INNS COM00029H Open Assessment

Y3843100

March 25, 2020

1 [20 marks] Discussion of architectures.

This section should:

- describe (briefly) the data you have, and how much there is of it.
- identify the type of problem
- identify which classes of architectures would be suitable
- give a brief discussion of the technical features of the architectures, and the advantages and disadvantages of each
- state which class of architecture you are going to use and justify your choice, relating the characteristics of the problem to the advantages/disadvantages of the architecture.

To do this you might need to:

- do some preliminary experiments with simple versions of the architecture to get a feel for what will work
- do some exploratory data analysis to see what the characteristics of the data are
- consider the principles involved and relate them to the problem.

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 **three** different conceptual models with respect to their ability to be configured for classifying our high dimensional CTG dataset.

1.0.1 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

[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`.

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. Furthermore a perceptron can perform only binary classification or at best *one-vs-many*.

1.0.2 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 high dimensional 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. An issue arrises with what network structure would be viable to capture the properties of our high dimensional data.

1.0.3 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 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.

1.1 Deep Learning Neural Networks

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. Because of our argument that low response from the RBF network is a useful indicator for an anomaly, we will still consider RBFs in this experiment. 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 and RBF networks for our two classification problems.

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

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

This section should

- Describe the chosen inputs to (and outputs from) the networks.
- Describe how the data you started with have been preprocessed.
- Give sufficient detail for someone else to process a new batch of data for use with the final trained network.
- State which training algorithm you selected, and explain how you selected that training algorithm. For this training algorithm, give sufficient detail to enable someone to use the same training algorithm in exactly the same way. This does NOT mean (for example) describing gradient descent in great detail. It DOES mean giving any parameters, initialisation, etc, even if they are the toolbox defaults.
- Explain the process you went through in making the selection of the final architecture, for example, the number of neurons or the number of layers to use.

To do this you might need to:

- Test of one or more networks to demonstrate the effect of different preprocessing choices on the performance of the network.
- Try different training algorithms on one or more networks to compare performance.
- Evaluate a number of networks, and record details of their structures and how they performed. You may summarise repeated tests of the same structure.

A crucial observation for our classification problems is to note that classifying biological anomalies is naturally going to mean our classes are unbalanced **[[insert histogram of classes]]**. This is the case for both the FHR patterns and the fetal state problems. To mitigate this, we use a simple oversampling method that replicates data from our underrepresented cases 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. We also one-hot encode our classes for use with MATLAB's `patternnet`.

Noise reduction by high frequency noise and FHR spike removal has been done by the original researchers, but plotting the correlation and distributions of the features reveals further preprocessing is required **[[insert graph of plotmatrix before preproc]]**. Unsurprisingly, we observe the mean, mode and median are highly correlated. We keep only the median as a feature importance analysis (pairwise comparison of the mutual information of each feature and the class $I(X,Y)$ in search of redundancy[3]) reveals that it has the highest predictive capabilities. It is important to note that the analysis (fig **insert feature importance fig**) also confirms that the features with highest predictive capabilities are the variance of the CTGs, tendency and the ones related to heart rate accelerations and decelerations. Subsequently we deal with 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.

[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`.

max epochs	loss goal	min gradient	line search f(x)	α	β	δ	γ	min step	max step
1000	0	1e-10	Charalambous' method[5]	0.001	0.1	0.001	0.1	1e-06	100

Table 1: Initialisation parameters for CGB

what you need to run this NN 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 1. 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 biases for the active regions across neurons. CGB also 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.

