Directional or Random? Comparing the Effectiveness of Pruning with Dropout - Experiment on Multi-Task Learning Neural Networks with GA-based Feature Selection

Xujun Ge

Research School of Computer Science,
The Australian National University,
Canberra, ACT 2600, Australia.
xujun.ge@anu.edu.au

Abstract. Based on the VehicleX dataset, this paper constructs a multi-task learning neural network model with feature selection based on genetic algorithm. We selected two pruning techniques based on distinctiveness, Weight Matrix and Neuron Behaviour, to obtain neuron similarity, and called these two methods directional methods. In contrast, Dropout, which is simple and effective but also contains randomness, is popular in recent years. We aim to compare the effects of these two types of methods on the performance of the model, and find that they have a significant effect on suppressing overfitting and improving the overall performance of the model. Among them, Weight Matrix has the best performance because it is directional and is the least affected by overfitting in the early training period. The Neuron Behaviour is affected by over-fitting more, and has the worst performance. Dropout performs slightly better, but due to its higher convergence limit, it requires more training time, and its performance is worse than Weight Matrix, which may be due to its randomness.

Keywords: neural network pruning · multi-task learning · dropout · genetic algorithm · feature selection.

1 Introduction

1.1 Background

In the field of deep learning, pruning and compression of neural networks has always been a hot topic, and great progress has been made in recent years. As early as 1989, Lecun, Denker and Solla [8] propose that unimportant parameters in the network can be deleted to achieve the effect of compression model. Han, Mao and Dally (2015) [4] proposed to apply different methods such as cropping, weight sharing, quantization, and encoding to model compression, and the excellent effects have also triggered discussion. Liu et al. (2019) [9] divides the pruning methods of neural networks into two categories, which are determined by humans and determined by pruning algorithms. In fact, all algorithms are designed by people, and all models need to be driven by algorithms. Therefore, these two categories represent predefined and automatic pruning respectively.

Gedeon and Harris (1991) [3] proposes an automatic pruning method based on distinctiveness. This method dynamically deletes redundant and complementary neurons during training. Gedeon [2] proposes a second pruning method on this basis in 1995, that is, using the trained weight matrix of the neuron networks to delete redundant and complementary neurons. For a more intuitive comparison with the first method, it is also called a static method. Gedeon also points out that although the second method is computationally cheaper, it ignores the dynamic changes of neurons during training. In this paper, the first method will be simply called as Neuron Behaviour, and the second method as Weight Matrix.

The above two methods represent automatic and predefined pruning respectively. They all have a clear directivity, that is, the algorithm has fixed conditions for pruning. They not only effectively compress the model, but also effectively prevent overfitting for complex models. One of the most popular methods to prevent overfitting in recent years is dropout. Although strictly speaking Dropout is not pruning, it effectively prevents complex co-adaptation between neurons by randomly discarding a certain proportion of neurons during each forward propagation process (Srivastava et. al, 2014) [12]. It can also be seen as a combination of multiple pruned models to achieve a similar effect to pruning. Compared with directional pruning, it has great randomness.

Multi-task learning is a concept corresponding to single-task learning. Its advantage is that it can learn multiple tasks at the same time when training one model. One of its keys is parameter sharing. Vandenhende et. al (2021) [13] introduced the current commonly used network structure and training methods, which includes hard parameter sharing and soft parameter sharing (see Fig. 1). [13] Although at present, the most discussed multi-task learning lies in the network structure and the optimization of loss [7, 11], this article will try to discuss the effect of different pruning methods in the multi-task learning model.

