Performance comparison of different convolutional neural network models and optimizers for dog bread classification

Xinyu Mao[1,†], Mingrui Xie[2,†] and Ruochen Zhu[3,4,†]

[1]College of Information Technology, Shanghai Ocean University, Shanghai, 200000, China
[2]School of Computer Science, China University of Geosciences, Hubei, Wuhan, 430074, China.
[3]School of Optoelectronic Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, China

[4]2020050901028@std.uestc.edu.cn
[†]These authors contributed equally.

Abstract. This study explores the performance of various deep learning models including ResNet152, VGG16, VGG19 and ResNet256 on the dog breed classification task. During training, observe the loss and accuracy trends. The loss gradually decreases, showing that the model is fitting the training data better. With the improvements of the capacity of the model, the accuracy trend shows a steady increase. These models converge after about 20 epochs and fluctuate little after that. The initial learning rate, adjustment factor and patience parameters play key roles in the convergence process. However, the achieved accuracy is below 90%, suggesting that further optimization or more complex architectures may be beneficial. Among all models, ResNet512 has the highest overall accuracy (83%), followed by ResNet256 (83%), VGG19 256 (79%) and VGG16 256 (78%). The ResNet model outperforms the VGG model in most cases, probably because its network structure reduces computational complexity while maintaining accuracy. Increasing the input size can improve the accuracy of the same network structure, such as ResNet 256 and ResNet 512, while modifying the network structure by adding more layers. Learning rate decay scheduling methods, such as ReduceLROnPlateau and CosineAnnealingLR, and optimizers such as SGD, Adam, and Adagrad, are explored as well.

Keywords: dog breed classification, convolutional neural network, deep learning.

1. Introduction

Dog breed identification is a challenging task in animal recognition research. Precise identification of dog breeds helps understand their traits, needs, and behaviors. This leads to better care and reduced harm to humans. Dog breed identification technology has practical applications in conservation, breeding programs, veterinary care, legal regulations, genetic research, disease diagnosis, and biodiversity preservation. However, dog breed classification faces challenges like inconsistent lighting, occlusions, and differing poses [1,2].

Before deep learning technologies, researchers used traditional image processing techniques and machine learning algorithms for dog breed classification. They extracted image features like color, texture, and shape, and used classifiers such as SVM, decision trees, or K-nearest neighbors. These methods had limitations handling complex image data, resulting in lower classification accuracy. With deep learning technologies, convolutional neural networks (CNNs) became the leading approach for dog breed classification. CNNs autonomously extract image features and perform classification, improving accuracy and generalization compared to conventional methods. Researchers experimented with neural network architectures like AlexNet, VGG, and ResNet, achieving promising outcomes [3,4].

In dog breed classification research, scholars made significant strides using various neural network models and optimization strategies. Some studies attained increased classification accuracy on specific datasets. However, most research focuses on distinct model structures, with less emphasis on different structures and parameter training optimization methods' effects on outcomes. This study aims to offer comprehensive guidance for dog breed classification tasks. This paper examines three neural network models: ResNet152, VGG16 and VGG19, comparing their performance in dog breed classification tasks by adjusting parameters, optimizers, and learning rates [5,6]. The research uncovers valuable insights for better understanding and optimization of dog breed classification tasks.

2. Method

2.1. Dataset

This work is conducted on the Stanford Dogs dataset for experiments, which has 20,580 images with 120 various categories of dog [7]. These images are sourced from ImageNet, a large database widely leveraged for image-based researches [8]. The Stanford Dogs dataset was constructed to solve fine-grained image classification tasks. In this dataset, each category has a unique label, and each image is accompanied with the location of the dog, marked by bounding boxs. By using this dataset, this study can evaluate and compare various model performances on the dog breed classification task. The diversity and complexity of the Stanford Dogs dataset provides a challenging testing scenario for deep learning models and helps us gain insight into how various models perform when solving practical problems.

2.2. Neural network

2.2.1. VGG16. Visual Geometry Group (VGG) is developed by reserachers from the University of Oxford. It achieved high performance in the 2014 ImageNet image recognition challenge, making it one of the best-performing image recognition models at that time.

