

Identification of Weather Phenomena Based on Lightweight Convolutional Neural Networks

Congcong Wang^{1,2,3}, Pengyu Liu^{1,2,3,*}, Kebin Jia^{1,2,3}, Xiaowei Jia⁴
and Yaoyao Li^{1,2,3}

Abstract: Weather phenomenon recognition plays an important role in the field of meteorology. Nowadays, weather radars and weather sensor have been widely used for weather recognition. However, given the high cost in deploying and maintaining the devices, it is difficult to apply them to intensive weather phenomenon recognition. Moreover, advanced machine learning models such as Convolutional Neural Networks (CNNs) have shown a lot of promise in meteorology, but these models also require intensive computation and large memory, which make it difficult to use them in reality. In practice, lightweight models are often used to solve such problems. However, lightweight models often result in significant performance losses. To this end, after taking a deep dive into a large number of lightweight models and summarizing their shortcomings, we propose a novel lightweight CNNs model which is constructed based on new building blocks. The experimental results show that the model proposed in this paper has comparable performance with the mainstream non-lightweight model while also saving 25 times of memory consumption. Such memory reduction is even better than that of existing lightweight models.

Keywords: Deep learning, convolution neural networks, lightweight models, weather identification.

1 Introduction

Weather phenomenon recognition plays an important role in the field of meteorology. At present, weather radars and weather sensors have become significant means of weather identification in a target area. However, in order to better understand weather phenomenon, we need to collect more detailed data at finer resolution. As a result, this can bring substantial cost in deploying and maintaining the devices.

¹ Faculty of Information Technology, Beijing University of Technology, Beijing, 100124, China.

² Beijing Laboratory of Advanced Information Networks, Beijing, 100124, China.

³ Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing University of Technology, Beijing, 100124, China.

⁴ Department of Computer Science and Engineering, University of Minnesota, Twin Cities, USA.

* Corresponding Author: Pengyu Liu. Email: liupengyu@bjut.edu.cn.

Received: 07 March 2020; Accepted: 15 May 2020.

In recent years, deep learning technology has affected many fields [Oh, Song, Kim et al. (2019); Yu, Liu, Wang et al. (2018); Guo, Chen and Qi (2019)]. In particular, convolutional neural networks (CNNs) have made remarkable achievements in a large number of challenging computer vision tasks [Zhang and Ma (2015)]. Modern neural network models have tens or even hundreds of millions of parameters [Wang, Zhang, Zhou et al. (2019)], which can improve the performance while greatly increasing the computation cost. For the above reasons, it is difficult to deploy these models to actual weather recognition devices. On the other hand, although some lightweight models can meet the computation requirements, they often result in significant performance degradation.

In this paper, we propose a lightweight model by stacking innovative building blocks. The model can be easily applied to small devices and implemented for intensive weather monitoring. Section 2 reviews prior research on weather phenomena identification. Section 3 describes the details of our method. Section 4 presents extensive experiment to demonstrate the model's performance. Section 5 concludes our work and discusses the potential future work.

2 Related work

Different from ordinary images, weather images possess characteristics of high complexity and diversity, which bring difficulties to design proper feature extractor. Therefore, weather phenomenon identification has become a difficult problem in the field of computer vision [Lu, Lin, Jia et al. (2014)]. In order to recognize weather phenomena, some researchers adopt the traditional machine learning method. First, they segment the image and the region containing weather features only. Then, they use HOG, contrast and other features to design the classifier [Liu, Li and Wang (2017)]. However, this method requires fine image segmentation and preprocessing, which is difficult to be applied in practice. For example, weather phenomena that need to be identified sometimes occur in complex scenarios, where it is not feasible to segment weather phenomena.

Elhoseiny et al. [Elhoseiny, Huang and Elgammal (2015)] first used convolutional neural networks for weather phenomenon recognition. In this paper, a simple eight-layer convolutional neural network is constructed to identify cloudy and sunny weather phenomena. This method achieves high precision without manual feature extraction or complicated pre-processing. Since then, there have been a lot of studies using convolutional neural networks for weather recognition [An, Chen and Shin (2018)]. With the increasing depth of the network, the identifiable weather types and the identification accuracy are constantly improved.

However, more complex networks cause the difficulty in training. To train a convolutional neural network with hundreds of millions of parameters, we need at least a few hundred thousand images to avoid overfitting. To solve this problem, one approach is to adopt a transfer learning, which aims at training large models with less data by fine-tuning the models that have been trained on other data sets. In this way, it does solve some of the training difficulties, but such a large model is complicated to deploy. In addition, transfer learning largely limits the design of the model, making it difficult to transform the parts beyond the full connection layer to better fit the data.

