Explain Different Neural Network Results By Sensitivity Analysis And Data Reduction Based On Genetic Algorithm

Xuecheng Zhang

Research School of Computer Science,
The Australian National University,
Canberra ACT 2601,
`u6284513@anu.edu.au`

Abstract. We live in a world surrounded by lies, but at the same time, it is often difficult for people to distinguish manipulated information[1]. It is predominant to find a tool to help people to detect deception. One method is to record the physiological signal of the observer watching the video as the output of the neural network. However, the neural network is lack of interpretability due to the black-box nature of neural networks[2]. Hence, there are two aims in my paper. The first aim is to evaluate the performance of **three-layer neural network**, **LSTM** and **multi-task learning neural network**(MTL) in deceit detection[3]. The second aim is to extract rules from these generated neural networks by sensitivity analysis and reduce the dimensionality based on the genetic algorithm and finally compare the results with the **decision tree** and **unused genetic algorithm**. In conclusion, the accuracy of **LSTM (59.3%)** and **MTL (62.0%)** exceeding the average score of human deception detection (54%) demonstrates that these tools are capable of helping people discriminate the deception and using sensitivity analysis along with data reduction based on genetic algorithm achieves **better accuracy** in all generated neural network structures compared with decision tree.

Keywords: neural network explanation · rule extraction · sensitivity analysis · genetic algorithm · LSTM · MTL

1 Introduction

1.1 Problem description

People's communications should be honest, faithful, and worthy of confidence in order to form enduring ties and achieve common goals. Despite the necessity of being able to identify falsehoods, people continuously fall short of being able to accurately detect lies[1].
When human lies, the abnormal physiological response will be detected from observers[4]. The galvanic skin response, pupil dilation, and increased heart rate seen in liars may have a subtle influence on behaviour, posture, or movement dynamics, allowing them visible to observers. People who are deceived are responsive to the overt cognitive and emotional messages of liars, and they obtain similar physiological responses with the liars[4]. The problem that the paper aims to solve is using observers' physiological signals as input and finding an effective and efficient neural network structure to predict whether the video is manipulated or not. If so, the approach may reveal the underlying distrust, assist individuals in recognising their own distrust and eventually lead to improved sincerity in public messaging.
Although the output and accuracy are obtained from a well-trained neural network, it is unknown how the neural network classifies the corresponding input because these interactions are interpreted incomprehensibly as weight vectors within a trained artificial neural network. With rule extraction, people can obtain benefits as it explicitly demonstrates the internal knowledge from the problem domain[5]. In addition, rule extraction can be used for further researches from the output of rules. Researchers can study how these signals relate to people who lie. Therefore, the paper also aims to improve the interpretability of the neural network and convince that the neural network is capable of being explained effectively. Finally, in order to compare and test the accuracy of interpretation, the decision tree is used as a baseline. (The specific reasons are explained in section **2.8**).

1.2 Definitions of Terms

Characteristic Input Method Characteristic input method aims to categorise the input patterns depending on the output. Those input patterns which trigger the output "ON" is regarded as a characteristic ON pattern[6]. When entering a new input, the distance between the input pattern and the center of other clusters needs to be calculated, and the input pattern is related to the nearest cluster.

Sensitivity Analysis In this paper, sensitivity analysis is the core principle of extracting rules from neural network output. It is to find the impact which an input has on the output of the network[6]. If the input A_i^p is a boundary, it indicates that a small change in $I^{(p)}$ will gain an enormous impact on the output C. In other words, the boundary can be determined base on the gradient value of output C with respect to the input A_i^p. (The detailed method and explanation is shown in section **2.5**).

Genetic Algorithm Genetic algorithm is a stochastic search-based algorithm based on the concepts of natural selection and genetics. The progress of GA mimics natural genetics undergoing recombination and mutation. Then, the children are generated and the process is repeated over numerous generations. Each individual is set fitness value and the higher fitness value obtains higher chance of being selected. GA algorithm is usually used when there is no optimal solution guaranteed. In my paper, GA is used for feature selection and its fitness function is accuracy of rule extraction. (The detailed method and explanation is shown in section **2.7**).

IF-THEN Rules The general form of **IF-THEN ELSE** is

$$\text{IF X } \in S(i) \text{ THEN } Y = y(i)$$

If X is a member of S(i), the output will be classified to particular class[7]. In neural network, the output Y can be regarded as $Y = f(S_j)$. Thus, given the **boundary** α, the **IF-THEN ELSE** statement can be expressed as

$$\text{IF } Y = f(S_j) \geq \alpha \text{ THEN } Y = 1 \text{ ELSE } Y = 0$$

2 Methods

The hardware basis of this paper is: Memory: 16.00GB; CPU: Intel(R) Core(TM) i7-7700HQ CPU @2.80GHz 2.80 GHz; Operating system: Windows 10; Software: python3.7

2.1 Data Preprocessing

The original dataset consists of 121 columns and 368 rows. The first column records p_{id} and v_{id}, the last column records binary output indicating whether the video is manipulated, and the rest of 119 columns record psychological signal **BVP**, **GSR**, **ST**, **PD** respectively. The subject belief experiment involves 23 valid participants and each of them has watched 16 unique videos. **pd.isnull** is performed on every column and none of the columns contains NA value. Since the dataset contains no gaps, filling in additional information in the data is not required. However, after browsing the dataset, I find that the magnitude of the data is not particularly the same. For example in **Fig.1**, the mean in column **6_bvp** is 0.005597, compared with the mean in column **20_bvp** is 480.46910045.

