Proceedings of the 5th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/22/20231210

Research on optimization of forest fire early warning task by neural architecture search

Junwei Bai [1,†], Kexin Sun[2,†] and Song Yue [3,4,†]

[1]Artificial Intelligence Industry College, University of Jinan, Guangzhou, 510555, China
[2]Sydney Smart Technology College, University of Northeastern, Qinhuangdao, 066000, China
[3]Computer Science and Engineering College, University of Changchun of Technology, Changchun, 130000, China

[4]1308270087@qq.com
[†]All these authors contribute equally.

Abstract. We employed Neural Architecture Search (NAS) technology in this research paper and compared it with the classic convolutional neural network structures LeNet-15 and VGG (Visual Geometry Group)16. The objective was to optimize the performance of early warning forest fire image classification tasks. We adopted a public dataset "Forest Fire Early Warning 2 Classification" from the Baidu PaddlePaddle platform, which comprises images of fires and no-fires under various environmental conditions. A convolutional neural network (CNN) model was automatically designed through AutoKeras and underwent 20 training epochs. In the experimental results, NAS outperformed with an accuracy rate of 92%, surpassing LeNet-15 (83%) and VGG16 (49%). However, its training time was longer at 33 seconds, and GPU utilization was higher, ranging from 28% to 33%. Despite the room for improvement in training time and resource utilization, NAS has proven its superiority in complex image classification tasks due to its high accuracy. Although NAS has room for enhancement in terms of training time and resource utilization, its outstanding performance in the image recognition task of early forest fire warning shows great potential for future research and application.

Keywords: NAS, early warning forest fire image classification, CNN, performance, high precision

1. Introduction

Forests are critical resources of enormous importance to the planet. It provides us with clean air, helps regulate the climate, maintains water supplies, reduces soil erosion and flooding, provides food and raw materials, and provides habitat for a wide variety of wild species. In recent years, however, the forest has been threatened by devastating wildfires frequently. The 2019-2020 Australian forest fires have burned millions of hectares of forests and killed at least 33 people. The Amazon rainforest fire in 2019 was one of the worst fires in the Amazon rainforest on record, whose smoke lingered for months and had long-term negative impacts on the ecosystem and climate change. It is distressing the scale and

DOI: 10.54254/2755-2721/22/20231210

devastation of these forest fires. Therefore, the work of protecting forests and monitoring forest fires in real-time has become extremely urgent.

Forest fires are a natural phenomenon that has occurred for millions of years. Still, the increasing frequency and severity can be attributed to various human activities, including climate change, deforestation, and land use change [1]. The effects of these fires are far-reaching, with significant impacts on biodiversity, carbon storage, and the people who depend on forests for their livelihoods and income. In addition, forest fires release large amounts of carbon dioxide and other greenhouse gases into the atmosphere, contributing to global warming.

To solve these problems effectively, advanced monitoring and management strategies are required. At present, automatic forest fire monitoring technology has become an essential means to realize real-time monitoring and early warning of forest fires to take timely measures for firefighting and rescue [2]. Sensor technology is a commonly used forest fire monitoring technology that can detect automatic fire by monitoring parameters such as temperature and smoke. However, sensor technology has a high trigger false alarm rate, which may be affected by environmental factors, resulting in unstable fire detection accuracy and reliability. In addition, sensor technology requires a large amount of equipment and manpower input, which is costly and challenging to achieve full coverage. Video monitoring technology is another commonly used forest fire monitoring technology, which can realize automatic fire detection through camera monitoring [3]. However, the effect of video surveillance technology is affected by factors such as weather and light, and false positives or false positives may occur. At the same time, video monitoring technology requires a large amount of bandwidth and storage space, which is expensive and difficult to achieve full coverage. Intelligent algorithm monitoring technology is an emerging forest fire monitoring technology that can realize automatic fire detection through the training and prediction of machine learning algorithms. Innovative algorithm monitoring technology is flexible and scalable and can be adjusted and optimized according to conditions. However, intelligent algorithm monitoring technology still needs to be mature enough and requires further research and development.