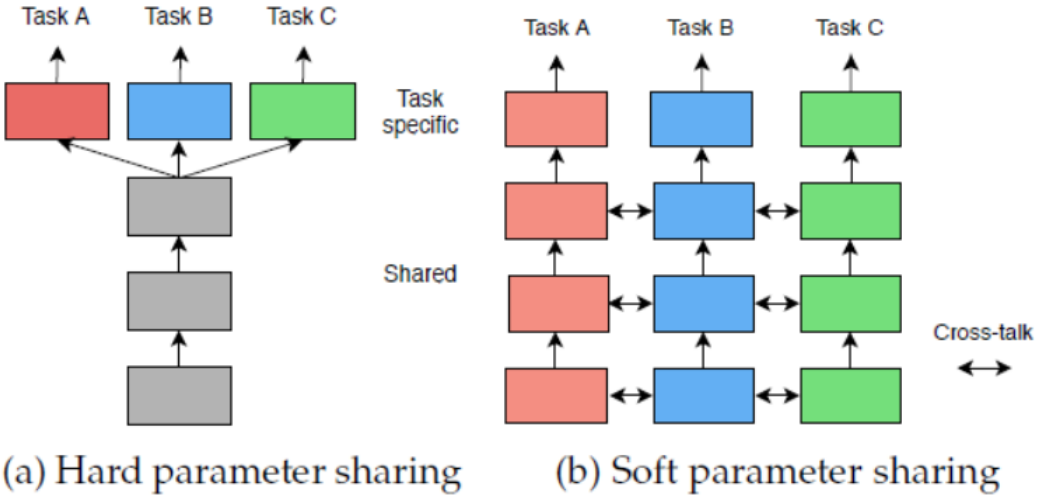


Fig. 1. Soft Parameter Sharing and Hard Parameter Sharing in Multi-Task Learning
Note: Retrieved from *Multi-Task Learning for Dense Prediction Tasks: A Survey* [13]

1.2 Dataset Description

The dataset used in this article is VehicleX V1 dataset, and is based on the *Simulating Content Consistent Vehicle Datasets with Attribute Descent* published by Yao et. al in 2020 [14]. The dataset used in this article was generated by the publicly aviablable 3D engine VehicleX. This data set is also the largest synthetic dataset currently used for vehicle re-ID. The dataset contains 2048 features extracted from ResNet which is pretrained on ImageNet. The dataset has a total of 75,516 image feature sets, which are divided into training set, validation set and test set at a ratio of close to 6:2:2. In addition, the data set also has the ground truth of each picture, including 9 labels such as vehicle ID, camera height, and vehicle color ID. This article uses vehicle type ID and vehicle color ID.

We believe that the number of features in this dataset is relatively large. When conducting large-scale experiments, 2048 features is a satisfactory number. However, in the model in this article, a large number of features may cause the model to be too complex and easy to overfit. Therefore, in the preprocessing process, this paper uses genetic algorithm to perform feature selection.

1.3 Task Description

This article will design a multi-task learning neural network using the VehicleX dataset [14]. The two tasks chosen are the classification of vehicle type ID and vehicle color ID. The neural network model is a classic hard parameter sharing model [13], that is, it contains many shared layers. For these shared layers, the two methods used by Geodon [2,3] are attempted in this article for pruning. Then, a third method which simply using dropout to the baseline model is implemented. Finally, this paper compares their effects on model performance. The goal of this paper is to analyse whether the random dropout is able to replace some directional pruning methods on a multi-task learning model. As two similar but different optimization model methods, comparing and analyzing them is helpful to better understand their advantages and disadvantages, so that these methods may be improved (for example, part of the concept of dropout is applied to traditional pruning methods). In the current discussion of multi-task learning, the research on pruning is not popular, so this article also has a certain reference significance for the optimization of multi-task learning.

2 Methodology

The methodology used in this article can be divided into the following steps. First, the data is preprocessed with feature selection using genetic algorithm. Since the tasks performed by Yao et. al (2020) [14] are different from those of this article, we will then design a basic multi-task learning model as the baseline of this article. After that, the two methods proposed by Gedeon [2,3], and a third method using dropout will be implemented separately, and the results of the three models will be compared.