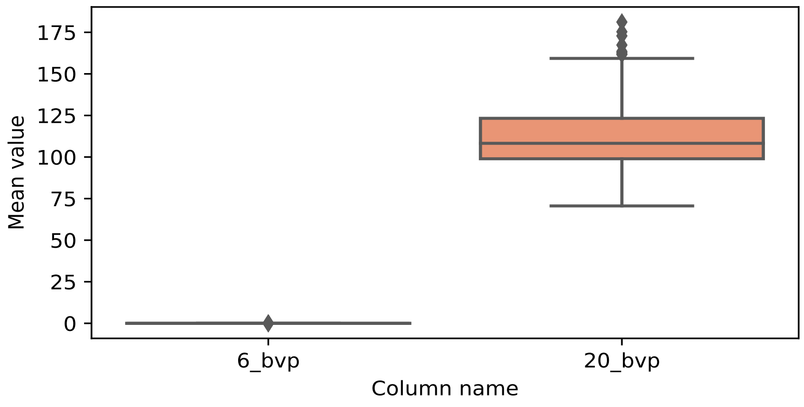

Fig. 1. Different mean value in 6_bvp and 20_bvp

The magnitude between these two columns is as much as 100 times. If normalization is not applied to the dataset, it may cause the neural network to over-learn the column with larger values. The normalization can rescale all columns from zero to one so that different values will be rescaled to the same ranges. In multi-task learning, groups are divided according to the p_id extracted from the first column and thus each of 23 group contains 16 observation data watching from different videos.

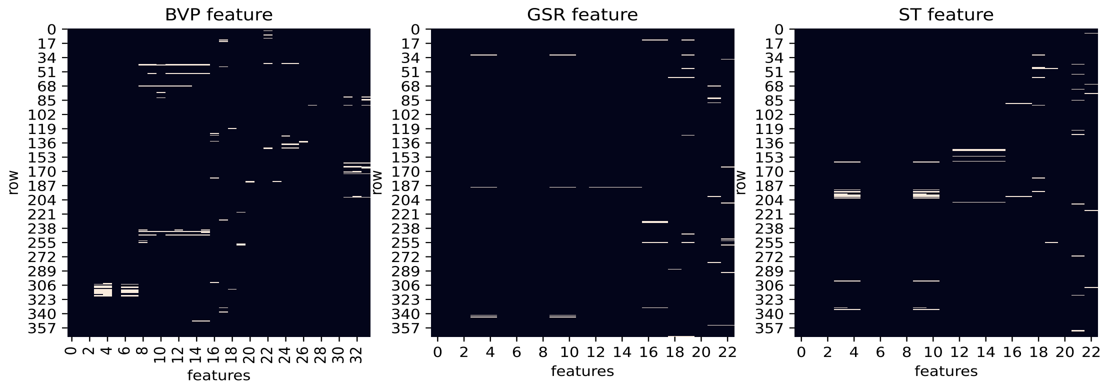

Fig. 2. Noise in BVP,GSR and ST. The white pattern represents the noise.

Afterwards, I find that BVP,GSR, ST contain abnormal data (Fig.2). In Fig.2, those white patterns indicate the noises. To de-noise these data features, inspired by [4], **Butterworth Filter** is introduced to reduce the noise effect without data removal. I use the same setting in [4] – Butterworth Filter with an order of 6 and a cut-off frequency of 0.5 Hz, 0.2 Hz, 0.3 Hz respectively to form a low-passed (LP) BVP, GSR, ST signal. After applying the Butterworth Filter (in Fig. 3), the noise in GSR and ST feature reduces significantly. Thus, we apply Butterworth Filter to GSR and ST feature with 0.2 Hz and 0.3 Hz respectively.

