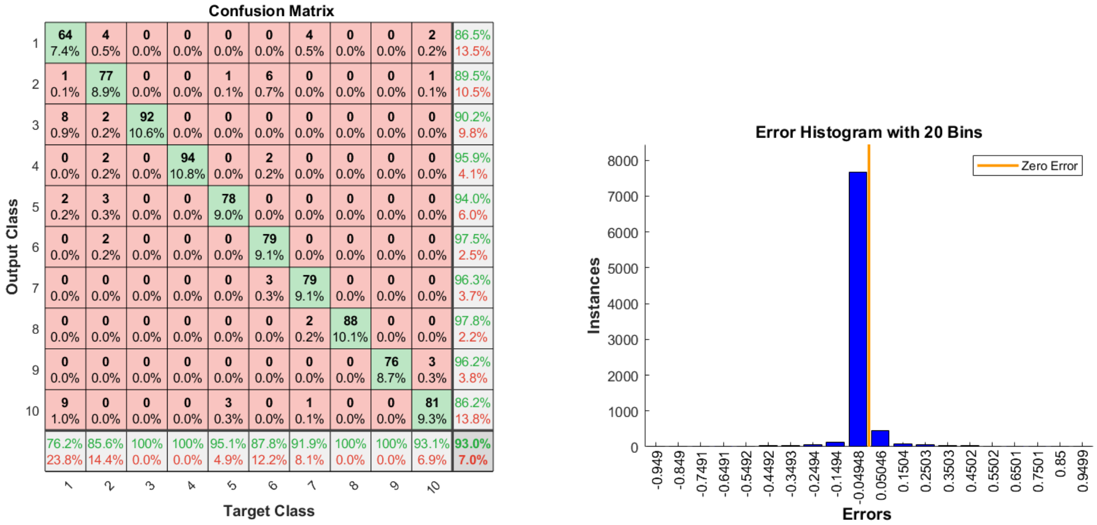


Figure 7: CE and Error rate for the FHR pattern network



	1	2	3	4	5	6	7	8	9	10	
1	64 7.4%	4 0.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 0.5%	0 0.0%	0 0.0%	2 0.2%	86.5% 13.5%
2	1 0.1%	77 8.9%	0 0.0%	0 0.0%	1 0.1%	6 0.7%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	89.5% 10.5%
3	8 0.9%	2 0.2%	92 10.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	90.2% 9.8%
4	0 0.0%	2 0.2%	0 0.0%	94 10.8%	0 0.0%	2 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	95.9% 4.1%
5	2 0.2%	3 0.3%	0 0.0%	0 0.0%	78 9.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.0% 6.0%
6	0 0.0%	2 0.2%	0 0.0%	0 0.0%	0 0.0%	79 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	97.5% 2.5%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 0.3%	79 9.1%	0 0.0%	0 0.0%	0 0.0%	96.3% 3.7%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.2%	88 10.1%	0 0.0%	0 0.0%	97.8% 2.2%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	76 8.7%	3 0.3%	96.2% 3.8%
10	9 1.0%	0 0.0%	0 0.0%	0 0.0%	3 0.3%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	81 9.3%	86.2% 13.8%
	76.2% 23.8%	85.6% 14.4%	100% 0.0%	100% 0.0%	95.1% 4.9%	87.8% 12.2%	91.9% 8.1%	100% 0.0%	100% 0.0%	93.1% 6.9%	93.0% 7.0%

Figure 8: Performance of the average FHR pattern model

4 [20 marks] Further application

We will assume we are dealing with standard 2D ultrasounds as opposed to 3D or 3D real-time (4D) scans on the basis that data from 2D ultrasounds is the most common.[8] A problem of high dimensionality such as diagnosis of fetal abnormalities from ultrasound images would benefit from a more complicated DLNN structure. Relating back to the discussion of DLNN structures in section 1.4, CNNs are a good candidate for the task in terms of training time and accuracy. The non-fully connected convolution layers reduce the number of parameters to be learned to a polynomial space compared to a shallow MLPs which grow exponentially with the dimensions of our problem. We can assert that the two assumptions about locality and invariance that CNNs have are very appropriate given the context. An abnormality would appear in a specific section of the scan and therefore it is logical to assume that the proximity of pixels is important to assess if something is wrong. Furthermore, a

[8] E. Sheiner et al. "A comparison between acoustic output indices in 2D and 3D/4D ultrasound in obstetrics". In: *Ultrasound in Obstetrics & Gynecology* 29.3 (2007), pp. 326–328. DOI: 10.1002/uog.3933.