2.1 Data Preprocess with Feature Selection

The dataset has been introduced above. Although the features have already been extracted through ResNet, in order to avoid the impact of different feature dimensions on the model training, this study carried out max-min normalization on the data. This ensures that the scale of each feature is unified under the premise that its distribution is not affected. In addition, the ground truth data of vehicle type ID and vehicle color ID are added to the end of each corresponding piece of data. Finally, since the original data set lists the feature data of each picture as a separate npy file, in order to facilitate file reading, the merge operation is used, and three files respectively representing train set, validation set and test set are generated, with the ratio of approximately 6:2:2.

Feature selection is widely used in the field of machine learning because it can not only reduce the computational complexity by dimensionality reduction, but also prevent overfitting. In 2006, Huang and Wang perform genetic algorithms for feature selection, and still achieve high accuracy on 11 real datasets after dimensionality reduction [5]. This article attempts to reduce the 2048 features in the dataset to 1024. Each chromosome consists of 2048 binary values, and its total is 1024. We use Euclidean distance to measure the degree of discrimination between two features, that is, the greater the distance, the greater the degree of discrimination between the two vectors. And in order to evaluate the fitness of the retained feature, we also need to calculate the distance between the retained feature vector and the vectors represented by all abandoned features, and sum them; moreover, for an entire chromosome, its fitness is the sum of the fitness values of the 1024 retained features. In order to facilitate the observation of the evolutionary trend, the fitness value we used is the sum of the above distances divided by 1024*1024 (because the summing operation is performed twice). Since a larger fitness value indicates that the feature combination is more distinguishable from other features, they are considered better individuals.

In addition, in order to better test the generated chromosomes, we set the random seed to 1000. We use single point crossover and set the crossover rate to 0.8. The mutation strategy we use is shuffling indexes, that is, each gene has a 5% probability of changing position. The probability of chromosome mutation is set to 0.2. Since tourism selection usually has a faster convergence rate, it is used here. Three individuals are randomly selected from the population, and the individual with the highest fitness is selected to enter the new population, and repeat this procedure until the population is the same as the previous generation. Finally, we set the population as 20 and process for 100 generations. The result is shown in Figure 2. It can be seen that at about 80 generations, the maximum fitness we emphasis tends to be stable. This means that the algorithm has converged, and the best individual we obtained has a higher degree of discrimination from other unselected features. In 2.2, we will tune the hyper-parameters of the designed baseline model with the selected features and compare the performance using the same model trained by the original features to observe the result of feature selection.

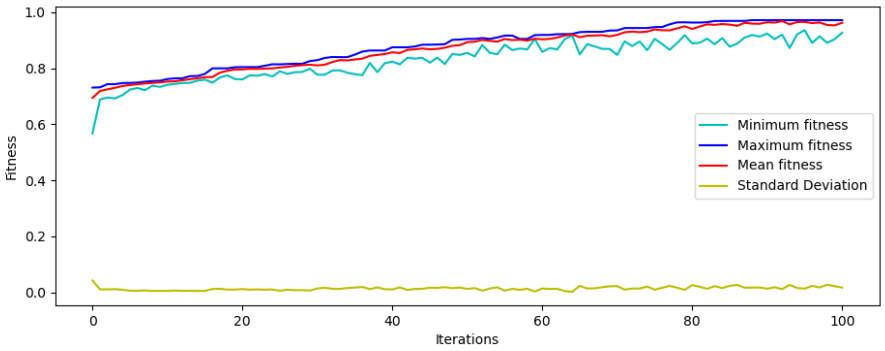


Fig. 2. Feature Selection Process with Genetic Algorithm

2.2 Baseline Model Design

The multi-task learning neural network contains a total of 3 shared layers (one input layer, two hidden layers) and one output layer for each classification task. Task 1 (classify vehicle types) has 11 output values and task 2 (classify vehicle colors) has an output size of 12. The weights of the two tasks are 50% respectively. Since the input layer has