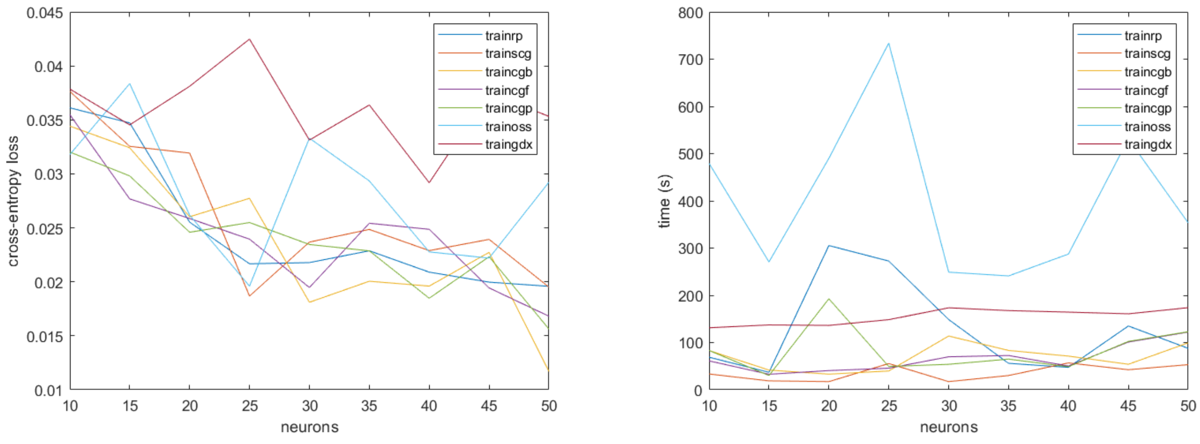


Figure 1: CE loss and training time for different training algorithms

how did we end up with this architecture The input layer for both problems consisted of the reduced feature set (excluding correlated) for a total of 19 neurons. Achieving the optimal trade-off between network complexity and performance was done by incrementally evolving the MLP structure for both problems independently. Each network was ran with one hidden layer with $1, \ldots, 100$ neurons and was evaluated against the misclassification rate and the loss function. To reduce the potential of overfitting, we find the model with the least neurons that is within a threshold of the best model:

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

The threshold value of 1% is chosen heuristically. No experimentation with additional hidden layers was conducted due to the acceptable performance of a single hidden layer network and the concern of overfitting.

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

Discussion of the performance can be seen in the results section 3. Our output layers have 10 and 3 neurons respectively for the two problems due to our one-hot encoding of the classes.

3 [20 marks] Results and evaluation

This section should

- Explain the metric or metrics you have used for comparison between networks.
- Give a synopsis of the results obtained from the final selected network.
- Evaluate the results, in relation to the problem posed in the scenario.

To do this you might need to:

- Consider different metrics for performance, appropriate to the problem. Remember that a Mean Squared Error (MSE) on its own is not always helpful in judging how well something works.
- Identify anything of interest in the results, such as areas of particularly good or poor performance, or variation between different training runs.
- Reflect on the conclusions that you may draw from the results, and whether they are showing that the neural network is useful in this case.

All models have been trained with a cross-validation with a ratio $\{0.7, 0.15, 0.15\}$ respectively for the train, test and validation datasets. All metrics are obtained against the test set as it is the only truly unbiased estimator.[6] As each datapoint is from a separate patient,[7] we randomly assign single entries to the three sets as opposed to doing block randomisation.

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 is. CE is shown to lead to better multinomial classification models compared to MSE.[8] A prerequisite to use it is the output neurons to have a softmax/sigmoidal activation function, which we conveniently have. We use cross-entropy predominantly as a model evaluation tool. 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 2) To mitigate the possibility of good results purely due to chance, an experiment was conducted where the selected model was 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 3, 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

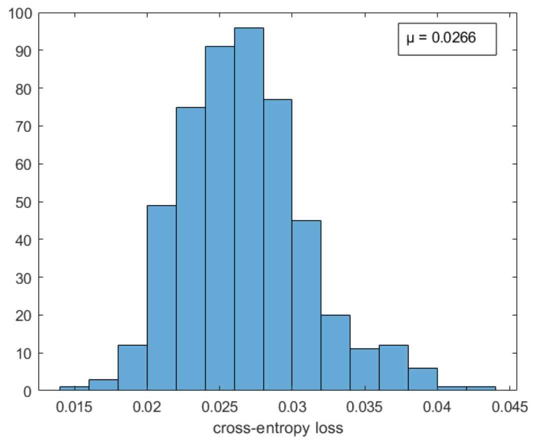
[6]Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009, p222.