The VGG16 structure has 16 weight layers, 13 of them are convolutional layers and 3 are fully connected layers, along with 5 max-pooling layers and an output layer. VGG16 gained attention due to its simple yet effective design. Its main features are three folds. Firstly, small-sized (3x3) convolutional kernels. The convolutional layers of VGG16 use 3x3 kernels, which decrease the storage consumption and computational burden compared to other models with larger kernels. The small-sized kernels also help to learn features at multiple levels. Secondly, multiple layers of network architecture. VGG16 has a deep network structure that can effectively learn high-level features in images. This depth architecture achieved a high level of accuracy in image recognition tasks at that time. Thirdly, more convolutional and fully connected layers. VGG16 uses more layers, which enables the network to learn more complex features and patterns in images.

VGG16 remains a very classic CNN architecture, broadly used in deep learning research and education. In the following experiment, the VGG16 model pre-trained on ImageNet was used, and the classifier 6 fully connected layer was unfrozen. A neural network was constructed with two linear layers and a ReLU activation function, with 256 output of the first linear layer. After the output of the first linear layer, a ReLU activation function was used for non-linear transformation. A Dropout layer was added after ReLU to reduce the risk of overfitting. Finally, a linear layer with 120 elements was used to represent the output of the neural network and output the predicted class probability.

2.2.2. VGG19. It is another CNN architecture. Similar to VGG16, VGG19 achieved higher performance on ImageNet classification challenge. The VGG19 structure has 19 weight layers, including 3 fully connected layers, and 16 convolutional layers, along with 5 max-pooling layers and an output layer. The main features of VGG19 are similar to those of VGG16, both using small-sized (3x3) convolutional kernels and deep network architecture. This enables VGG19 to effectively learn high-level features in images as well.

The main difference between VGG19 and VGG16 lies in the number of convolutional layers, where VGG19 is deeper. This means that VGG19 has a deeper network compared to VGG16, theoretically allowing it to extract more descriptive features. However, this increased depth also results in higher computational complexity and a larger number of parameters. As a result, training and deploying VGG19 require longer training times and higher computational resources.

2.2.3. ResNet-152. ResNet-152 is a variant of the Residual Network (ResNet) consisting of 152 layers. ResNet was proposed by Kaiming He and his colleagues in 2015 to address the degradation problem in deep convolutional neural networks, where the accuracy of the model decreases as the model get deeper. By introducing residual learning, ResNet successfully solved this problem and achieved outstanding results in the ImageNet competition.

The main components of ResNet-152 are residual blocks, which enable the transfer of information across layers. Each block has several convolutions and a "skip connection" that allows the input to bypass the convolutional layers and directly reach the next layer. This helps to avoid the vanishing gradient phenomenon, allowing the network to increase its depth while maintaining high accuracy.

ResNet-152 has shown excellent performance, such as in image classification, object detection, and semantic segmentation. In fact, the ResNet architecture has become one of the cornerstones of the deep learning field, and many other network architectures have also drawn inspiration from its design. The last fully connected layer has 256, 512, and more nodes.

2.3. Optimizer

2.3.1. Stochastic gradient descent (SGD). It is a widely used optimization algorithm in deep learning [9]. It is an iterative method that updates the model's parameters in each iteration using a mini-batch. SGD aims to minimize the cost function by updating parameters towards the direction of negative gradient. The learning rate is a hyperparameter that adjusts the length of the steps taken by the optimizer. In SGD, the learning rate is constant and the same for all parameters, which can be problematic because it may converge slowly or overshoot the optimal solution.

To mitigate this issue, various modifications to the basic SGD algorithm have been proposed, including momentum, Nesterov accelerated gradient, and adaptive learning rate methods such as AdaGrad and RMSProp.

2.3.2. Adaptive moment estimation (Adam). It is an adaptive learning strategy that has gained popularity in recent years [10]. It is a combination of the ideas behind momentum and RMSProp. Adam maintains an exponentially decaying average of past gradients and their squares, and uses this information to adaptively adjust the learning rate. It also incorporates bias correction to account for the initial estimates of the moments being zero. Adam is well-suited for large-scale, high-dimensional problems with sparse gradients, such as those encountered in deep learning. However, it has several hyperparameters that need to be tuned, including the learning rate, the momentum parameters, and the epsilon value.