2048 features, which tends to cause overfitting, so the learning rate is set to be small. In order to help accelerate the convergence and optimize the training effect, the Batch Normalization layer that can directly act on the loss function [10] is used. In order to make the distribution of the output tensor more stable, batch normailization layers are all added before the activation layer. (Ioffe and Szegedy, 2015) [6] In addition, since the input size is 1024, in order to avoid excessive reduction of the number of neurons between the two layers of neural networks, the number of neurons in each layer of the neural network in this model is either 1/2 or 1/4 of the previous layer. Moreover, since constructing the base model also needs to consider the pruning method used later, having more neurons in the model can increase the possibility of similar neurons. Therefore, the number of neurons in the first hidden layer is set to 512. The optimizer used is SGD. In addition to the number of neurons in each layer mentioned above, other adjusted parameters include mini-batch size, learning rate and L2 Penalty. This report first adjusts the learning rate, and then adjusts the mini-batch size and L2 penalty after obtaining a suitable learning rate. Among them, an excessively large mini-batch size will greatly slow down the training speed, and although regularization can help solve the problem of overfitting, the effect is not strong in this model. Table. 1 selects several typical settings for display, and indicates the number of epochs with their best performance. After considering overall speed and comprehensive accuracy, the bolded setting is selected. In addition, in order to judge the effect of feature selection based on genetic algorithm, we use the selected parameters and the same model, and use the original dataset without dimensionality reduction to train. The results obtained are also shown in Table 1. It can be seen that the feature selection we use reduces the dimensionality while also ensuring the accuracy.

Table 1. Parameter Settings of the Baseline Model

Features	Epoch	Mini-batch Size	Learning Rate	L2 Penalty	T1 Accuracy	T2 Accuracy
1024	50	256	0.01	0	35.7%	55.9%
	400	**256**	**0.001**	**0**	**37.4%**	**55.8%**
	600	256	0.0005	0	37.3%	55.8%
	650	512	0.001	0	36.6%	55.3%
	50	256	0.001	0.1	33.4%	32.9%
2048	400	256	0.001	0	34.2%	56.1%

Note: T1 and T2 refers to Task1 and Task2 respectively.

Fig. 3 shows the training loss and validation loss under this setting. It can be seen that the validation loss of the model has converged. This shows that the accuracy under this setting is the best at 400 epochs. Therefore, the following analysis will be carried out on the basis of this setting and the number of epochs.

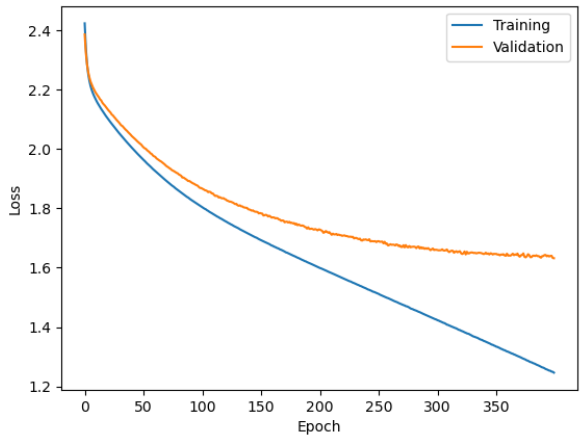


Fig. 3. Average Training Loss and Validation Loss for two Tasks of Baseline Model