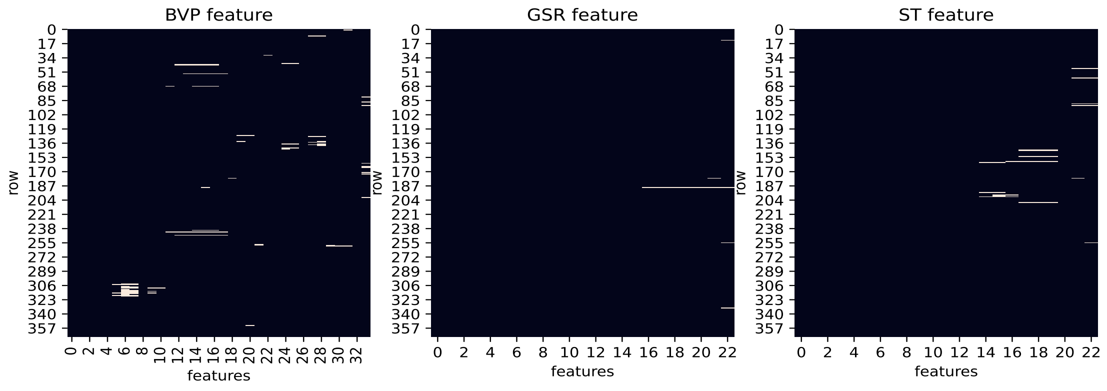

Fig. 3. After de-noise, noise in BVP,GSR and ST. The white pattern represents the noise.

Subsequently, **BVP**, **GSR**, **ST** and **PD** these four values are time sequence data. The features of the neural network cannot be directly used in the **three-layer neural network** and **multi-task learning neural network**. To maintain the information of time sequence, according to the method in this paper [4], *minimum value*, *maximum value*, *mean value*, *standard deviation*, *root mean square*, *means of the absolute values of the first difference* and *means of the absolute values of the first difference* of BVP, GSR, ST and PD classes are calculated respectively. Whereas, **LSTM** does not require the above processes since it can naturally handle time sequence.
Finally, the dataset is split into three parts: **training set** (60%), **valid set** (20%) and **test set** (20%). Ultimately, the random state is added to improve reproducibility.

2.2 Three-layer Neural Network

The **Universal Approximation Theory** [8] has proved that a multi-layer neural network can approximate arbitrary functions. Furthermore, there is no existing algorithm to predict the deception with given physiological signals. So, first of all, I utilise the**three-layer neural network** to recognise the manipulated information.
A three-layer neural network consists of input layer, hidden layer and output layer. The number of input neurons is the same as input features which is **32** and the neural network contains **two** output layers. For the output neurons, I choose the same settings **(512 hidden neurons)** with the paper purposed by *Xuanying Zhu et al.* [4]. It can be proved that a three-layered neural network with **sigmoid hidden activation** on the hidden layer can represent continuous or other types of functions described on compact sets [10]. Additionally, the sigmoid function can map any input to the range of 0 to 1. The **Cross-Entropy Loss** is suitable for classification problem. The Cross-Entropy Loss describes the distance between two probability distributions. When it applies to binary classification, our predicted probabilities are p and $1 - p$ for each category. The optimizer in the neural network is used as Adam [11]. The algorithm is able to alter the parameters according to their importance. Especially when training the input with a lower frequency, Adam updates these parameters with larger values. Since our data has 32 attributes and only 368 examples, the amount of data is small, so Adam is more appropriate.
What is more, **12-fold cross-validation** is used to evaluate whether neural network training is over-fitting. After testing several times, the **learning rate** and **epoch** are set to **0.0001** and **300** respectively because they have the best performances in the validation dataset. In addition, the seed is set to keep reproducibility. The final result containing accuracy, precision and recall of the test database is showed in **Table 1**.

2.3 LSTM neural network

However, the dataset contains time domain features such as pupil dilation and heart rate. The method of a feed-forward three-layer neural network may undermine these time series. Therefore, in order to use the data series for prediction, the paper introduces **LSTM** deep neural network model. **LSTM** contains three gates to control the information flow which decides whether to remember or to forget in the time sequence. LSTM structure can remember critical information and forget unnecessary information.
A three-dimensional input is required by the input layer of the LSTM neural network, which includes the batch size for training, input feature dimension, and time step of each input data. In each epoch, the input of LSTM neural network alters over time. The input of LSTM contains time steps and features and the input shape is reshaped to $(1, batch, 119)$. Also, the initial value of h_0 and c_0 is set to 0. In each epoch, LSTM will update h_n and c_n. The settings of the loss function, hidden activation function, optimizer, 12-fold cross-validation are the same in the three-layer neural network. The size of the hidden neuron is tested ranging from 32 to 256 and choose the best performance which is 64. Subsequently, after several attempts, the highest validation score is achieved when **learning rate** and **epoch** are set to **0.0001**,**350** respectively. Additionally, I assign the value of 0 to Pytorch seed and Numpy seed to maintain reproducibility. The final result containing accuracy, precision and recall of the test database is presented in **Table 1**.