NAS(Neural Architecture Search), a technique for automating the design of neural networks, has enormous potential to have a major impact on the practicality of deep learning [4]. Currently, the technique has achieved state-of-the-art performance on several tasks. A typical NAS technique includes two phases: a search phase and an evaluation phase. In the search phase, NAS aims to find an optimal neural network architecture. This process is usually searched using reinforcement learning, evolutionary algorithms, or gradient descent. In this stage, NAS generates candidate network architectures and evaluates and screens them to find optimal ones. In the evaluation phase, NAS trains the optimal network architecture from scratch and verifies it on the test data. This process is often computationally resource-intensive and time-intensive but ensures that the final network architecture is selected with the best performance and generalization capabilities. Specifically, compared with traditional models (such as VGG, LeNet [6], etc.), the neural network architecture searched by NAS has the following advantages [5,6]:

First, higher performance: NAS can automatically design neural networks and generate highly optimized network architectures with better performance and generalization capabilities. Compared with traditional models, the network architecture searched by NAS can achieve higher accuracy and lower error rate on the same data set.

Second, smaller model size: NAS can generate a more petite model size, reducing the storage and computing costs of the model. Compared with the traditional model, the network architecture searched by NAS can reduce the number of parameters and the amount of calculation of the model while maintaining performance.

Third, better interpretability: NAS can generate a more interpretable network architecture, allowing people to better understand the network structure and feature extraction methods. Compared with the traditional model, the network architecture searched by NAS can better explain the network's decision-making process and feature extraction method.

Fourth, more vital generalization ability: NAS can generate a network architecture with a more vital generalization ability to better adapt to different data sets and tasks. Compared with the traditional

DOI: 10.54254/2755-2721/22/20231210

model, the network architecture searched by NAS can better adapt to other data sets and functions and has better migration ability.

In conclusion, the neural architecture search technique is an up-and-coming deep learning technique that can automate the design of high-performance neural networks. With the continuous improvement of computing resources and algorithms, NAS technology will be more widely used in the future.

The paper aims to explore an efficient method for forest fire monitoring based on image recognition. From the perspective of a third party, objectively compare the accuracy, versatility, training time, and other parameters of different architectures, and observe the effect of different methods on forest fire monitoring. Thus, it can be verified that the neural network architecture obtained by NAS search has higher performance, smaller model size, better interpretability, and more vital generalization ability to perform real-time monitoring tasks of forest fires better.

2. Method

2.1. Comparison of model choices

We chose the classic LeNet-15 and VGG16 network structures as benchmarks for comparison with NAS. Below are the advantages of the two models and why I chose them.

LeNet-15: LeNet is a classic convolutional neural network and the basis for many modern CNN designs. Its main advantage lies in its simplicity and efficiency [7]. The structure of LeNet is simple and training is fast, so it can serve as a very good baseline model under limited resources. Furthermore, although LeNet was designed for recognizing handwritten digits, its structure has proven effective for various image recognition tasks. Therefore, it can provide me with a benchmark to compare the performance of NAS.

VGG16 [8]: VGG16 is a deep convolutional neural network, with the advantage of in-depth and strong performance. The architecture of VGG16 reaches a depth of 16 layers, enabling it to learn more complex feature representations. VGG16's excellent performance in image recognition competitions such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has proven its strong performance. Therefore, choosing VGG16 as a benchmark model can provide a high-performance standard, letting you know the performance that NAS needs to reach or surpass.

When choosing a benchmark model, a wide-ranging model needs to be chosen, which can provide a comprehensive performance comparison. In this case, LeNet and VGG16 are good choices because they represent relatively simple and complex CNN architectures, respectively. This way, the performance of NAS can be compared with simple and complex models to get a comprehensive performance evaluation.

2.2. *Dataset selection*

We chose a public dataset called "Forest Fire Early Warning 2 Classification," provided by the Baidu PaddlePaddle platform [9]. This dataset contains 202 high-quality JPG images, of which 101 are forest fire images and the other 101 are no fire images. Both fire and no-fire images include various environmental conditions, such as day, night, different seasons, and different weather conditions, to ensure the model's wide adaptability and robustness.

We carried out several preprocessing steps on the dataset to enhance the model's learning ability and prevent overfitting [10]. Firstly, we carried out horizontal and vertical flipping, which allowed the model to learn the features of fire and no-fire images from different angles. Secondly, we also performed image rotation to allow the model to learn the rotational invariance of fire and no-fire images. All these data augmentation steps were performed using an open-source Python image processing library.

In terms of data partitioning, we chose a common 8:2 partition ratio, using most of the images (approximately 81.6%) as the training set to optimize our model parameters, and the remaining images (approximately 18.4%) as the test set to fairly assess the model's performance on unseen data. This partitioning strategy can help us ensure the model's generalization ability.

The main reason we chose this dataset is its rich and challenging content. Fire images and no fire images have obvious visual differences, but also some subtle similarities, which pose high demands on

DOI: 10.54254/2755-2721/22/20231210

the model's recognition ability. In addition, this dataset is widely considered a benchmark dataset in the field of forest fire detection, allowing us to fairly compare our model with other research results.