2.3 Apply Neuron Behaviour and Weight Matrix

The realization of Neuron Behaviour is based on the distinctiveness property of neuron. Geodon and Harris (1991) [3] believe that the neuron output activation vector can represent the functionality of each neuron in the pattern space, which can determine the distinctiveness of the neuron. By calculating the angle between every two vectors (ranging from 0-180 degrees), the similarity of the two neurons can be judged. The standards given by Geodon and Harris are as follows: If the angle between two neurons is less than 15°, they can be considered to be extremely similar; and if the angle is greater than 165°, then they are considered complementary. For two similar neurons, the weight of one neuron will be added to the weight vector of the other neuron, and the neuron will be deleted. The two neurons of the complementary will be deleted together. In this method, the output activation vector representing the distinctness of the neuron is activated by the sigmoid function, so it will be normalized into [0, 1]. When implementing this method, this article limits the pruning time, ie in the entire model training process Pruning is not performed in the first 1/3 of the stage. In the middle 1/3 stage, pruning is performed every 20 epochs, and pruning will only be performed on the premise that the loss continues to decrease. During the rest 1/3 epochs, the model will continue training without pruning until convergence.

The implementation of Weight Matrix isvery similar to Neuron Behavior. There are two main differences: First, the Weight Matrix does not compare the activation vector, but directly compares the weight vector of the two neurons; second, it uses the weight matrix of the trained model for calculation and comparison, and prunes The latter weight matrix isused to retrain the model, so it is a static and computationally cheaper method.

It is worth noting that when implementing the two methods, in order to compare with Dropout more intuitively, this article does not choose to directly delete neurons that need to be pruned, but uses a zero mask similar to Dropout, and sets the weight of pruned neurons to zero. In addition, the algorithm also avoids repeated calculations of the same pair of neurons, including pruned neurons that use zero mask.

2.4 Apply Dropout

As emphasized above, although Dropout has an effect similar to pruning, it is different from traditional pruning methods. In each forward propagation, each neuron has the same probability of being discarded. Therefore, different neurons are trained in each iteration, and the final model can be regarded as a combination of several models. [12] This article uses the BatchNorm layer, so when using Dropout, it can be placed after the BatchNorm layer to get better results. [1] (Chen et. al, 2019) They combine Batch Normalization and Dropout in their Independent Component Layer and found that it can reduce the interaction between two neurons and thus speeds up the convergence speed of the network and brings a more stable training process.

In addition, in order to better compare with the previous two methods, this article only sets the dropout layer after the hidden layer that has been pruned. In this research, that is the second hidden layer. Moreover, for the dropout layer, the probability of discarding neurons is set according to the ratio of pruned neurons in the first two methods. For example, if the first two methods discarded 20% of neurons, the probability of discarding neurons in this Dropout layer is also set to 20%.

3 Results and Discussion

3.1 Effectiveness of Neuron Behaviour and Weight Matrix

Two pruning methods, Neuron Behaviour and Weight Matrix, are performed on the baseline multi-task leaning model. Here we also try different angle thresholds for the two methods. That is, the maximum angle at which the two vectors are considered similar is gradually increased from 15° to 30°, and the minimum angle at which the two vectors are considered to be complementary is gradually dropped from 165° to 150°. This step is to obtain different pruning ratios by controlling the angle threshold, so as to set the dropout rate and then to compare. Table2 lists the accuracy and training time of the two methods.

It can be seen from the table that with the following four angle threshold settings, the two methods have better training effects on the two tasks than the baseline model. The following two points can also be summarized from the results of the two methods. First, a smaller angle threshold (such as 15°, 165°) can usually bring better accuracy; Second, the Weight Matrix' performance is better than Neuron Behaviour, and under the condition that the weight matrix has already been obtained from the pre-trained model, the Weight Matrix is significantly faster than Neuron Behaviour (it only takes 2/3 of the time used by Neuron Behaviour). The reasons for these two points will be analyzed and discussed below.