2.4 Multi-task learning neural network

Along with ANN and LSTM, the paper also introduces **multi-task learning neural network** inspired by the paper[4]. Multi-task learning is a paradigm model in which machine learning models are trained using data from numerous tasks at the same time, utilising shared representations to understand the shared information across them[9]. Instead of training the single task like general neural network and LSTM, **MTL** neural network trains several tasks simultaneously. By dividing the entire task into small related tasks, these small tasks can learn general features in the **shared layer** and learn specific features in the **task-specific layer**. In this dataset, it records different observers' physiological signals. The deception may happen when the observers' physiological signals vary significantly. These signals from different individuals may have similar features to learn mutually in order to improve performance. Since multi-task learning prefers similar tasks, MTL neural network is implemented to predict whether the video is manipulated.
The MTL neural network consists of input layer, shared layer and a task-specific layer with 23 towers (Fig.4). Since multi-task learning may ruin the time sequence, preprocess procedure as mentioned in 2.1 is predominant to maintain the time sequence and thus the input size in MTL is 32 as described in **2.1**. According to [4], for the shared layer, two fully connected layer with 175 neurons are constructed for 23 tasks to learn general features. For task-specific layer, one hidden layer with 50 neurons and an output layer with 2 neurons are constructed for each tasks to learn their specific features. Shared layer and task-specific layer uses the same optimizer (Adam optimizer) and criterion

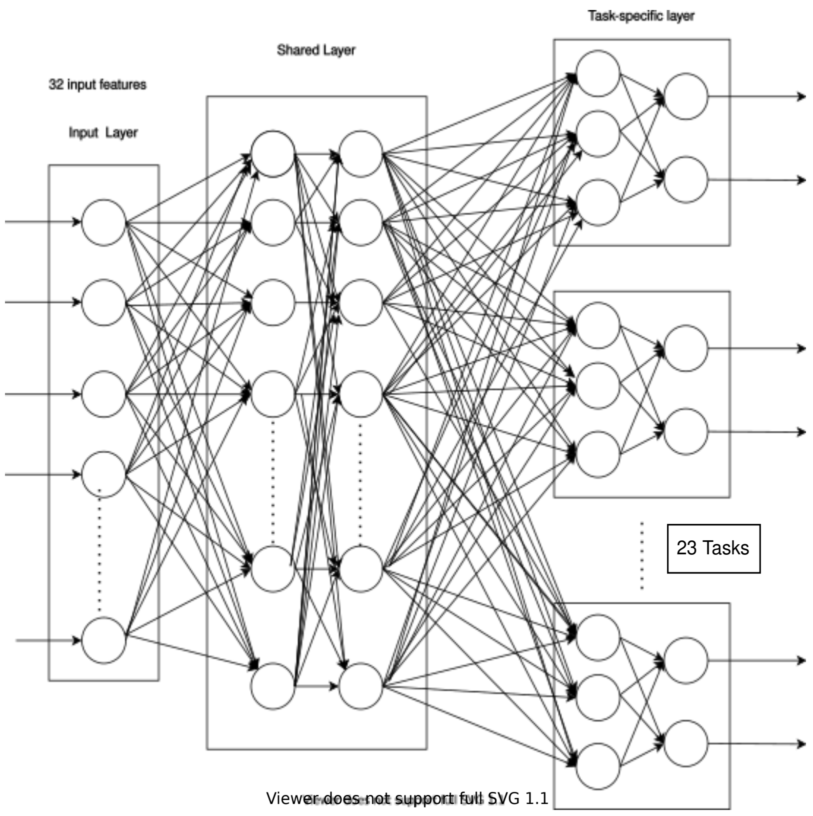

Fig. 4. The structure of multi-task learning neural network

(CrossEntropyLoss) as described in three-layer neural network. For the shared layer, each epoch requires one back-propagation. For the task-specific layer compared to the shared layer, each task requires to update their losses and perform back-propagation respectively. To implement that situation, I update each parameters in the tower respectively by executing torch.optim.Adam(mtl.tower[i].parameters(),lr=learning_rate).step() in which i represents the tower number. The **loss function** calculates the sum of the loss of each tower. *Retain_graph* is required to store the back-propagation graph until all back-propagation processes are finished. Additionally, **12-fold cross validation** is used to evaluate whether neural network training is over-fitting. After several tests comparing the validation set, the learning rate and epoch is set to is **0.00001** and **400** respectively. Finally, seed is set to keep the reproducibility. The final result containing accuracy, precision and recall of test database is showed in **Table 1**.