Instead of employing more complex networks, some studies have focused on designing

lighter models to solve those problems, given the redundancy of convolutional neural networks [Li, Wang and Li (2019)]. There are two main ways to build a lightweight network. One is the 1×1 convolution module for dimension reduction, which is largely used by SqueezeNet [Iandola, Han, Moskewicz et al. (2016)] to reduce the number of parameters in the model. The other is the depth-wise convolution, which is now an integral part of lightweight networks. MobileNetV1 [Howard, Zhu, Chen et al. (2017)], MobileNetV2 [Sandler, Howard, Zhu et al. (2018)], MobileNetV3 [Howard, Sandler, Chu et al. (2019)] and ShuffleNet [Zhang, Zhou, Lin et al. (2018)] make extensive use of such modules to reduce the redundancy of neural networks. In addition, there are other ways to reduce network redundancy, such as pruning [Han, Pool, Tran et al. (2015)], quantification [Jacob, Kligys, Chen et al. (2018)], knowledge distillation [Hinton, Vinyals and Dean (2015)], etc.

In this paper, we will investigate weather phenomenon recognition with the usage of lightweight networks. Most lightweight networks tend to produce lower accuracy. To this end, the proposed model leverages both the strengths of lightweight models in reducing computing consumption and the novel structure in advanced deep learning models to ensure the performance.

3 Method details

This section will cover the specific model architecture and implementation details.

3.1 Depth-wise separable convolutions building block

For building lightweight models, one common approach is to extract feature map with the usage of depth-wise convolution. Such models have far less parameters compared with the conventional convolution models and thus have lower operational cost.

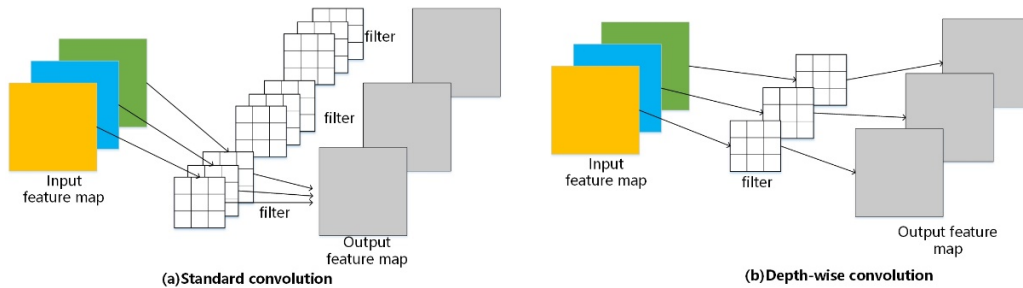


Figure 1: Standard convolution and Depth-wise convolution

As can be seen in Fig. 1, in ordinary convolution, the computation of feature map requires convolutional operations with each convolution kernel, while in depth-wise convolution, each feature graph only needs convolutional operation with its corresponding convolution kernel. Assuming we are using a convolutional kernel of 3-by-3, this structure can reduce the number of parameters by about 60%.

An important factor in determining the performance of a network is its depth [He, Zhang, Ren et al. (2016)]. The increasing depth can cause the larger perceptive field of the network, which provides a better chance at extracting more abstract image features.

Hence, we intend to use the depth-wise convolution to construct the building blocks shown in Fig. 2, and expand the depth of the network by stacking the building blocks, so as to improve the network performance.

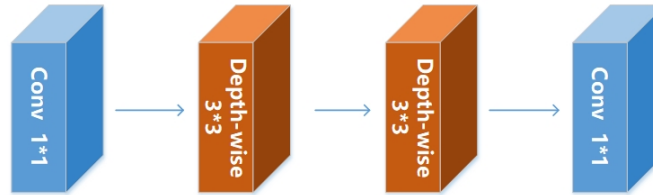


Figure 2: The building blocks

3.2 Nonlinearities

In order to obtain the nonlinear capability, the neural network model needs to add the nonlinear layer, also known as activation functions. The most widely used activation function is ReLU. An alternative to this is the swish [Ramachandran, Zoph and Le (2017)] activation function:

$$\text{swish } x = x \cdot \sigma(x) \quad (1)$$

Although the swish function has shown to outperform other activation functions (e.g., ReLU) in different test cases, the high complexity of the sigmoid operation makes it slow in computation. One solution is to replace sigmoid with an approximation of the sigmoid function. The final activation function is called the hard swish:

$$\text{hard swish} = x \left(\text{ReLU}(x + 3) \right) / 6 \quad (2)$$

3.3 Channel attention

To further improve the performance of the network, here we introduce a building block in the stacked layers. The existing convolution is carried out in two dimensions without considering the correlation between channels. Hu et al. [Hu, Shen and Sun (2018)] proposed a squeeze and excitation module (SE module) to model the correlation between channels, also known as the channel attention.

