Model optimization strategies based on deep neural networks Learning and application of pruning optimization algorithms

To cite this article: Ni Ni and Shaoshao Xu 2022 *J. Phys.: Conf. Ser.* **2303** 012033

View the article online for updates and enhancements.

PRIME
PACIFIC RIM MEETING
ON ELECTROCHEMICAL AND SOLID STATE SCIENCE
HONOLULU,HI
October 6-11, 2024
Joint International Meeting of
The Electrochemical Society of Japan (ECSJ)
The Korean Electrochemical Society (KECS)
The Electrochemical Society (ECS)
Early Registration Deadline:
September 3, 2024
MAKE YOUR PLANS
NOW!

Model optimization strategies based on deep neural networks Learning and application of pruning optimization algorithms

Ni Ni[1, †], Shaoshao Xu[2, *, †]

[1]Hebei University of Technology, Tianjin Province, China 300000;

[2]Xiamen University, Malaysia, Sepang, Malaysia, 43900;

[*]Corresponding author: EEE1909252@xmu.edu.my

[†]These authors contributed equally.

Abstract -Deep learning is becoming increasingly important in the context of machine learning, with notable performance achieved in many experiments and attempts. However, if one wants to port a deep learning network from a computationally powerful CPU platform to a small embedded mobile device, it will be constrained by various aspects such as power consumption storage. This paper addresses this problem by simplifying the model, i.e. by model compression. Specifically, by collating work on compression pruning techniques from recent years, the importance of the parameter weights of the data model in the training phase is analysed and then the model is tested with new data in the inference phase to reduce the parameter weights and avoid over-parameterisation, thus achieving model optimisation. Ultimately, an effective idea for model pruning was derived: using the absolute magnitude of the parameters and feature outputs to measure their importance, while analysing the effect of parameter clipping on the loss function. This paper provides a summary distillation of relevant model compression work in recent years to provide a reference for research enthusiasts in the field of pruning and to pave the way for further research.

1. INTRODUCTION

In recent years, with the rapid development of GPUs, people have started to take a new interest in the research of deep learning. Deep learning network has been widely used in several fields, such as image processing, speech recognition, autonomous driving and so on, all of which have become hot areas of deep learning. Despite the great success of deep neural network algorithms in machine learning networks, the number of layers of the model has been increased and the number of parameters have grown exponentially in order to improve the performance of the model. This has also become a major limiting factor when porting the models to mobile devices. The models should not only have good performance in scientific research, but also be able to be ported to embedded devices with less computational power, for applications in the military as well as in all aspects of human life. Therefore, how to perform sparse operations on a convolutional neural network model so that it can be applied to mobile devices while keeping the performance of the model constant or less variable has become a hot issue developing rapidly.

2. MODEL PRUNING

What is model pruning? As the name implies, pruning means cutting something, i.e. cutting connections or neurons, removing some connections and corresponding neurons in a convolutional neural network whose weights are less than a certain threshold. This is of course only one strategy of pruning. As pruning developing, pruning strategies with different granularity and guidelines have emerged. Generally speaking, a pruning structure has three parts:

First part: Training a densely connected model.

Second part: Cutting out the connections or neurons.

Third part: The remaining neuronal connections are refined and iterate on the operation. Eventually a more sparse and performance-satisfying model is obtained.

Usually for the pruning of the connection is a very irregular operation. It is generally necessary to maintain a matrix equal to the dimension, which is the mask matrix. The position of the mask matrix corresponding to 0 is the weight to be clipped. The mask matrix corresponds to 1 position for the weight to be retained.