2.5 Sensitivity Analysis

Finding boundary The main idea of sensitivity analysis is to find the **boundary** of every attribute. Considering $\{v_1, \ldots, v_n\}$ for each continuous attributes A_i and decision boundary x_i splits the classes into two parts, i.e. subset A with $\{v_1, \ldots, v_i\}$ where $A_i < x_i$ and subset B with $\{ v_i + 1, \ldots, v_n \}$ where $A_i > x_i$ [12]. I assume that each attribute A_i has a **continuous value**. Obviously, if A_i is the boundary in i_{th} attribute, a small change in A_i will be more likely to convert the output to another class. In addition, in calculus, the derivative of a function at a certain point describes the rate of change of the function near this point. The greater the derivative at a certain point, the greater the rate of change of that point is in the function. Thus, the problem can be converted to find the maximum first-derivative value of **output with respect to** A_i. In the article written by *AP Engelbrecht* and *HL Viktor*, they mentioned that a "measure of closeness" of an attribute value to the boundary value(s) of that attribute is assigned using sensitivity analysis of the ANN response with respect to input parameter A_i.[12]. The maximum value of **F'** that approximates the function F$(A_i^{(p)}) = \frac{\partial C_j}{\partial A_i^p}$ can be regarded as the boundary of A_i. In this paper, the approach to approximate the function **F** is using Local regression (full explanation in **2.6**).

$$\frac{\partial C_j}{\partial A_i^p} \tag{1}$$

Although the boundary for each attribute is calculated, it is unknown that whether larger or smaller than the boundary triggers the rule. To simplify the problem, I only care about the situation when the value is **smaller** than the boundary for each class because the values which are larger or smaller than the boundary are symmetrical to each other. If the value $A_i^{(p)}$ is smaller than the boundary, the correspondent output class $O^{(p)}$ obtains a vote. In the end, the class with the **most elections** will be the more likely output class if the value of the attribute is less than the boundary.

Evaluation based on characteristic patterns After using *Euclidean distance* to determine which **label** the new value belongs to, each attribute value is compared to the boundary. If the value is smaller than the boundary, it indicates that the class maintains the same as the boundary of class and its corresponding class obtains a vote. On the contrary, if the value is bigger than the boundary, it indicates that the class alters to another class and the opposite class obtains a vote. Finally, the class possessed with the most vote is elected to be the output prediction. The result is shown in **Table 2.**

2.6 Local Regression (LOWESS[13])

For the regression analysis, the simplest method is the linear regression method. However, the data with periodicity and volatility cannot be fitted in simple linear regression, otherwise, the model has large variance and low bias. On the other hand, using polynomial regression might over-learn the dataset causing low variance but high bias. Using local regression, the bias and variance are balanced and a smooth line is generated which demonstrates the trend of data point distribution. The principle of *LOWESS* is diminishing the effect of outliers by weighted window. The *LOWESS* approach entails running a sequence of local linear regressions, each of which is limited to a specific window of x-values. In each window, the point closest to the center of windows obtains the biggest weight, while the points on both sides have the least weight. From the article [13], the weight should be chosen under the below conditions.

$$\begin{aligned} W(x) &> 0 \text{ for } |x| < 1 \\ W(-x) &= W(x) \\ W(x) &> 0 \text{ for } |x| < 1 \\ W(-x) &= W(x) \end{aligned}$$

In this case, I chose (2) as weight function which matches the above conditions.

$$W(x) = (1 - |x|^3)^3 \tag{2}$$

After defining the weight function $W(x)$, the data in window ranging from $[d_1, d_2]$ maps from -1 to 1 and w_i is calculated for each point. Finally, weighted regression is used to calculate $\hat{y} = X(X^T wX)^{-1}X^T wY$ [13].

Algorithm 1: Implement LOWESS[13]

```
Let function $W(x) = (1 - |x|^3)^3$, $B(x) = (1 - x^2)^2$;
Calculate the weight matrix $w$ by W(x);
Let $iter$ to be 2 or 3, $n$ to be length of data points ;
Let $delta$ be matrix filled with ones ;
for $iteration \leftarrow 0$ to $iter$ do
    for $i \leftarrow 0$ to $n$ do
        weight[i] = $delta$*w[i] ;
        Calculate $\hat{y}[i]$ by solving $X(X^T wX)^{-1}X^T wY$ ;
    end
    Calculate the loss $e = y - \hat{y}$ ;
    $delta$ = B($\frac{e}{6s}$) ;
end
return $\hat{y}$
```

From the above algorithm, we generate the $\hat{y}$ array which approximates to function **F**.