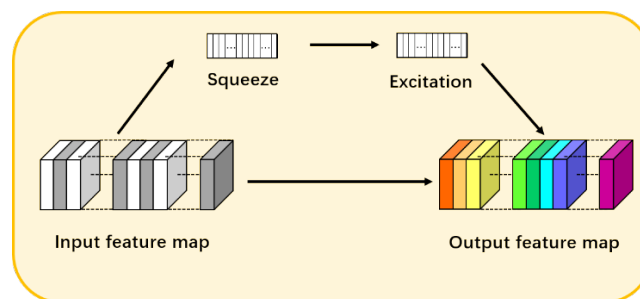


Figure 3: Squeeze and excitation

The structure of the module is shown in Fig. 3. Like an external module, the SE module assigns different weights to each channel of feature map, which is called channel

attention. The details of this approach are as follows: the feature map is compressed through the global average pooling, so that each channel only retains an averaged value as its feature, that is, squeeze. Then we can generate the attention weight for each channel through another fully connected layer after the squeeze values. These attention weights (namely excitation) are multiplied with the input feature map at each channel to obtain the final output feature map. The resulting output feature map has the same dimension as the original feature graph. However, it has different weights assigned to different channels according to their importance. By employing the channel attention mechanism, the model is capable of identifying most relevant channels while filtering out redundant and trivial information from other channels, which helps boost the performance.

3.4 Model architecture

These structures are employed to rebuild the building blocks in Fig. 4.

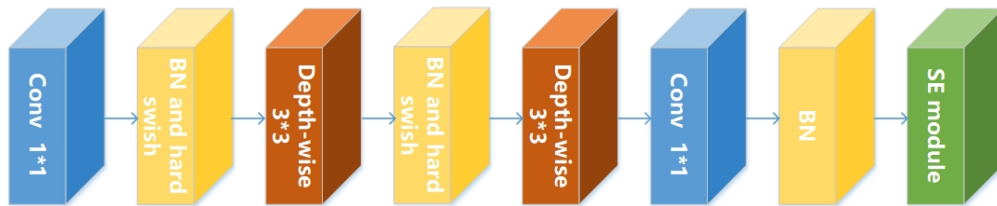


Figure 4: New building blocks

The feature extraction part is mainly composed of two layers of convolution kernel of 3×3 , and the convolution mode is depth-wise convolution. Non-linear relationships are captured more efficiently by stacking multiple convolutional layers in this model. Such capacity would directly impact the feature extraction performance.

Moreover, the feature extraction capacity of the neural network is positively correlated with the receptive field of neurons in the network. The convolution kernel of 3×3 in two layers has the same receptive field as the convolution kernel of 5×5 in one layer:

$$RF_{l+1} = RF_l + (k_size_{l+1} - 1) * f_stride_l \quad (3)$$

RF refers to receptive field, k_size refers to the size of convolution kernel, f_stride refers to the convolution stride.

Since depth-wise separable convolution is widely used in the model, a convolution kernel of 1×1 is set to change the network width to improve the performance of the model.

The nonlinear part uses the hard swish and adds a BN layer to make the network easier to converge. At the bottom of the model is an SE module with skip connect.

The detailed model structure is shown in Tab. 1. Input denotes the input feature map's size, Operator denotes the function unit, E-size denotes the expand size in building blocks, #out denotes the number of output channels, and Stride denotes the step length of convolution. The performance of the network is demonstrated in the experimental section.

Table 1: Specification for the proposed model

Input	Operator	E-size	#out	Stride
$224^2 \times 3$	conv2d	-	8	2
$112^2 \times 8$	block	16	12	2
$56^2 \times 12$	block	24	18	1
$56^2 \times 18$	block	36	24	2
$28^2 \times 24$	block	48	32	1
$28^2 \times 32$	block	64	48	2
$14^2 \times 48$	block	96	96	1
$14^2 \times 96$	conv2d	-	364	1
$14^2 \times 364$	pool	-	-	1
$1^2 \times 364$	Fc	-	6	1

3.5 Scaling

Scaling up the model is a common way to achieve better accuracy while shrinking the model tends to save more computing resources. We define a width multiplier α to scale the model. It will be shown in experiment section.