2.1 Redundancy of Convolutional Kernel Filters

In paper "Understanding and Improving Convolutional Neural Network via Concatenated Rectified Linear Units", it is proposed that there is phase complementarity in the convolutional kernel. One convolutional kernel is equal to the other convolutional kernel multiplied by negative one, and the two convolutional kernels are linearly related. That is, filter redundancy. Assuming that the jth filter is φ_j, which is a vector with directions. $\varphi_i = \mathrm{argmin}_{\varphi_j} < \varphi_i,\ \varphi_j >$ This formula can be used to find the inner product of the ith filter and the jth filter. If the phases are opposite, the product is minimal (modular normalization), which satisfies the phase complementarity mentioned earlier. Next, a cosine similarity is defined. $\mu_i^{\varphi} =< \varphi_i, \overline{\varphi_i} >$, φ_i is a filter. $\overline{\varphi_i}$ is the filter from the remaining filter that has the smallest cosine similarity value to φ_i. If for each φ_i one can be found a $\overline{\varphi_i}$, such that the cosine similarity of the calculation is minimal and close to -1, the whole curve is closer to the left side of 0 after fitting. This means that the easier it is to find a filter with the opposite phase. The complementary characteristics of shallow filters in the network are more obvious.

In order to make the redundancy more convincing, the authors define the concatenated rectified linear unit (CReLU) $\mathrm{crelu}(x) = (\mathrm{relu}(x), \mathrm{relu}(-x))$. The convolutional layer output and the convolutional layer output feature vector multiplied by the negative one are stitched together after the activation function, which is equivalent to doubling the number of channels, and without adding additional parameters, replacing the ReLU unit with CReLU for one or more layers. Through experiments, it was found that the error rate was reduced.

Experimental results obtained on the CIFAR dataset

Table 1 Test set recognition error rates on CIFAR-10/100 using deeper networks.[1]

CIFAR-10			
Model	**Single**	**Average**	**Vote**
VGG	6.35	6.90±0.03	5.43
(conv 1)	6.18	6.45±0.05	5.22
(conv1,3)	5.94	6.45±0.02	5.09
(conv1,3,5)	6.06	6.45±0.07	5.16

CIFAR-100			
Model	**Single**	**Average**	**Vote**
VGG	28.99	30.27±0.09	26.85
(conv 1)	27.29	28.43±0.11	24.67
(conv1,3)	26.52	27.79±0.08	23.93
(conv1,3,5)	26.16	27.67±0.07	23.66

In order to exclude the effect of increasing the number of channels, the authors conducted a set of comparative experiments in which the number of channels was reduced by one-half before splicing, so that the number of pre-learned channels became one-half of the previous number and the number of parameters was reduced.

Table 2 Test set recognition error rates on CIFAR-10/100.[2]

Model	**CIFAR-10**				**CIFAR-100**				**Params.**
	Single		**Average**	**Vote**	**Single**		**Average**	**Vote**	
	train	**test**			**train**	**test**			
Baseline	1.09	9.17	10.20±0.09	7.55	13.68	36.30	38.52±0.12	31.26	1.4M
+(double)	0.47	8.65	9.87±0.09	7.28	6.03	34.77	36.73±0.15	28.34	5.6M
AVR	4.10	8.32	10.26±0.10	7.76	19.35	35.00	37.24±0.20	29.77	1.4M
CReLU	4.23	8.43	9.39±0.11	7.09	14.25	31.48	33.76±0.12	27.60	2.8M
+ (half)	4.73	8.37	9.44±0.09	7.09	21.01	33.68	36.20±0.18	29.93	0.7M

Compare with other models

Table 3 Comparison to other methods on CIFAR-10/100.[3]

Model	**CIFAR-10**	**CIFAR-100**
(Rippel et al., 2015)	8.60	31.60
(Snoek et al., 2015)	6.37	27.40
(Liang & Hu, 2015)	7.09	31.75
(Lee et al., 2016)	6.05	32.37
(Srivastava et al., 2015)	7.60	32.24
VGG	5.43	26.85
VGG + CReLU	5.09	23.66

Findings on the larger ImageNet dataset

Table 4 Validation error rates on ImageNet.(Error rates with * are obtained by averaging scores from 10 patches.[4]