Table 2. Pruning Results of Neural Behaviour and Weight Matrix

Method	Angle Threshold	Neurons Pruned	Pruning Ratio (%)	T1 Accuracy (%)	T2 Accuracy (%)	Time Used (s)
Neuron Behaviour	(15,165)	22	17	36.8	53.4	1400.7
	(20,160)	26	20	36.3	53.8	1410.9
	(25,155)	47	37	35.6	53.0	1321.1
	(30,150)	54	42	36.4	53.1	1340.9
Weight Matrix	(15,165)	30	23	39.4	59.3	1043.8
	(20,160)	39	30	39.1	53.8	934.2
	(25,155)	68	53	39.2	58.4	954.1
	(30,150)	73	57	39.0	57.8	860.5

For the angle threshold, it can be seen that after increasing to 30°, the two methods can almost delete half of the neurons. This means that half of the neurons have high repetitiveness or complementarity. In this regard, my guess is that because the features in this dataset are extracted through ResNet, so although the features have already been reduced by genetic algorithm, there might still be strong linear relationship between features. Therefore, only a limited number of neurons can achieve a certain degree of accuracy in these two classification tasks. Based on the above assumptions, we think that within a certain threshold, more neurons can enrich the details of the model and increase the probability that the model is correct. This is also the reason why the more neurons are retained in these angle thresholds, the higher the accuracy rate. Perhaps (15, 165) is not the optimal angle threshold for this model, but this is not the focus of this article, so there is not much discussion here. Then why can pruning within an appropriate threshold improve performance? This will be discussed in the next paragraph.

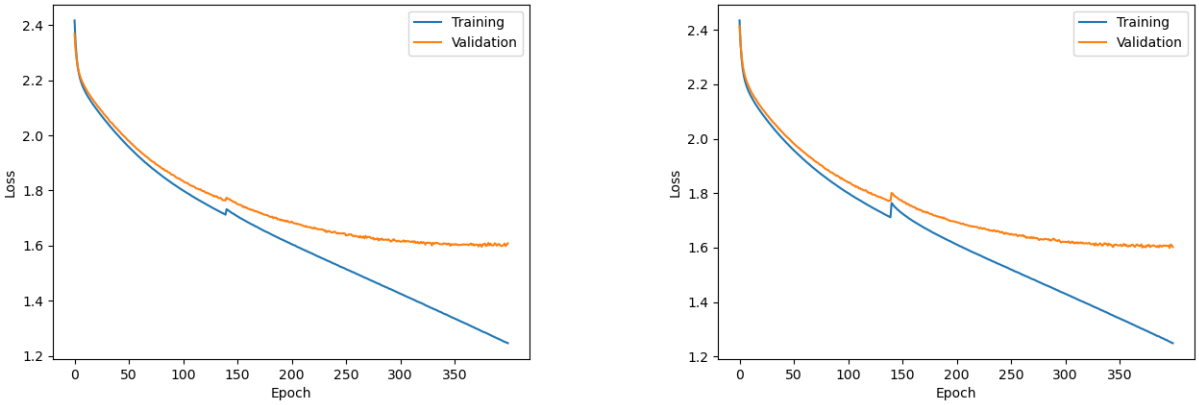
As mentioned in Methodology, due to the reason that Neuron Behaviour does not always perform pruning throughout the training process, but performs pruning operations in the middle 1/3 stage, it can be seen from Fig. 4 that regardless of the angle threshold, it is at 1/3 epochs Locally (around 130 epochs here), loss has a sudden increase, and the increase in loss usually represents a decrease in accuracy. And, the smaller the angle threshold, the smaller the increase in loss. This is because the angle threshold is small and there are fewer pruned neurons. These two methods are better than the baseline model, because pruning reduces the possibility of the model entering the local optimal solution by deleting duplicate or complementary neurons, and the increase in training accuracy is therefore slowed, thereby alleviating the over-fitting Happening. Different from the conclusion of Gedeon (1995) [2], in this model, Neuron Behaviour is not as effective as Weight Matrix. The possible reason is that in this study, pruning was performed after 1/3 of the model training, and this model has more features, which makes the possibility of overfitting is relatively higher. It can also be seen from Fig. 4 that when the validation loss stabilizes, the gap between the training loss and the validation loss becomes larger and larger. This also means that the model may be overfitted during the training phase. For Neuron Behaviour, although it is pruned at 1/3 period and continues to train, if the model has been over-fitted before, the impact of overfitting could still be left in the later training. For Weight Matrix, it is a model that is retrained with the weight matrix of the pre-trained model known. Therefore, it can reduce the impact of overfitting to a greater extent in the early stages of retraining.