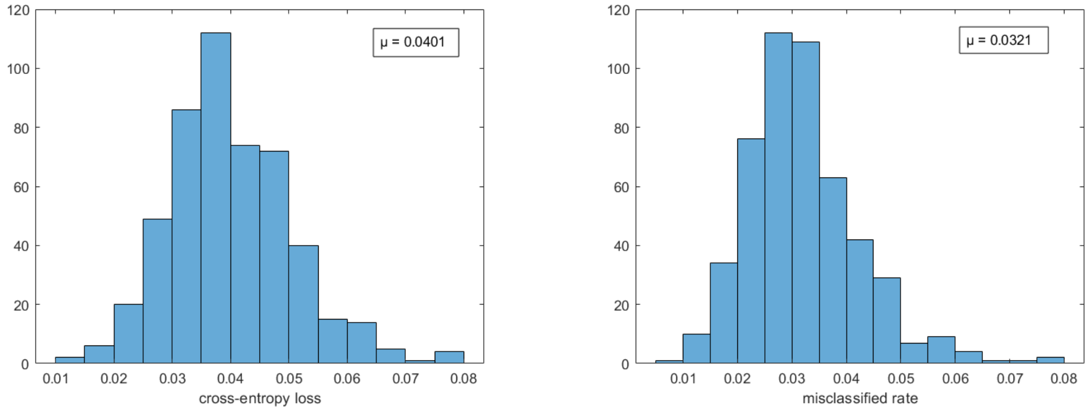
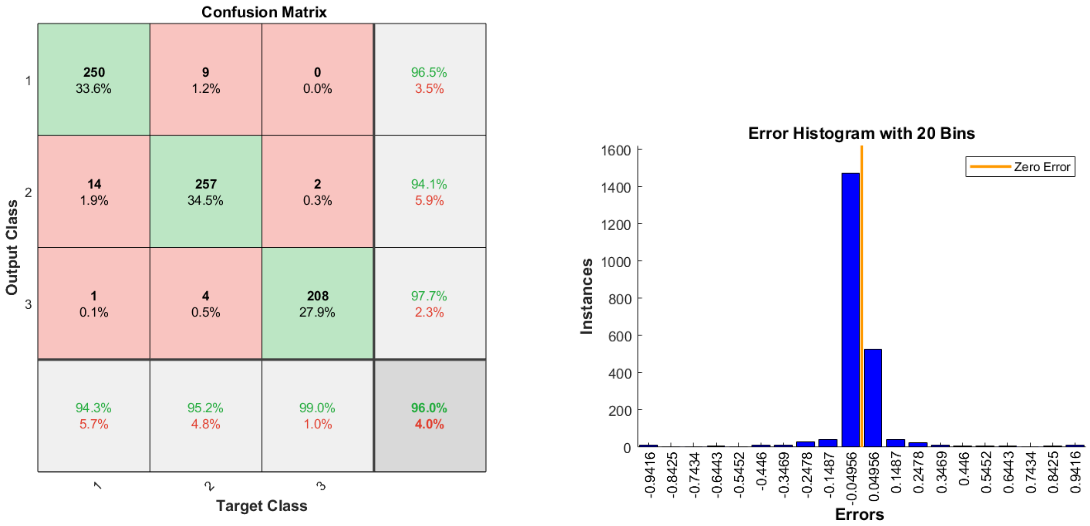


Figure 9: CE and Error rate for the NSP network



	1	2	3	
1	250 33.6%	9 1.2%	0 0.0%	96.5% 3.5%
2	14 1.9%	257 34.5%	2 0.3%	94.1% 5.9%
3	1 0.1%	4 0.5%	208 27.9%	97.7% 2.3%
	94.3% 5.7%	95.2% 4.8%	99.0% 1.0%	96.0% 4.0%

Figure 10: Performance of the average NSP model

standard MLP would not necessarily be able to discern the same abnormality appearing in two different sections of our image, whereas a space-invariant CNN can.

The input layer of the network is going to have a neuron for each pixel and the output is going to be a fully-connected layer with a neuron for each type of abnormality + the normal case. Trial-and-error procedure for estimating all the hyperparameters (depth, height and width of kernels, stride and padding) of the CNN are going to be required as there is no universally "good" network. The architecture, i.e. number of convolution-pooling pairs + fully connected layers at the end are also going to be tuned by a similar exploratory incremental approach as discussed in section 2.

Having large quantities of data gives us a convincing cross-validation split for our test, train and validation sets, without having to consider problems regarding scarcity of data (i.e. overfitting). This helps guarantee the efficacy of our model. Good practices for evaluating against the unbiased test set and using the least complex model with the highest accuracy as discussed in 2 still apply to DLNNs.