3.6 Tricks

Different training strategies have different effects on the model performance. Some work was unfairly compared with other state-of-the-art methods since the improvement were mainly from training tricks rather than methods themselves. We implement several useful training tricks for this task, which are discussed as follows.

3.6.1 Label smoothing

In image classification task, we usually use the one-hot form of label. Label smoothing is an improved form of one-hot label. It changes the one-hot label to:

$$y_{is}^i = \begin{cases} 1 - \frac{N-1}{N} \varepsilon, & \text{if } i = y \\ \frac{\varepsilon}{N} & , \text{ otherwise} \end{cases} \quad (4)$$

where ε is a small constant to encourage the model to be less confident on the training set. In this task, ε is set to be 0.1.

3.6.2 Learning scheduler

The learning rate of the optimizer is critical to the model performance, and thus a good learning scheduler is extremely important. We use the warmup strategy to change the learning rate. We spent 10 epochs linearly increasing the learning rate from $3e-4$ to $3e-3$. Then, the learning rate is decayed to $3e-4$ and $3e-5$ at 30th epoch and 60th epoch respectively.

3.6.3 Random erasing augmentation

The traditional data augmentation includes random clipping, rotation and so on. Zhong et al. [Zhong, Zheng, Kang et al. (2017)] proposed a new data augmentation approach named as Random Erasing Augmentation. For weather images, local information is irrelevant compared with global information, and random erasers can effectively avoid overfitting. The effect of random erasing is shown in Fig. 5.



Figure 5: Random erasing augmentation







4 Experiment

We compared the performance of mainstream models using migration learning with that of some lightweight models. The dataset and code are available at: <https://github.com/guhozhengling/lightweight-model-for-weather>.

4.1 Datasets

The experiment is conducted on two datasets, an open source dataset including four types of weather phenomena, which we called dataset four. Considering the lack of image types and high image recognition in the open source dataset, 12, 100 images were collected through the Internet, photography and academic exchanges. Then another dataset containing six weather phenomena including dew, freezing, haze, rain, dust and snow was constructed. Tab. 2 shows some information about this dataset.

Table 2: The dataset information

Examples of different types of images		
		
dew	frozen	haze
		
rain	dust	snow
Quantitative information		
Total number of images: 12,100		
The number of different types of images:		
dew: 1189, frozen: 660, haze: 3654, rain: 2061, dust: 1593, snow: 2934		

4.2 Experiment settings

In terms of data division, a fixed random seed was employed to randomly select 20% as the test set to ensure that the same data was used in each experiment. The experimental platform is Pytorch [Paszke, Gross, Massa et al. (2019)], the operating system is Ubuntu, and the hardware is Tesla V100.

In the setting of training parameters, the number of iterations is 100, and an early stop mechanism is set to avoid overfitting. The optimizer is Adam, the initial learning rate is $3e-4$, and the batch size is 64.

Accuracy and memory usage were evaluated against the mainstream models using transfer learning Vgg16, Vgg19, Resnet152, Densenet201, InceptionV3, and the lightweight models Squeezenet, Shufflenet, Efficientnet, and MobilenetV1-V3.

4.3 Experiment results

Tab. 3 and 4 show the performance comparison to the transfer learning models and some lightweight models respectively. Acc1 represents the accuracy on our dataset, and Acc2 represents the accuracy on the open source dataset. The performance is estimated by the accuracy and the video memory occupation. The higher the accuracy is, the less the video memory occupation is, and the better the performance of the model is.

It can be seen from the table that for transfer learning model, resnet152 achieves the best

accuracy in both datasets, but also occupies the most video memory. In comparison, the accuracy of the proposed model is 1.55% lower, but more than 25 video memory is saved. Our model is also the most competitive among the lightweight models.

Figs. 6 and 7 visually illustrate the performance of the different models on the two datasets. The horizontal axis represents the video memory occupation, and the vertical axis represents the accuracy. It can be seen that in both data sets, the proposed model is located at the upper left corner of the image. This shows that the proposed model is very competitive with other models in terms of performance.