Model	**top-1**	**top-5**	**top-1***	**top-5***
Baseline	41.81	19.74	38.03	17.17
AVR(conv1–4)	41.12	19.25	37.32	16.49
AVR (conv1–7)	42.36	20.05	38.21	17.42
AVR (conv1–9)	43.33	21.05	39.70	18.39
CReLU(conv1,4,7)	40.45	18.58	35.70	15.32
CReLU (conv1–4)	39.82	18.28	36.20	15.72
CReLU (conv1–7)	39.97	18.33	36.53	16.01
CReLU (conv1–9)	40.15	18.58	36.50	16.14
CReLU (all)	40.93	19.39	37.28	16.72

This demonstrates that structural unit design is more effective at shallow layers of convolutional neural networks and less effective at deeper layers of convolutional neural networks, echoing the previous conclusion that shallow layers of convolutional neural networks are more redundant and deeper layers of convolutional neural networks learn more abstract features with less redundancy.

There are various ways to prove the redundancy of a filter, either theoretically, directly or indirectly through experimental results, as long as the redundancy of the filter is proven to show that model pruning makes sense.

Next, arguing about parameter distribution before and after pruning: Before pruning, it is a single peak, and after pruning, it becomes two peaks, and the overall proportion is reduced. The weight of the peak represents diversity. The better the diversity, the stronger the filter's ability to learn features,and the lower the overall proportion mean that the overall overfitting is less risky.

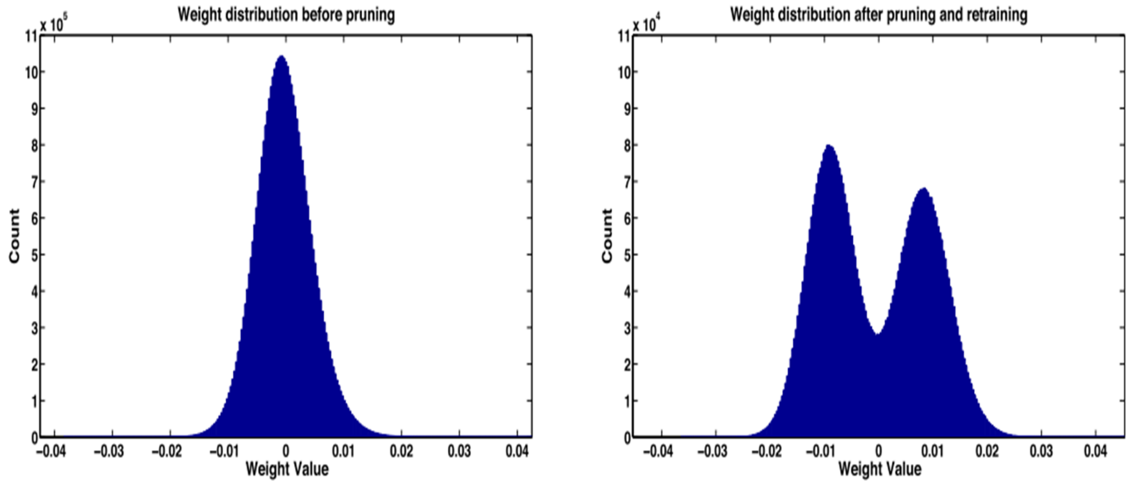

Figure 1 : Weight Distribution Before and After Parameter Pruning.
Learning both Weights and Connections for Efficient Neural Networks.[5]

2.2 Regularization and Dropout/Dropconnect

Dropout is the deletion of neurons. For a convolutional neural network, one layer of neurons corresponds to one layer of the feature map. So Dropout is a structured pruning, and Dropconnect is the de-pruning of connections between neurons, which corresponds to unstructured pruning and is more irregular than Dropout. However, both Dropout and Dropconnect are used in the training phase before testing. So they are not strictly speaking considered to be true methods of model pruning.[6]