3.2 Traditional Pruning versus Dropout

From the results of the above two methods, take two values with similar pruning ratios. Take 0.20 and 0.42 from Neuron Behaviour, and take 0.23 and 0.53 from Weight Matrix. These two values are approximately 0.2 and 0.5 respectively. After the second hidden layer processed by the above two pruning methods, the dropout layer is added, and the dropout rate is set to 0.2 and 0.5 respectively. Table 3 shows the results.

Table 3. Pruning Results of Neural Behaviour and Weight Matrix

Technique	Pruning/Dropout Rate	T1 Accuracy (%)	T2 Accuracy (%)	Epoch Number	Time Used (s)
Dropout	0.20	37.1	56.9	400	851.5
Neuron Behaviour	0.20	36.3	53.8	400	1410.9
Weight Matrix	0.23	40.4	61.0	400	1044.3
Dropout	0.50	38.8	57.3	600	1336.6
Neuron Behaviour	0.42	36.4	53.1	400	1581.9
Weight Matrix	0.53	39.3	60.1	400	1026.4



a. Threshold: (15, 165) b. Threshold: (30, 150)

Fig. 4. Sudden Increase in Loss of 2 Angle Thresholds

It can be seen from Table 3 that a higher dropout rate brings a better convergence limit [1], i.e., when the dropout rate is 0.2, it only takes 400 epochs to converge, but after increasing the dropout rate to 0.5, It takes 600 epochs to converge. A better convergence limit means that the model has a higher upper limit. And because the duration of the validation loss decrease is longer, the impact of overfitting is also reduced to a certain extent. But this also means longer training time. In the case of similar pruning/dropping rates, Weight Matrix has the best effect, and the effect of Dropout is also stronger than Neuron Behaviour. We think that continuing the discussion in 3.1, this is because of the model used in this study, the Weight Matrix is the least affected by the model's overfitting, and it has directivity, that is, the discarded neurons are considered the most useless, so It works best. Dropout is affected by overfitting similarly to Weight Matrix, but it drops neurons more randomly. Neuron Behaviour performs relatively poorly in this model. The main reason may be that it does not start pruning at the early stage of training in order to reduce the complexity of the operation, so it introduces more over-fitting effects.

Table 4 lists the best settings of the three techniques. Compared with the baseline model, they both have higher accuracy for the two classification tasks. They all achieved the best performance while discarding the least invalid neurons. And, on the whole, Weight Matrix has the best performance in this model, both in terms of accuracy and time consumption.

Table 4. Pruning Results of the 4 Models

Technique	Pruning/Dropout Rate	T1 Accuracy (%)	T2 Accuracy (%)	Epoch Number	Time Used (s)
Dropout	0.50	38.8	57.3	600	1336.6
Neuron Behaviour	0.20	36.3	53.8	400	1410.9
Weight Matrix	0.23	39.4	59.3	400	1043.8
None (Baseline)	0	37.4	55.8	400	992.4

4 Conclusion and Future Work

This article is based on the two directional pruning techniques proposed by Gedeon and Harris (1991) [3] and Gedeon (1995) [2], and compares them with the recently popular Dropout method of randomly discarding neurons. This article uses VehicleX data set [14] to establish a two-task learning model, the two classification tasks both have more than ten classes. At the same time, in order to reduce the computational complexity and reduce the degree of overfitting, this paper uses the genetic algorithm to halve the 2048 features. By comparing the training performance of the original data set on the same model, it is found that the result of this feature selection is satisfactory. However, during the training ,this paper finds that the model still has strong overfitting. Under this premise, all three techniques have a good inhibitory effect on overfitting. After comparison, Weight Matrix has the best effect because it is less affected by overfitting during training and is directional. Although the Neuron Behaviour used in

this article considers the dynamic changes of neurons, it is affected by the overfitting to the most extent, so the final performance is the worst. Dropout's performance is somewhere in between. It can effectively reduce the influence of overfitting like the Weight Matrix, but the results show that since it deletes neurons at random, its performance is not as good as the Weight Matrix, and as it has better convergence limit, it requires more time to train. Overall, Weight Matrix performs best on this model due to its highest accuracy and the least training time.