2.7 Genetic Algorithm

After performing sensitivity analysis, various boundaries are generated. That is to say, each column has a unique corresponding boundary. Nevertheless, if the columns do not have much influence on the result, adding them to the knowledge may lead to the deviation of the result. Therefore, the genetic algorithm is introduced to perform data reduction. A genetic algorithm is a random global search and optimization method developed by imitating the evolution mechanism of natural genetics. In this paper, the goal of the genetic algorithm is to select the subset of features with the best explanation accuracy, i.e. the best prediction accuracy of the generated neural network output. The genetic algorithm consists of chromosome, fitness function, selection, reproduction, and mutation. The chromosome is encoded in binary with length of feature size. 0 in array represents that the particular feature is not used, whereas 1 in array represents that the particular feature is used. Initially, 50 different individuals are initialised and encoded in binary. The fitness function is related to the explanation accuracy. The higher explanation accuracy obtains larger fitness value and thus obtains a higher probability of being selected. Proportional Selection is used as selection method. The reproduction method is *recombination with single-point crossover* and the **cross rate** and **mutation** are set to **0.8** and **0.003** respectively. For each iteration of crossover or mutation, different random numbers will be generated. When the random number is less than the cross rate or mutation rate, the process of crossover or mutation will be carried out correspondingly. The crossover and mutation can substitute deficit solutions with good solutions in population. Thus, after executing GA, the unsatisfied features can be eliminated and several decent solutions can be maintained. In the beginning, I set the number of generations to 50 but it takes a long time to generate the result. To reduce the computational time, the number of generations is set to **30** and both execution time and result are satisfying. In order to maintain the randomness of genetic algorithm, the random state is not set in the genetic algorithm. The explanation accuracy is presented in **Table.2**.

2.8 Decision Tree

Decision tree uses CART algorithm [14]. It utilises a binary recursive segmentation technology, which divides the current sample into two sub-samples, and thus each non-leaf node obtains two branches. As CART algorithm only generates binary tree, only binary decision i.e. “True” or “False” can be made in every decision. The “True” or “False” decision is similar to “IF-THEN rules” in rule extraction method. Hence, the decision tree can be used for rule extraction. The decision tree classifier divides the attributes depending on the *Gini Index.* The Gini index defines the probability that two samples are randomly selected from the dataset but their labels are inconsistent and thus the lower Gini index shows that the dataset has higher purity. Since the model is completely different from sensitivity analysis and it can be regarded as a baseline to be compared with sensitivity analysis.
The implement of the decision tree is relatively straightforward. After generating the output of the neural network, 80% of result is used for training decision tree model and 20% of the result is used for testing. After fine-tuning the parameters, the max-depth of DecisionTreeClassifier is adjusted to 3. Finally, the random state is set to 0 in order to maintain reproducibility.

3 Result and Discussion

In this paper, we have two aims. The first aim is to juxtapose the performance of three-layer neural network, LSTM and MTL neural network. To evaluate the effectiveness of these models, statistical metrics like accuracy, average precision, recall and F1score are calculated over 10 experiments based on the test dataset(Table 1.). The second aim is to introduce several rule extraction methods and compare their performances with decision tree classifier as a baseline. To evaluate the performance of these methods, each method needs to predict the output of three different neural networks and report the average value over 10 experiments(Table 2.). In these experiments, the random states are temporarily removed to maintain the randomness and the average accuracy is obtained.

3.1 Neural Network Effectiveness

Multi-task neural network obtains the highest accuracy (62.0%) and average precision (66.7%). It demonstrates that when there are similarities in the data, we can use multi-task learning neural network to train these groups separately but concurrently in the shared layer to obtain satisfactory results. LSTM neural network offers a satisfactory output. It possesses the highest average recall (73.2 %) and F1-score (67.8%). While, at the same time, the three-layer neural network conducts a deficit result. The mechanism of the general three-layer neural network is relatively insufficient to learn time series data. Even though the three-layer neural network has already undergone preprocessing, the

performance of three-layer neural network is not as superior as LSTM neural network. That is to say, LSTM is excel at time series indeed.
Moreover, we focus on the comparison of LSTM and MTL neural networks. Although MTL neural network has the highest accuracy, its average F1 score is lower than that of LSTM. It indicates that MTL neural network is more vulnerable to the imbalanced dataset. From my perspective, I think it is the complicated structure of MTL that makes it easy to over-train the data with the same result in the dataset.
In addition, another reason for the lower F1 score of LSTM is related to the loss function. The loss function, in my case, is calculated by the sum of the loss of each task (**Fig. 5**). The setting of loss function may be problematic since the learning difficulty of independent tasks is different. Different tasks may be in different learning stages. For example, task A is close to convergence, whereas task B is still not well-trained. Those tasks in the middle shown in (**Fig. 5**) clearly converge nearly in 50 epochs, but simultaneously, those tasks like in pink converge approximately in 300 epochs. The fixed weight setting may restrict multi-task learning and provide an unsatisfactory result.