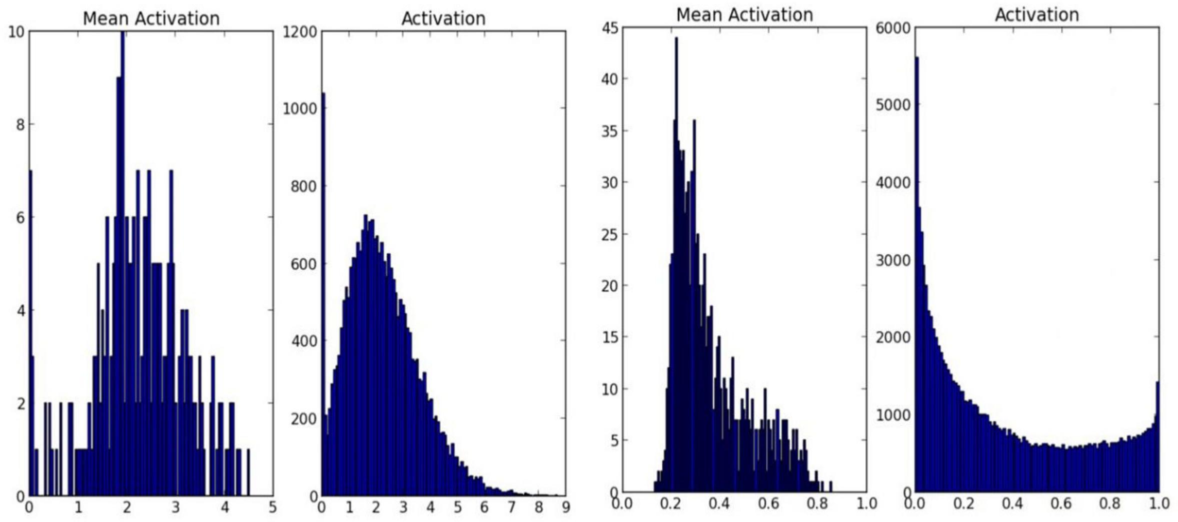

Figure 2 Effect of dropout on sparsity.[7]

The horizontal axis x is the amplitude value. From the above figure, the activation value is smaller, and the network is more sparse, and the risk of overfitting is lower.

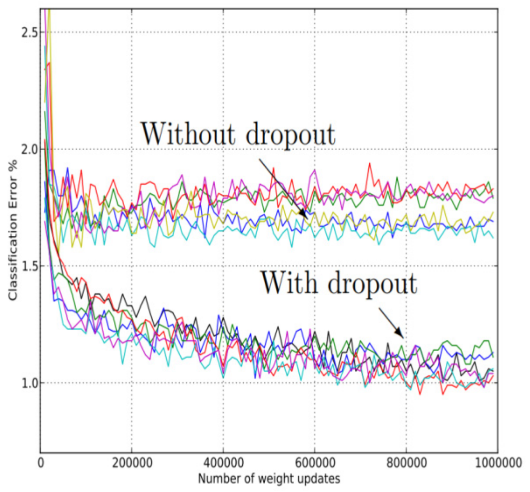

Figure 3 Test error for different architectures with and without dropout.
Dropout: A Simple Way to Prevent Neural Networks from Overfifitting.[8]

The above figure further verifies the conclusions obtained in Figure 2 and Figure 3. The abscissa is the number of updates and the ordinate is the error rate. And the error rate with dropout will be lower. So the conclusion can be summarized: Dropout is a very effective regularization, which proves the effectiveness of model pruning and the redundancy of weights.

The reason why model pruning is essentially because there is redundancy in the convolutional neural network, and deleting some of the redundant parts will not affect the performance of the model.

In general, any method that can reduce overfitting and reduce generalization error is called regularization.[9] In the neural network there are two types of parameters, w and b. W is called weight. B is called bias. W is as the unknown coefficient in the model. The value of the situation can directly determine the trend of the curve. The value of b will not affect the shape of the curve, but can only change the model translation. Therefore L1, L2 regularization is more for w. L1 and L2 regularization is actually L1 and L2 norm.