Table 3: Comparison with transfer learning models

Model name	Acc1	Acc2	Memory usage (MB)
InceptionV3	82.97	90.95	733.33
Resnet152	92.98	96.55	829.00
Densenet201	92.61	95.26	510.66
Vgg16	89.45	95.26	735.52
Vgg19	89.66	93.07	775.68
Proposed model	91.43	96.55	32.97
Performance gap	-1.55	0.00	25.14 times

Table 4: Comparison with other lightweight models

Model name	Acc1	Acc2	Memory usage (MB)
Squeezenet	72.97	89.66	92.62
Shufflenet	89.53	95.19	60.71
Efficientnet-0	87.74	93.53	123.20
Efficientnet-1	87.00	90.95	175.33
Efficientnet-2	86.84	92.24	187.06
Efficientnet-3	87.08	93.97	250.01
MobilenetV3-large	90.89	94.40	118.06
MobilenetV3-small	90.47	93.53	40.35
Proposed model	91.43	96.55	32.97

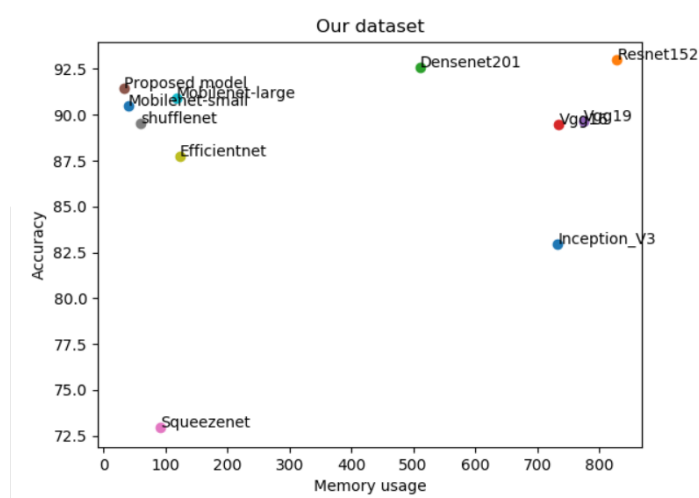


Figure 6: Performance on our dataset

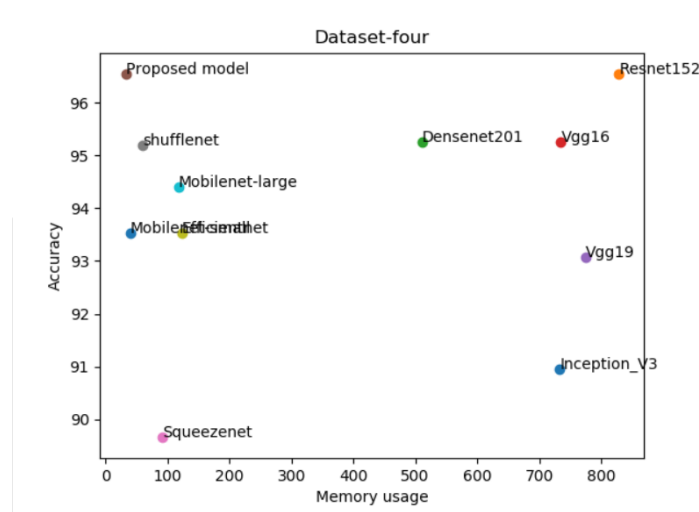


Figure 7: Performance on dataset four

4.4 Scaling

In general, a wider model often leads to performance improvements. When the model needs to be migrated to another task or has different requirements on the computing power of the device, it is more convenient to adjust the width coefficient than to redesign the model. In our experiment, the performance was further improved when the proposed model width was extended to 2.5 times. In other words, if the computing power allowed, the accuracy could be improved by consuming more video memory. Note that this is only in this task. The width coefficient needs to be adjusted to the specific situation in other tasks.

4.5 Ablation experiment

In the experiments above, we employed the best network models and tricks. In this section, we will set up some comparative experiments to verify the benefit of each component used in our proposed method for the classification of weather images.

By employing the depth-wise convolution, the model has very small parameters. In addition to the redundancy of traditional convolution, the high accuracy of such a lightweight model is also due to the use of swish activation function and the integration of lightweight SE module. As shown in Tab. 5, swish activation function replaced with the ReLU activation function and the SE module was removed respectively. It can be seen that the accuracy has decreased to a certain extent, while the parameters have only a slight reduction or no change. Tab. 6 shows the role of tricks in training. The result of no tricks will be used as the baseline, and the accuracy will be improved after adding different tricks.