2.3. Model and training process

Our target model is a Convolutional Neural Network (CNN) designed automatically by AutoKeras [11]. During the model construction process, we first set the image input node, then added a normalization layer for data preprocessing, to reduce numerical instability during the model training process.

Next, we added a convolutional block, which includes multiple convolutional layers and a Dropout layer. The dropout rate is set at 0.4 to prevent the overfitting of the model. However, we did not use image augmentation as we had already applied image enhancement on the dataset during the preprocessing stage.

Lastly, we added a classification head to perform a binary classification task for forest fires. We used AutoKeras' AutoModel function to automate the training and optimization of the model, limiting the maximum number of attempts to one, to save on training time.

The model was trained for 20 epochs on the training set. After training, we exported and saved the model in TensorFlow model format. To ensure successful model saving, we also provided an exception handling mechanism - if the saving in TensorFlow format fails, the model would be saved in the HDF5 format.

We reloaded the saved model, providing custom objects from AutoKeras during loading to correctly parse the model structure. Following this, we used TensorFlow's model visualization tool to save the structure of the model as an image file, facilitating subsequent analysis and discussion.

Finally, we made predictions on the test set and printed the prediction results. The prediction results are a two-dimensional array, with each row corresponding to a sample, and each column corresponding to the predictive probability of a class. And the Structure explored by NAS is shown in Figure 1.

Input Layer: The input layer receives an input with a shape of (None, 180, 180, 3), where None can be any batch size, 180x180 is the spatial resolution of the input, and 3 represents the number of channels (e.g., RGB channels for color images). The output of this layer has the same shape as the input, which is (None, 180, 180, 3).

Cast To Float32 Layer: This layer takes the input from the previous layer and casts it to 32-bit floating-point format. The output shape remains the same as the input, (None, 180, 180, 3).

Normalization Layer: This layer normalizes the input data, which helps in stabilizing and accelerating the training process. The input and output shapes are the same, (None, 180, 180, 3).

2D Convolution Layer (conv2d): The first 2D convolution layer takes an input with shape (None, 180, 180, 3) and applies convolutional filters to it, resulting in an output with shape (None, 178, 178, 32). Here, 32 represents the number of filters used.

2D Convolution Layer (conv2d_1): The second 2D convolution layer takes an input with shape (None, 178, 178, 32) and applies convolutional filters, outputting a shape of (None, 176, 176, 32).

Max Pooling 2D Layer (max_pooling2d): This layer down-samples the input by taking the maximum value over the window defined for each dimension, effectively reducing the spatial resolution. It takes an input of shape (None, 176, 176, 32) and outputs a shape of (None, 88, 88, 32).

Dropout Layer: This layer randomly sets a fraction of the input units to 0 at each update during training time, which helps prevent overfitting. The input and output shapes are (None, 88, 88, 32).

2D Convolution Layer (conv2d_2): The third 2D convolution layer takes an input with shape (None, 88, 88, 32) and has an output shape of (None, 86, 86, 32).

2D Convolution Layer (conv2d_3): The fourth 2D convolution layer takes an input with shape (None, 86, 86, 32) and outputs a shape of (None, 84, 84, 32).

Max Pooling 2D Layer (max_pooling2d_1): Similar to the previous max pooling layer, this layer down-samples the input and has an input shape of (None, 84, 84, 32) and an output shape of (None, 42, 42, 32).

Dropout Layer (dropout_1): Another dropout layer with input and output shapes of (None, 42, 42, 32).

DOI: 10.54254/2755-2721/22/20231210

Global Average Pooling 2D Layer: This layer computes the average value of each feature map and is used to reduce the spatial dimensions. The input shape is (None, 42, 42, 32), and the output shape is (None, 32).

Dense Layer: A fully connected layer that takes input of shape (None, 32) and outputs a shape of (None, 1). It is used to perform the final classification.

Classification Head with Activation Function: The final layer takes the output from the dense layer, which has a shape of (None, 1), and applies an activation function to make the final prediction. The output shape remains (None, 1).