The norm is actually an extension of the concept of the distance between two points in space. W is as a point in a high-dimensional vector, and the distance from which to the origin is the L2 norm when the distance from the point of origin is Euclidean. Connecting the same points of the L2 norm, Circles with the center of the circle as the origin and radius with the L2 norm can be obtained. The L1 norm is similar to the L2 norm. Only when calculating the concrete value, the calculation method is to add the absolute value of the coordinate value. By connecting points of the same L1 norm, a square centered at the origin that has been rotated 45 degrees can be obtained.

According to the definition, there are many kinds of norms. $L_p$norm, here p can go to any number greater than 2. It is also possible to choose a number from 0-1. But only when the value of p is greater than or equal to one, the set of constituent points is a convex set, and the convex set has a good feature.

If the function is a convex function and the feasible field is a convex set, for the convex optimization problem, the regularization of L1 and L2 is to some extent using the feature of the convex set. The result of the previous hidden layer is input to the output layer, then the result is multiplied by w, and next it plus b to get a linear result. The result is then computed by a nonlinearity (softmax function), and finally the result is brought into the loss function, using a maximum likelihood estimation or cross-entropy to calculate the probability.In this process it is certain to find a set of w and b to minimize the loss function. This set of parameters is not unique. Although in the end, after training, it can converge to the maximum value of the same loss function. But the initial random values of w and b have a great influence on the final values of w and b obtained after training.

$\min J(W, b, x)$ $s.t. ||W|| - C \leq 0$， J is a loss function, which consists of three parameters: w, b, x. Parameter b determines the translation of the graph. X is the data for the training set. The focus on determining whether the model is overfitting is on the parameter w. Thus regularize the emphasis constraint w. Control parameter x by giving the parameter a range of feasible fields, and find the maximum value within the range of feasible fields.

For L1, the L1 distance from w to the origin is less than or equal to C, and this L1 distance is also called the Manhattan distance. For L2, the L2 distance from w to the origin is less than or equal to C, which is the Euclidean distance. Expressed in terms of Lagrange multiplier: $L(W, \lambda) = J(W) + \lambda(||W|| - C)$ $\min_W \max_\lambda L(W, \lambda)$ $s.t. \lambda \geq 0$, it determined the range of feasible domains is allowed to find the most valuable point under the constraint.

The L1 norm and the L2 norm constrain W, and the corresponding feasible field is a convex set, and the constraint of a convex set does not change the nature of the original problem. So the L1, L2 regularization problem under Lagrange is still a convex optimization problem. (Usually L2 regularized expressions: $L(W, \lambda) = J(W) + \lambda||W||$). With $L(W, \lambda)$ gradient of 0, W can be found. The role of the Lagrange multiplier is to adjust the gradient of the loss function and the gradient of the feasible domain range, i.e. the constraint condition. That is, the gradient size of the loss function is divided by the gradient size of the corresponding function of the constraint condition.

Since L1 regularization brings sparseness and decouples the relationship between some features and features, L2 regularization is to reduce the absolute value of the weights. Both L1 and L2 regularization constrain the weight W, but they have different effects. So the combination of L1 and L2 cannot be affected . And the addition of regularization is not much for the result deviation.

3. ANALYSIS OF CLASSIC ARTICLES ON PRUNING TECHNIQUES

The basic idea of pruning is to cut out unimportant parts. Among them, the greed algorithm is a more common method, ranking the importance and eliminating the unimportant parts. [10]So how to judge the importance. We will refer to the relevant literature from following ideas.

Idea 1: magnitude-based weight pruning: Evaluate importance by the absolute size of the parameter or feature output.

Group LASSO is used to structure the sparse weight to judge the importance by the absolute value of the parameter (or feature output).

The first paper mainly aims to solve the problem of deep neural network application on some simple devices that cannot provide high performance due to the large amount of computation, through the evaluation and comparison of structured sparse and unstructured sparse for acceleration performance and sparsity under L1 regularization of AlexNet model with GPU enabled. In this paper, a structured learning method is proposed to directly learn the group Lasso regularization of the deep neural network model.