2.3.3. Adaptive gradient algorithm (Adagrad). Adagrad is another adaptive learning rate. Adagrad has the advantage of being relatively simple to implement, requiring only a few additional lines of code compared to basic SGD. However, it has been shown to be less effective than Adam for some problems, particularly those with large numbers of parameters. Additionally, Adagrad may converge to a suboptimal solution if the learning rate is not appropriately tuned.

2.4. *Learning rate scheduler*

2.4.1. ReduceLROnPlateau method. This method dynamically adjusts the learning rate based on the results of the validation set. When the performance on validation set stop improving for a consecutive number of epochs, the learning rate will be decreased according to previous configuration. This process continues until the learning rate reaches a minimum limit. The core idea of this method is to monitor the validation performances and reduce the learning rate if the model stops improving, to prevent the local minimum optimization.

2.4.2. CosineAnnealingLR method. This method adjusts the learning rate in the form of a cosine function to slowly decrease to a minimum value and then gradually increase back up. The method improves the training effectiveness of the model by gradually decreasing the learning rate during training and keeps the model from converging to local minimum. Specifically, the learning rate linearly decreases from the initial value to the minimum value, then gradually increases back up through the cosine function, with one full cycle lasting for T_max epochs. After the learning rate reaches the minimum value, CosineAnnealingLR repeats the cycle as needed to further decrease the learning rate.

Both methods adjust the learning rate for increasing the training effectiveness. ReduceLROnPlateau dynamically adjusts the learning rate based on the validation set results and is suitable for situations where the learning rate needs to be adjusted dynamically during training. CosineAnnealingLR improves the training effectiveness of the model by gradually decreasing the learning rate and is suitable for situations where the training period is relatively long.

3. Result

3.1. *Result of convergence*

As demonstrated in Figure 1, with the training progressed, the loss decreased gradually, indicating that the model was learning to fit the training data better.

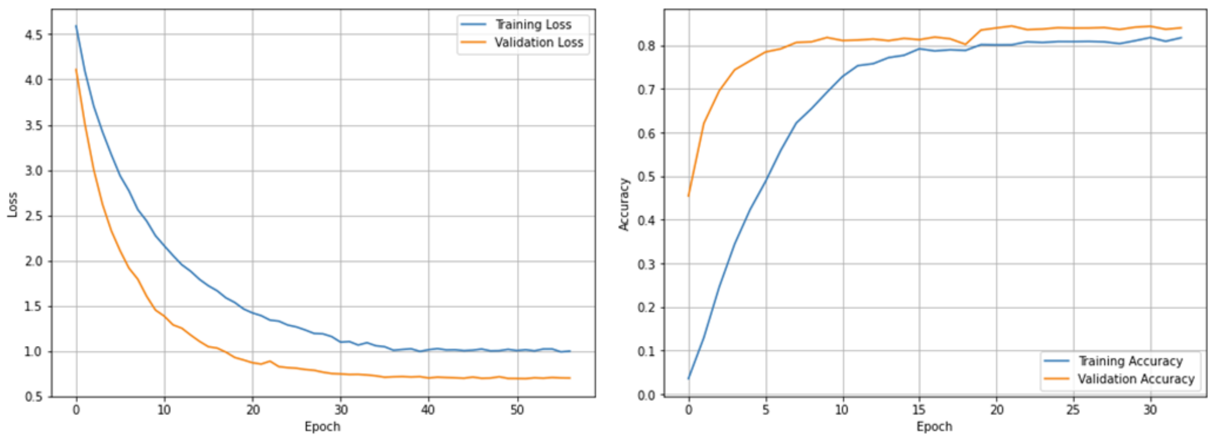

Figure 1. Loss and accuracy curves.

Regarding accuracy, the trend showed a steady increase during training, indicating that the model was improving in its ability to classify the data correctly. The accuracy trend was generally more stable than the loss trend, with no significant fluctuations after convergence. It is worth noting that the accuracy of the model did not reach an impressive rate (lower than 90%), which may suggest that the model could still benefit from further optimization or more complex architectures.