Table 1. Accuracy, Average Precision and Average Recall of three-layer neural network, LSTM and MTL neural network based on subjective belief dataset

Method	Accuracy	Average Precision	Average Recall	Average F1 score
Three-layer neural network	50.7%	58.2%	59.8%	58.9%
LSTM	59.3%	63.0%	**73.2%**	**67.8**%
Multi-task learning (MTL) neural network	**62.0%**	**66.7%**	65.4%	66.0%

3.2 Rule Extraction Performance

The method of **Sensitivity Analysis + GA** performs the best solution among all generated neural network structures compared with unused genetic algorithm and decision tree algorithm. Compared with sensitivity analysis and sensitivity analysis + GA, the algorithm with GA is much more accurate than the algorithm only using sensitivity analysis, and even improves the accuracy by approximately 10% in the result of LSTM and MTL neural network. That is to say, GA algorithm utilised as feature selection extracts the important features from the output of neural network and subsequently, these features can be used for rule extraction.
For sensitivity analysis, it obtains **86.4%** accuracy in rule extraction of three-layer neural network and the accuracy is higher than decision tree classifier(77.0%). However, the performance in rule extraction of the sophisticated neural network is unacceptable. Especially in MTL, the accuracy drops to **54.3%**. The above condition indicates that sensitivity analysis alone cannot correctly extract rules from the deep neural network.
Talking to rule extraction of MTL, all models perform ineffectively in rule extraction of MTL. The best model (sensitivity analysis + GA) is dependent on sensitivity analysis and thus its performance may restrict by sensitivity analysis. In addition, the structure of MTL model consisting of shared layer and task-specific layer is more complex than the LSTM model. The classification result obtained by MTL is extraordinarily different from Characteristic ON and Characteristic OFF results during the evaluation process obtained by *Euclidean distance* classifier. For example, those data that belong to the 0 class are classified as Characteristic ON when they are classified by Euclidean distance. The above problems may lead to a bad performance on rule extraction.

Table 2. Perform 10 experiments with each method extracting rules from each output of neural network and report the average score

Rule extraction method / Neural network method	Three-layer neural network	LSTM	MTL
Decision Tree	77.0%	81.6%	60.9%
Sensitivity Analysis	86.4%	73.0%	54.3%
Sensitivity Analysis + GA	**89.1%**	**82.0%**	**64.2%**

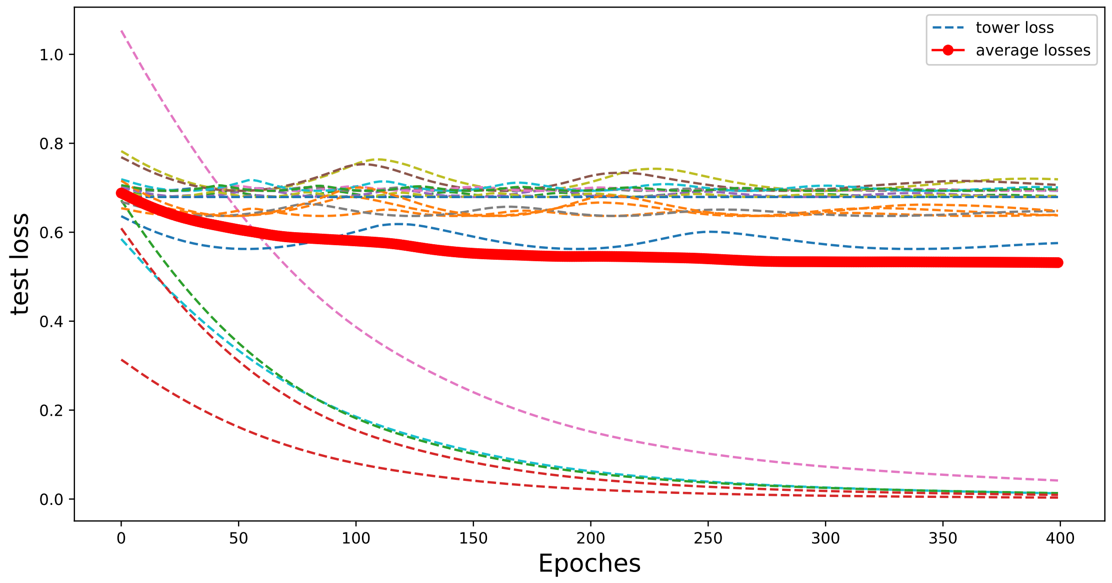

Fig. 5. Record 23 tasks of test loss and the average loss among multi-task learning neural network

4 Conclusion and Future work

4.1 Conclusion

We live in a society surrounded with lies, yet it is difficult for individuals to tell the difference between real and manipulated information. The paper firstly utilises the three-layer neural network, LSTM and MTL neural network to detect deception from observers' physiological signals. As a result, **LSTM** and **MTL** generate an acceptable accuracy surpassing human deception recognition. In this way, the neural network can assist people in recognising dishonesty. Next, since people are unknown about the behaviour of hidden neurons in the well-trained neural network, I attempt different methods to extract rules from these three neural network structures. Sensitivity Analysis with dimension reduction based on genetic algorithm successfully explains the well-trained neural network with the highest score among other tools. However, as described in section 3.1, the loss function in MTL should be improved. Also, sensitivity analysis does not reach a competent result in rule extraction of the output of MTL neural network. Thus, the methods should be improved and developed to obtain a better result.