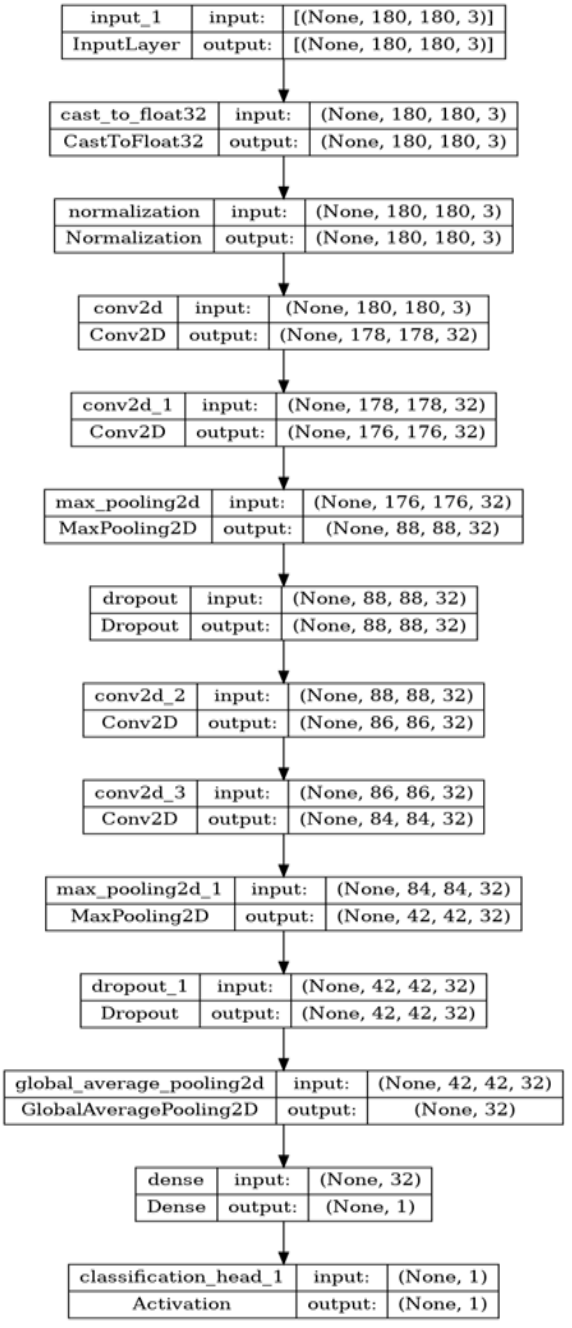

Figure 1. Structure explored by NAS.
Photo Credit: Original

DOI: 10.54254/2755-2721/22/20231210

2.4. Experimental setup and environment

All experiments were conducted on a computer with two RTX 3090 graphics cards. We used CUDA 11.2 to support parallel processing and GPU computing. All models were implemented and trained using the TensorFlow 2.0.0 framework. TensorFlow provides low-level API access, allowing us to fully utilize modern hardware capabilities [12]. In this environment, we achieved efficient training and optimization of the model.

In the experiments, we used the default TensorFlow settings unless otherwise specified. The optimizers, loss functions, and evaluation metrics we used are all explicitly described in the model training section.

3. Results

3.1. Summary of the results

This study used three neural network models: NAS, LeNet-15, and VGG16, to process and analyze a forest fire monitoring dataset. Each model was trained and tested to determine its accuracy in predicting the likelihood of fires, the loss values, and the time required for model training and prediction. Among all the tests, the NAS model performed the best in accuracy, reaching 92%, achieved under the following hyperparameter settings: (batch-size=32, dropout rate=0.4, epoch=10). However, the LeNet-15 model outperformed the other two models regarding run time, with an average run time of 22.58 seconds. The VGG16 model performed worst among these three models.

3.2. Specific experiment results

NAS (Neural Architecture Search) has an accuracy of 92%, a training time of 33.00 seconds, and a GPU utilization of 28% to 33%. And the specific experiment result is shown in Table 1.

LeNet-15 has an accuracy of 83%, a training time of 22.58 seconds, and a GPU utilization of 2% to 5%.

VGG16 has an accuracy of 49%, a training time of 24.00 seconds, and a GPU utilization of 5% to 11%.

Table 1. Specific experiment results.

	Accuracy	Training time	GPU-utilization
NAS	92%	33.00s	28%-33%
LeNet-15	83%	22.58s	2%-5%
VGG16	49%	24.00s	5%-11%

3.3. Comparison of experiment results

3.3.1. Accuracy comparison. Based on our experimental results, NAS performed the best in terms of accuracy, achieving an accuracy rate of 92%. In contrast, the accuracy rates of LeNet-15 and VGG16 were 83% and 49%, respectively. It is evident that NAS significantly outperforms the other two architectures in terms of accuracy [13-16].