This research has certain limitations. First of all, the model constructed in this article is only used on one data set, so it may not have universal applicability. In this regard, other data sets could be tried; or we could use only a part of the features in this data set to observe whether the phenomenon of overfitting is reduced, and try these methods for such a model. In addition, because this article focuses on the horizontal comparison of the three methods, the same hyper-parameters are used for different methods. In fact, because different methods may prefer different hyper-parameter settings, it is possible to set different hyper-parameters for these methods and compare them longitudinally.

In addition, the Neuron Behaviour used in this article considers the calculation time and the natural convergence of the model in later training period, only the pruning operation is performed in the middle stage. We can try to advance the pruning operation or cover the entire training process to see if better results can be achieved.

This article uses two binary classification task models. Different combinations such as classification and regression, regression and regression could be tried, or more tasks could be applied in the model to observe the impact of pruning for a more complicated model. In the application of the multi-task learning model, the weight setting of the loss of the two tasks is also a very important influencing factor. This article only sets the weight of the two losses to 50% respectively. Kendall, Gal and Cipolla (2018) [7] propose to introduce the Bayesian framework from the perspective of prediction uncertainty, and automatically set weights according to the current size of each loss; Sener and Koltun (2018) [11] also propose to construct the Pareto of all losses to obtain a combination of multiple hyper-parameters in one training period, which are interesting findings worth trying. Moreover, this article uses genetic algorithm to perform feature selection, and you can also try to optimize hyperparameters. Another follow-up study may be to investigate the relationship between the best feature selection ratio and the best pruning ratio.

References

1. Chen, G., Chen, P., Shi, Y., Hsieh, C.Y., Liao, B., Zhang, S.: Rethinking the usage of batch normalization and dropout in the training of deep neural networks (2019)
2. Gedeon, T.: Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour. In: Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems. pp. 26–29 (1995). https://doi.org/10.1109/ANNES.1995.499431
3. Gedeon, T..H.D.: "network reduction techniques. In: Proceedings International Conference on Neural Networks Methodologies and Applications, AMSE, San Diego. vol. 1, pp. 119–126 (1991)
4. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding (2016)
5. Huang, C.L., Wang, C.J.: A ga-based feature selection and parameters optimizationfor support vector machines. Expert Systems with Applications **31**(2), 231–240 (2006). https://doi.org/https://doi.org/10.1016/j.eswa.2005.09.024
6. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift (2015)
7. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics (2018)
8. LeCun, Y., Denker, J., Solla, S.: Optimal brain damage. In: Touretzky, D. (ed.) Advances in Neural Information Processing Systems. vol. 2. Morgan-Kaufmann (1990)
9. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning (2019)
10. Santurkar, S., Tsipras, D., Ilyas, A., Madry, A.: How does batch normalization help optimization? (2019)
11. Sener, O., Koltun, V.: Multi-task learning as multi-objective optimization (2019)
12. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014)
13. Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., Van Gool, L.: Multi-task learning for dense prediction tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence p. 1–1 (2021). https://doi.org/10.1109/tpami.2021.3054719, http://dx.doi.org/10.1109/TPAMI.2021.3054719
14. Yao, Y., Zheng, L., Yang, X., Naphade, M., Gedeon, T.: Simulating content consistent vehicle datasets with attribute descent. In: ECCV (2020)