In the next part of the thesis, the regularized generalized deep neural network model is applied to the structure sparse learning model of convolution layer. From generalization to specialization.

About experiment:

In the MNIST experiment, base on Caffe LeNet and a MLP network.The author tries to apply SSL method to LeNet model (based on MNIST)，First, filter-wise and channel-wise were carried out for all conv1 with respect to unimportant convolution kernels and channels.By comparing with baseline, extract five features: Error, Filter, Channel, Flop and Speedup for reference comparison, and draw the conclusion: reducing unimportant convolution kernels and channels not only minimizes error but also greatly reduces FLOPS and computation time. Then LeNet model carries out shape-wise based on SSL method. It is obvious that the effect of reducing FLOPS and calculation time is also significant.For multilayer perceptrons, extending SSL from the convolution layer to the full connection layer forces group Lasso regularization to be applied to the input and output of neurons, and SSL can also effectively find coefficient relations to distinguish weights.

In Experiment 2, we conducted structural pruning on ConvNet and ResNet for CIFAR-10. Reduce weight matrix’s dimension, basing on fifilter-wise and shape-wise sparsity by ConvNet in GEMM. And regularize the DNNs' depth on *ResNet's* depth-wise sparsity. In the meanwhile, as for *ConvNet* on CIFAR-10, they do row-wise and column-wise sparsity. In conclusion, SSL is performed as a structural regularization to dynamically learn better network structure (including the number of filters and the shape of filters) to reduce errors.

In Experiment 3, they use *AlexNet* with ILSVRC 2012 to evaluate SSL.AlexNet was replicated by image, the image was scaled and clipped, the data was enhanced by image, and the center clipping was used for verification. In SSL, AlexNet receives structure regularization training first. When convergent, the zero group is removed and DNN with new structure is obtained. Finally, fine-tune the network without SSL to regain accuracy.

In conclusion, this paper proposes a structured sparse learning method to regularize deep neural networks, and proves that this method accelerates effectively on both CPU and GPU with little error.

L1 regularization of the scale factor of channel pruning based on BN layer, sparse channel.

The second paper proposes a new channel-level CNN learning method that can solve the problem of reducing model size, reducing running memory, and reducing the amount of computation under the premise of ensuring model accuracy and accuracy.[11] Specifically, by performing sparsely induced regularization of the scale factor of the batch normalization layer, the unimportant channels are identified and then pruned by the scale factor of the BN layer tending to 0, so as to obtain a more compact CNN model. By pushing the value of the BN scaling factor to zero by L1 regularization, we can identify insignificant channels (or neurons) because each scaling factor corresponds to a specific convolutional channel (or one neuron in a fully connected layer). Describes the advantages of channel-level sparsity: flexible, easy to implement, and can be applied to any typical CNN model and fully connected network. Challenge: By adopting grouped LASSO, additional regularized gradients need to be calculated for all the fifilter weights.[12]Idea: Each channel corresponds to a scale factor, and the scale factor is multiplied by the output of the channel. Then the joint training weight and the scale factor are combined. Next, the scale factor is sparsely regularized, and finally the channel with a small coefficient is deleted. Based on the CIFAR and SVHN datasets, different penalty coefficients are set for VGG, ResNet, and DenseNet. And L1 regularization was performed. Experimental results: Under the condition that the accuracy of the model is basically unchanged, the network slimming can greatly reduce the parameters and effectively reduce the FLOP. In network slimming, there are two key hyperparameters: Coefficients for construction percentage and sparse regularization terms.