[7]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.

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

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 4) We repeat the experiment detailed in the previous paragraph in order to obtain a consistent average model for the fetal state prediction task. Similarly we observe that the error distribution is satisfactory centred around the zero-error margin (5). 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 which even with the synthetic oversampling 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 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, which is an important property considering the task.



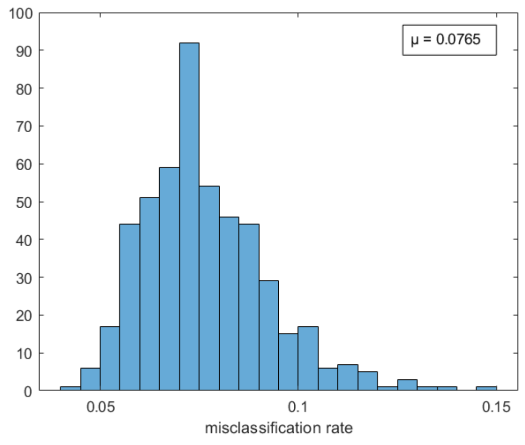
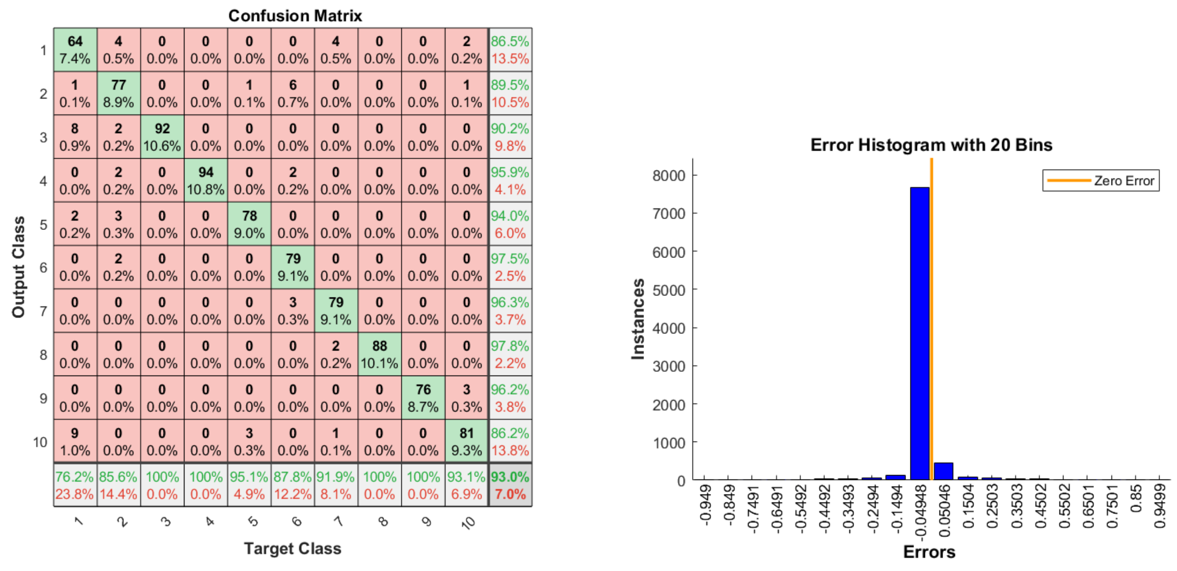


Figure 2: CE and Error rate for the FHR pattern network

4 [20 marks] Further application

In the previous sections you used a neural network to convert cardiotocogram features into a diagnosis. Another tool for detection and diagnosis of fetal abnormalities is the ultrasound scan, that produces an image of the a section through the fetus. Interpreting fetal scans is a highly complex task which require years of training. Assume the availability of a large number of fetal scan images, both normal and with some abnormality, and labelled to indicate different types of abnormalities. The task is for a neural network to process new ultrasound



Output Class \ Target Class	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 3: Performance of the average FHR pattern model