In terms of convergence time, the model seemed to reach convergence after approximately 20 epochs, which is a reasonable number for a deep learning model. The initial learning rate, adjustment factor, and patience parameters likely played an important role in achieving this convergence. However, it is important to note that the exact convergence time may vary from data and model architectures.

In summary, the loss and accuracy trends during training indicate that the model is learning to fit the data and improving in its ability to classify the data correctly. The model converged after approximately 20 epochs, with no significant fluctuations afterwards. The initial learning rate, adjustment factor, and patience parameters played an important role in achieving convergence, and further optimization or more complex architectures may be necessary to improve the model's accuracy.

3.2. Result comparison of different networks

In the results of the VGG16, the top ten data have been selected for analysis in Table 1. The overall accuracy of VGG16 is 0.78, but its performance is not satisfactory in recognizing certain breeds. For example, the accuracy of Shih-Tzu recognition is only 0.61, indicating the model could be further improved in recognizing Shih-Tzu. In addition, the accuracy of VGG16 in recognizing Rhodesian_ridgeback breeds are also low, at 0.59. VGG16's performance is relatively good in recognizing other breeds, with accuracy generally above 0.7.

Table 1. Detailed performances of top ten categories generated from VGG16.

	precision	recall	f1-score	support
Chihuahua	0.83	0.63	0.72	38
Japanese_spaniel	0.88	0.84	0.86	45
Maltese_dog	0.73	0.87	0.79	61
Pekinese	0.85	0.77	0.81	44
Shih-Tzu	0.61	0.69	0.65	55
Blenheim_spaniel	0.91	0.94	0.93	53
papillon	0.9	0.96	0.92	45
toy_terrier	0.73	0.65	0.69	49
Rhodesian_ridgeback	0.59	0.68	0.63	38
Afghan_hound	0.93	0.89	0.91	64

As demonstrated in Table 2, the accuracy of VGG19 is 0.79, slightly higher than VGG16. VGG19 performs better than VGG16 in recognizing Shih-Tzu and Rhodesian_ridgeback breeds, with accuracies of 0.69 and 0.68, respectively. However, VGG19's performance in recognizing other breeds is similar to that of VGG16.

The overall accuracy of ResNet with different fully connected layer settings are 0.8309 (fully connected layer is 256), 0.8301 (fully connected layer is 512), and 0.82 (more fully connected layers). It can be observed that the overall accuracy of ResNet is higher than that of VGG16 and VGG19.

ResNet's performance in recognizing Shih-Tzu and Rhodesian_ridgeback breeds are also better than that of the VGG series networks. For example, ResNet's accuracy in recognizing Shih-Tzu breeds reached 0.82, far higher than that of VGG16 and VGG19. Similarly, in recognizing Rhodesian_ridgeback breeds, ResNet's accuracy also reached 0.78, which is also superior to the VGG series networks.

Table 2. Result comparison of various models.

	Macro Avg			Weighted Avg			ACC
	precision	recall	f1	precision	recall	f1	
vgg 16 fully connected layer is 256	0.78	0.78	0.77	0.79	0.78	0.78	0.78
vgg 19 fully connected layer is 256	0.79	0.79	0.78	0.8	0.79	0.79	0.79

Table 2. (continued).

resnet 152 fully connected layer is 256	0.83	0.83	0.82	0.84	0.83	0.83	0.83
resnet 152 fully connected layer is 512	0.83	0.83	0.83	0.84	0.83	0.83	0.83
resnet 152 more fully connected layers	0.82	0.82	0.81	0.83	0.82	0.81	0.82

3.3. Result comparison of different optimizers and learning rate schedulers

In the study of dog breed classification based on the ResNet152 model, the author explored the impact of different optimizers and learning rate scheduling methods on the final evaluation metrics. Six control experiments were designed in this study, using two learning rate scheduling methods and three types of optimizers, namely ReduceLROnPlateau (lr_scheduler), CosineAnnealingLR (lr_scheduler), SGD (optimizer), Adam (optimizer), and Adagrad (optimizer),

Table 3. Results comparison of various learning rate schedulers and optimizers.