The second idea focuses on the effect of parameter clipping on training set error, for which the optimal brain damage technique can be considered as the originator of the research. One of the main ideas of《Optimal Brain Damage》is to remove the link between parameter magnitude and the effect of the parameter on the training set error, This property is called saliency in the article and is also loosely defined as the magnitude of the effect on the objective function brought about by the removal of a parameter. The optimal brain damage technique is specified by selecting a trained advanced network structure, using the second-order partial derivative values of the parameters as a criterion for parameter significance, cutting out the relatively insignificant ones, and iterating the above steps. In the experiment, the authors removed the penultimate layer, reducing the number of parameters by half. The training set and generalisation error increased by a tenth and 50%, while the recognition accuracy of the test set increased slightly. The later OBS technique is able to remove more parameters with the same error rate. It is still based on the Hessian matrix and uses a local model of the error surface to analyse the effect on the error caused by changes in the values of the weights. This method is applicable to all models for which a parametric gradient can be derived. Comparing the magnitude based pruning methods, OBD and OBS, the first two methods are able to remove the correct weights in some cases, while OBS is always able to remove the correct weights and does not need to retrain the model by back propagation. The impact of incorrectly removed weights using the magnitude based pruning method and the OBD technique cannot be reduced by retraining. At the same time, retraining the model after processing the two pruning methods may result in the wrong weight being set to 0 and the correct weight becoming larger (The redundant weights that should be removed are called correct weights and the weights that should be retained after pruning are called error weights). This is the opposite of the OBS procedure, where the effects caused by the removal of the wrong weights accumulate as they are removed, eventually causing a significant increase in error.

All of the above approaches incorporate pruning into the iterative optimization process, and constant pruning retraining is costly, which makes them difficult to apply to new architectures and tasks. In the artical: SNIP: Single-shot Network Pruning based on Connection Sensitivity, the effect of the connection on the variance scaling initialisation of the loss function is introduced as a saliency criterion.[13] By subtracting redundant connections before training the processed sparse network, this approach eliminates the need for pre-training and complex pruning schedules, alleviates the dependence of pruning on loss values, and is more easily applied to a variety of architectures. Such as convolutional, residual and recurrent networks, Compared to algorithms such as SWS, DNS, LC, SNIP can maintain good accuracy when the sparsity is increased to extreme levels, as shown in Table 5. It is worth noting that SNIP not only has the above-mentioned advantages, but also that it does not require additional hyperparameters or modifications to the program based on pruned objects compared to other models that have the same advantages. In summary, the core of the SNIP approach is variance scaling initialisation, which is used to ensure the accuracy of saliency measure reliable and model-agnostic. Trim irrelevant parts (e.g. the background of the image) and keep the most recognisable parts needed to identify the classification.

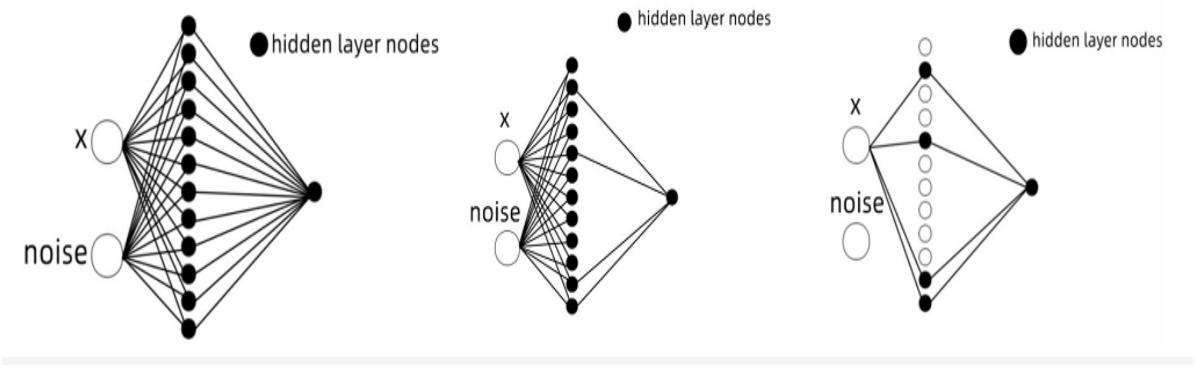

Figure 4 200 samples are generated by $\cos(x)$ and then randomly generated noise is added to the clean samples.[14]