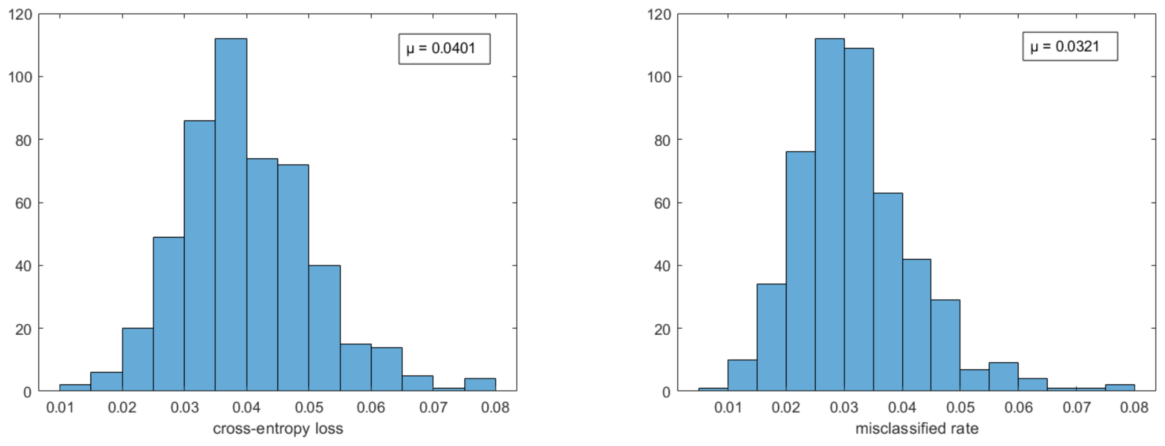


Figure 4: CE and Error rate for the NSP network

images, and to indicate which images needed further investigation. This section should discuss the issues you would need to consider in relation to:

- selection of an architecture
- construction of the network
- use of data for training
- evaluation of the network

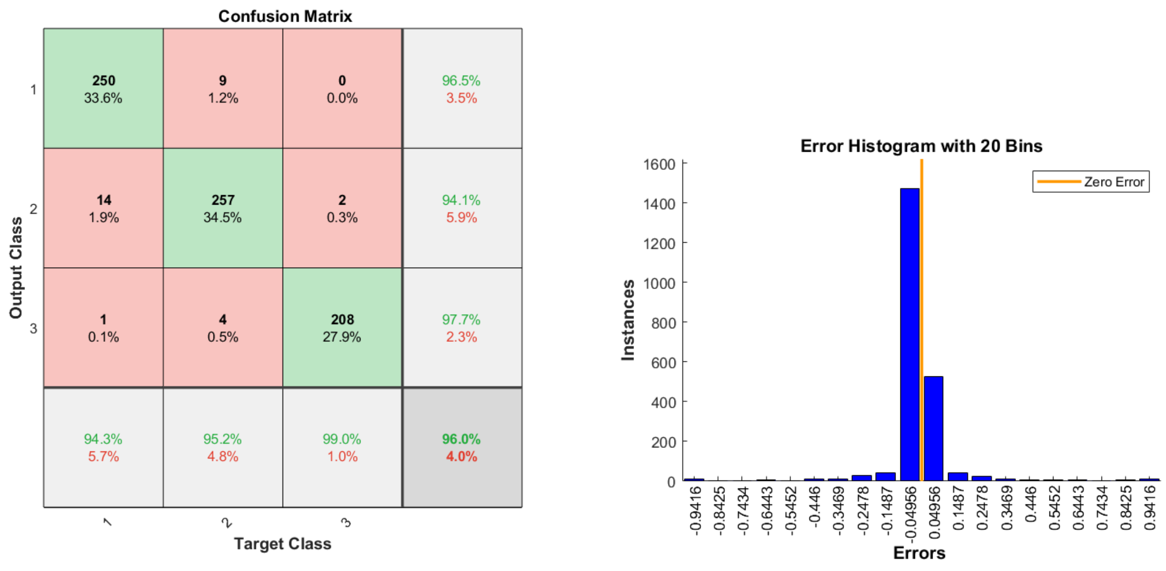


Figure 5: Performance of the average NSP model