Table 5: Comparison in different structure

Structure	Accuracy	Memory Usage (MB)
Without SE	90.65	31.66
Without Swish	90.88	32.97
Original	91.43	32.97

Table 6: Comparison in different tricks

Tricks	Accuracy	Promote
Baseline	89.31	-
Add Label Smoothing	90.01	0.7
Add Scheduler	90.21	0.2
Add Erasing	91.43	1.23

5 Conclusions

In this paper, a lightweight model is proposed for weather classification and a weather dataset is built. We have reviewed efficient structures and used them in our model. Through the experiment by our weather dataset, the proposed model can save 25 times memory usage comparing with the best transfer learning model with only 1.55% accuracy lost. Compared with other lightweight models, the proposed model achieves the optimal efficiency and accuracy. Also, proposed model performs equally well on other datasets. In addition, we have studied the scaling method that can easily expand the model to other tasks.

Funding Statement: This paper is supported by the following funds: National Key R & D Program of China (2018YFF01010100), National natural science foundation of China (61672064), Basic Research Program of Qinghai Province under Grants No. 2020-ZJ-709 and Advanced information network Beijing laboratory (PXM2019_014204_500029).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

- An, J.; Chen, Y.; Shin, H.** (2018): Weather classification using convolutional neural networks. *International Symposium on Computers and Communications*, pp. 245-246.
- Elhoseiny, M.; Huang, S.; Elgammal, A.** (2015): Weather classification with deep convolutional neural networks. *International Conference on Image Processing*, pp. 3349-3353.
- Guo, Y.; Chen, L.; Qi, L.** (2019): R2N: a novel deep learning architecture for rain removal from single image. *Computers, Materials & Continua*, vol. 58, no. 3, pp. 829-843.
- Han, S.; Pool, J.; Tran, J.; Dally, W.** (2015): Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems*, pp.

1135-1143.

He, K.; Zhang, X.; Ren, S.; Sun, J. (2016): Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778.

Hinton, G.; Vinyals, O.; Dean, J. (2015): Distilling the knowledge in a neural network. arXiv:1503.02531.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W. et al. (2017): Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861.

Howard, A.; Sandler, M.; Chu, G.; Chen, L. C.; Chen, B. et al. (2019): Searching for mobilenetv3. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1314-1324.

Hu, J.; Shen, L.; Sun, G. (2018): Squeeze-and-excitation networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132-7141.

Iandola, F. N.; Han, S.; Moskewicz, M. W.; Ashraf, K.; Dally, W. J. et al. (2016): SqueezeNet: alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size. arXiv:1602.07360.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M. et al. (2018): Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704-2713.

Li, Y.; Wang, Y.; Li, D. (2019): Privacy-preserving lightweight face recognition. *Neurocomputing*, vol. 363, pp. 212-222.

Liu, Y.; Li, H.; Wang, M. (2017): Single image dehazing via large sky region segmentation and multiscale opening dark channel model. *IEEE Access*, vol. 5, pp. 8890-8903.

Lu, C.; Lin, D.; Jia, J.; Tang, C. K. (2014): Two-class weather classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3718-3725.

Oh, B. D.; Song, H. J.; Kim, J. D.; Park, C. Y.; Kim, Y. S. (2019): Predicting concentration of pm10 using optimal parameters of deep neural network. *Intelligent Automation and Soft Computing*, vol. 25, no. 2, pp. 343-350.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J. et al. (2019): Pytorch: an imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, pp. 8024-8035.

Ramachandran, P.; Zoph, B.; Le, Q. V. (2017): Searching for activation functions. arXiv:1710.05941.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. C. (2018): Mobilenetv2: inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510-4520.

Wang, F.; Zhang, L. L.; Zhou, S. W.; Huang, Y. Y. (2019): Neural network-based finite-time control of quantized stochastic nonlinear systems. *Neurocomputing*, vol. 362, pp. 195-202.

Yu, R. G.; Liu, Z. Q.; Wang, J. R.; Zhao, M. K.; Gao, J. et al. (2018): Analysis and application of the spatio-temporal feature in wind power prediction. *Computer Systems Science and Engineering*, vol. 33, no. 4, pp. 267-274.

Zhang, X.; Zhou, X.; Lin, M.; Sun, J. (2018): Shufflenet: an extremely efficient convolutional neural network for mobile devices. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848-6856.

Zhang, Z.; Ma, H. (2015): Multi-class weather classification on single images. *IEEE International Conference on Image Processing*, pp. 4396-4400.

Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; Yang, Y. (2017): Random erasing data augmentation. arXiv:1708.04896.