Table 5 Pruning results on LeNets and comparisons to other approaches.[15]

Method	**LeNet-300-100 err.(%)**	**LeNet-5-Caffe err.(%)**
Ref	1.7	0.9
SWS	1.9	1.0
DNS	2.0	0.9
LC	3.2	1.1
SNIP	1.6	0.8

4. Conclusion

In order to reduce the computation and memory footprint, the method of simplifying the model is called model compression. Includes pruning, quantification, low rank decomposition, knowledge distillation. This article is mainly about the pruning part. The core idea of pruning is to cut out the unimportant parts. Since the L0 norm converts parametric clipping into an optimization combination problem, L1 and L2 regularization play an important role in this. L1 regularization introduces sparsity, removing the coupling between features and features. L2 regularization reduces the absolute value of the weight. In the interpretation of classic papers, there are two main ideas to discuss how to judge whether this part needs to be cut out. The first idea is that when the importance is judged by the absolute size of the eigenvalue, the coupling between features is sparse based on L1 regularization. The first article is about structured sparse weights based on group LASSO. The second document is about the L1 regularization of the scale factor on the channel at the BN layer. Idea 2: Consider the impact of parameter clipping on loss. The previous two articles mention that OBD and OBS measure the significance of weights based on hessian matrices. In the third paper, SNIP pruning technology has better universality than the previous two papers. And the effect of the connection on the initialization of the variance of the loss function is used as a significance criterion in the literature, which greatly reduces the computational cost and also speeds up the training speed. About the problems and future research directions of model compression: Most of the current model compression methods are only used in traditional or classical models, and further improvement techniques are needed to deal with more complex tasks.

REFERENCES

[1] Mao, H., Han, S., Pool, J., Li, W., Liu, X., Wang, Y., & Dally, W. J. (2017). Exploring the granularity of sparsity in convolutional neural networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). https://doi.org/10.1109/cvprw.2017.241

[2] Zheng, Q., Tian, X., Yang, M., Wu, Y., & Su, H. (2020). PAC-Bayesian framework based drop-path method for 2D discriminative convolutional network pruning. *Multidimensional Systems and Signal Processing*, *31*(3), 793-827.

[3] Browne, D., Giering, M., & Prestwich, S. (2020). PulseNetOne: Fast Unsupervised Pruning of Convolutional Neural Networks for Remote Sensing. *Remote Sensing*, *12*(7), 1092.

[4] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

[5] Shang, W., Sohn, K., Almeida, D., & Lee, H. (2016, June). Understanding and improving convolutional neural networks via concatenated rectified linear units. In *international conference on machine learning* (pp. 2217-2225). PMLR.

[6] Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, *28*.

[7] Zhang, M., Li, L., Wang, H., Liu, Y., Qin, H., & Zhao, W. (2019). Optimized compression for implementing convolutional neural networks on FPGA. *Electronics*, *8*(3), 295.

[8] Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. The Journal of Machine Learning Research, 20(1), 1997-2017.

[9] Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

[10] Zhao, Z., Wu, S., Qiao, B., Wang, S., & Chen, X. (2018). Enhanced sparse period-group lasso for bearing fault diagnosis. *IEEE Transactions on Industrial Electronics*, *66*(3), 2143-2153.

[11] Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6848-6856).

[12] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE international conference on computer vision (pp. 2736-2744).

[13] Lee, N., Ajanthan, T., & Torr, P. H. (2018). Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

[14] Ghiasi, G., Lin, T. Y., & Le, Q. V. (2018). Dropblock: A regularization method for convolutional networks. *Advances in neural information processing systems*, *31*.

[15] Bertocchi, C., Chouzenoux, E., Corbineau, M. C., Pesquet, J. C., & Prato, M. (2020). Deep unfolding of a proximal interior point method for image restoration. *Inverse Problems*, *36*(3), 034005.