	Macro Avg			Weighted Avg			ACC
	precision	recall	f1	precision	recall	f1	
ReduceLROnPlateau -SGD	0.74	0.69	0.65	0.74	0.69	0.66	0.69
ReduceLROnPlateau -Adam	0.83	0.83	0.82	0.84	0.83	0.83	0.83
ReduceLROnPlateau -Adagrad	0.82	0.82	0.82	0.83	0.82	0.82	0.82
CosineAnnealingLR-Adagrad	0.82	0.82	082	0.83	0.82	0.82	0.82

Table 3 demonstrates the evaluation results of different learning rate scheduling methods on a classification task. In the comparison, ReduceLROnPlateau-SGD has the lowest accuracy (0.69) and f1-score (0.66), indicating that it has the worst overall performance. On the other hand, ReduceLROnPlateau-Adam has the highest accuracy (0.83) and f1-score (0.83), suggesting that it achieves the best performance among all methods. The results of the other three methods, including ReduceLROnPlateau-Adagrad, CosineAnnealingLR-Adagrad, and CosineAnnealingLR-Adam, are relatively similar, with 0.82 accuracy and around 0.82-0.83 f1-score.

In terms of precision and recall, most methods have similar values, with an average of around 0.82-0.83. However, ReduceLROnPlateau-SGD has relatively low values in both metrics (0.74 and 0.69, respectively), indicating that it may have difficulty correctly identifying some of the positive samples.

Overall, the results suggest that the choice of learning rate scheduling method can influence the performance. Among the tested methods, ReduceLROnPlateau-Adam achieves the best overall performance.

4. Conclusion

This work analyzes in detail the performance of several deep learning models, including ResNet152, VGG16 and VGG19 in the dog breed classification task. During training, our research observe a gradual decrease in loss and a steady increase in accuracy, indicating that the model is making progress in learning to fit the data. The model converges after about 20 epochs, and the initial learning rate, adjustment factor, and patience parameter play a key role in achieving convergence. However, the achieved accuracy is lower than 90%, indicating that there is room for improvement, such as further optimization or using a more complex network architecture. The results of this study demonstrate that the ResNet outperforms the VGG in most cases, possibly due to its reduced computational complexity while maintaining accuracy. At the same time, this study found that increasing the input size can improve the accuracy rate, but the accuracy may decrease slightly when increasing the number of network layers. This study also explores learning rate decay scheduling methods and optimizers, such as ReduceLROnPlateau, CosineAnnealingLR, SGD, Adam, and Adagrad, which may also have an impact on improving model performance. This study provides valuable insights into the dog breed classification task, while providing guidance for further improving model performance. Future research can focus on how to improve existing models, explore novel network architectures, and optimize for specific categories to achieve better performance in dog breed classification tasks.

References

[1] Ráduly, Z., Sulyok, C., Vadászi, Z., & Zölde, A. (2018). Dog breed identification using deep learning. In 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), 000271-000276.
[2] Borwarnginn, P., Kusakunniran, W., Karnjanapreechakorn, S., & Thongkanchorn, K. (2021). Knowing your dog breed: Identifying a dog breed with deep learning. International Journal of Automation and Computing, 18, 45-54.
[3] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., et, al. (2018). Recent advances in convolutional neural networks. Pattern recognition, 77, 354-377.
[4] Kim, P., & Kim, P. (2017). Convolutional neural network. MATLAB deep learning: with machine learning, neural networks and artificial intelligence, 121-147.
[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 770-778.
[6] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., et, al. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115, 211-252.
[7] Khosla, A., Jayadevaprakash, N., Yao, B., & Li, F. F. (2011). Novel dataset for fine-grained image categorization: Stanford dogs. In Proc. CVPR workshop on fine-grained visual categorization (FGVC), 2(1), 1-2.
[8] Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2019). Do imagenet classifiers generalize to imagenet?. In International conference on machine learning, 5389-5400.
[9] Woodworth, B., Patel, K. K., Stich, S., Dai, Z., Bullins, B., et, al. (2020). Is local SGD better than minibatch SGD?. In International Conference on Machine Learning, 10334-10343.
[10] Zhang, Z. (2018). Improved adam optimizer for deep neural networks. In 2018 IEEE/ACM 26th international symposium on quality of service, 1-2.