4.2 Future work

The problem of multi-task learning that the paper introduces is fixed weight in the loss function. A good method of weighting should be dynamic, which can be adjusted according to the learning stages of different tasks, the learning difficulty, and even the learning effect. Gradient normalisation (GradNorm) [15] uses different weights to enable independent tasks to obtain the same learning rate. Dynamic task prioritization [16] hopes to offer more weighting value to the tasks that are burdensome to learn. These methods can be used for improving MTL neural network. In addition, since complex model may influence the performance, some other light model containing fewer parameters like GRU[18] can be used and compare with MTL performance.
Moreover, this paper also introduces sensitivity analysis with a genetic algorithm to extract rules from the neural network. However, its performance on the complex neural network like multi-task learning does not produce a decent result. One problem is that the Euclidean distance is not sufficient to classify the complex neural network output. I may use the Gaussian mixture model to classify the output with probability instead of directly giving the label. Another solution is utilising DeepRED [17] purposed by Zilke rather than sensitivity analysis. DeepRED uses a divide-and-conquer method that describes each layer by the previous layer [7]. The final rule that explains the whole DNN is created by combining all of the results on previous layers. I may use DeepRED on multi-task learning and compare the result with sensitivity analysis + GA.

References

1. L. ten Brinke, K. D. Vohs, and D. R. Carney, "Can Ordinary People Detect Deception After All?," Trends in Cognitive Sciences, vol. 20, no. 8, pp. 579–588, Aug. 2016, doi:10.1016/j.tics.2016.05.012.
2. Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang, "A Survey on Neural Network Interpretability," arXiv:2012.14261 [cs], Mar. 2021, [Online]. Available: https://arxiv.org/abs/2012.14261.
3. X. Zhu, Z. Qin, T. Gedeon, R. Jones, M. Z. Hossain, and S. Caldwell, "Detecting the Doubt Effect and Subjective Beliefs Using Neural Networks and Observers' Pupillary Responses," Neural Information Processing, pp. 610–621, 2018, doi: 10.1007/978-3-030-04212-7_54.
4. X. Zhu, T. Gedeon, S. Caldwell, R. Jones, and X. Gu, "Deceit Detection: Identification of Presenter's Subjective Doubt Using Affective Observation Neural Network Analysis," 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oct. 2020, doi: 10.1109/smc42975.2020.9283210..
5. R. Andrews, J. Diederich, and A. B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks," Knowledge-Based Systems, vol. 8, no. 6, pp. 373–389, Dec. 1995, doi: 10.1016/0950-7051(96)81920-4.
6. T. D. Gedeon and H. S. Turner, "Explaining student grades predicted by a neural network," Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan), 1993, doi: 10.1109/ijcnn.1993.713989.
7. T. Hailesilassie, "Rule Extraction Algorithm for Deep Neural Networks: A Review," arXiv:1610.05267 [cs], Sep. 2016, Accessed: May 27, 2021. [Online]. Available: https://arxiv.org/abs/1610.05267.
8. B. Csáji and H. Ten Eikelder, "Approximation with Artificial Neural Networks," [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.2647&rep=rep1&type=pdf.
9. M. Crawshaw, "Multi-Task Learning with Deep Neural Networks: A Survey," arXiv:2009.09796 [cs, stat], Sep. 2020, Accessed: May 27, 2021. [Online]. Available: https://arxiv.org/abs/2009.09796.
10. Y. Ito, "Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory," Neural Networks, vol. 4, no. 3, pp. 385–394, 2019, doi: 10.1016/0893-6080(91)90075-G.
11. D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv.org, 2014. https://arxiv.org/abs/1412.6980.
12. A. Engelbrecht and H. Viktor, "Rule improvement through decision boundary detection using sensitivity analysis," Lecture Notes in Computer Science, pp. 78–84, 1999, doi: 10.1007/bfb0100474.
13. W. S. Cleveland, "Robust Locally Weighted Regression and Smoothing Scatterplots," Journal of the American Statistical Association, vol. 74, no. 368, pp. 829–836, Dec. 1979, doi: 10.1080/01621459.1979.10481038.
14. S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 3, pp. 660–674, 1991, doi: 10.1109/21.97458.
15. Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks," arXiv:1711.02257 [cs], Jun. 2018, Accessed: May 28, 2021. [Online]. Available: https://arxiv.org/abs/1711.02257.
16. M. Guo, "Dynamic Task Prioritization for Multitask Learning," 2018. Accessed: May 28, 2021. [Online]. Available: https://openaccess.thecvf.com/content_ECCV_2018/papers/Michelle_Guo_Focus_on_the_ECCV_2018_paper.pdf.
17. J. Zilke, "Extracting Rules from Deep Neural Networks," M.S. thesis, Computer Science Department, Technische Universität Darmstadt, 2015.
18. K. Cho et al., "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," arXiv.org, 2014. https://arxiv.org/abs/1406.1078.