3.3.2. Training time comparison. In terms of training time, LeNet-15 had the shortest average training time at 22.58 seconds, followed by VGG16, with a training time of 24 seconds. Although NAS performed excellently in terms of accuracy, its training time was the longest, at 33 seconds. This shows that LeNet-15 and VGG16 have certain advantages over NAS in terms of training time.

DOI: 10.54254/2755-2721/22/20231210

3.3.3. GPU usage comparison. In terms of GPU utilization, NAS had the highest usage, ranging from 28% to 33%, while VGG16's usage ranged from 5% to 11%, and LeNet-15 had the lowest usage at just 2% to 5%. This shows that in terms of GPU resource usage, LeNet-15 is the most efficient.

3.3.4. Conclusion. Based on our results, the following conclusions can be drawn: In terms of accuracy, NAS performs the best; in terms of training time, LeNet-15 has the shortest training time; in terms of GPU utilization, LeNet-15 is the most resource-efficient.

4. Conclusion

In our research, we applied and tested NAS technology and compared it with the classic convolutional neural network structures LeNet-15 and VGG16. The objective was to optimize and enhance the performance of early warning forest fire image classification tasks. Upon comparison and analysis, we found that NAS significantly outperformed LeNet-15 and VGG16 in terms of accuracy, reaching a rate of 92%. This indicates that NAS possesses high accuracy, effectively handling and resolving complex image classification problems.

However, NAS's training time and GPU utilization increased compared to LeNet-15 and VGG16. This is because NAS requires extensive computations to find the optimal network structure, which increases training time and resource consumption. Therefore, despite NAS's excellent performance in terms of accuracy, there is room for improvement in training efficiency and resource utilization.

In conclusion, although there are issues with longer training time and higher resource utilization, NAS's high accuracy demonstrates that it is an extremely effective method for handling image classification tasks in early forest fire warning. Therefore, NAS has the potential for further optimization and improvement in the future, and to achieve high accuracy in early forest fire warning in practical applications. For further research on NAS, we recommend focusing on improving training efficiency and optimizing resource utilization to make NAS a more practical and efficient tool.

References

[1] Rowell, A., & Moore, P. F. (2000). Global review of forest fires. Forests for Life Programme Unit, WWF International, pp. 66-66.

[2] Barmpoutis, P. et al. (2020) 'A review on early forest fire detection systems using optical remote sensing', Sensors, 20(22), p. 6442. doi:10.3390/s20226442.

[3] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho and S. W. Baik, "Convolutional Neural Networks Based Fire Detection in Surveillance Videos," in IEEE Access, vol. 6, pp. 18174-18183, 2018, doi: 10.1109/ACCESS.2018.2812835.

[4] Zoph, B., and Le, Q. V. (2016). "Neural Architecture Search with Reinforcement Learning." arXiv [Preprint] arXiv:1611.01578.

[5] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv [Preprint] arXiv:1409.1556.

[6] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learn-ing applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.

[7] Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2017) 'ImageNet classification with deep convolutional Neural Networks', Communications of the ACM, 60(6), pp. 84–90. doi:10.1145/3065386.

[8] Wei, X. et al. (2019) 'Railway track fastener defect detection based on image processing and Deep Learning Techniques: A Comparative Study', Engineering Applications of Artificial Intelligence, 80, pp. 66–81. doi:10.1016/j.engappai.2019.01.008.

[9] Li, Y., Huang, C., Ding, L., Li, Z., Pan, Y., & Gao, X. (2019). Deep learning in bioinformatics: Introduction, application, and perspective in the big data era. Methods, 166, 4–21.

[10] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1), 60.

DOI: 10.54254/2755-2721/22/20231210

[11] Jin, H., Song, Q. and Hu, X. (2019) 'Auto-keras: An efficient neural architecture search system', Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining [Preprint]. doi:10.1145/3292500.3330648.
[12] TensorFlow. (2019). TensorFlow: An end-to-end open source machine learning platform. GitHub Repository.
[13] Chen, Y. et al. (2020) 'Efficient Evolutionary Deep Neural Architecture Search (NAS) by noisy network morphism mutation', Communications in Computer and Information Science, pp. 497–508. doi:10.1007/978-981-15-3415-7_41.
[14] Truong, A. et al. (2019) 'Towards Automated Machine Learning: Evaluation and comparison of AUTOML approaches and Tools', 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI) [Preprint]. doi:10.1109/ictai.2019.00209
[15] Zhang, B., Zhou, Z., Cao, W., Qi, X., Xu, C., Wen, W.: A new few-shot learning method of bacterial colony counting based on the edge computing device. Biology. 11, 156 (2022)
[16] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift, https://arxiv.org/abs/1502.03167.