BMBAD BJESR Mühendislik Bilimleri ve Araştırmaları Dergisi
Journal of Engineering Sciences and Researches

Optimizing Hyperparameters for Enhanced Performance in Convolutional Neural Networks: A Study Using Transfer Learning Models

Evrişimli Sinir Ağlarında Gelişmiş Performans için Hiperparametrelerin Optimize Edilmesi: Transfer Öğrenme Modelleri Kullanılarak Yapılan Bir Çalışma

[1]İbrahim AKSOY, [2]Kemal ADEM

[1]Aksaray Üniversitesi, Sosyal Bilimler Enstitüsü, Yönetim Bilişim Sistemleri Anabilim Dalı, 68100, Aksaray, TÜRKİYE
[2]Sivas Bilim ve Teknoloji Üniversitesi, Bilgisayar Mühendisliği, 58100, Sivas, TÜRKİYE

[1]aksoy545@hotmail.com, [2]kemaladem@sivas.edu.tr

Araştırma Makalesi/Research Article

ARTICLE INFO

Article history

Received : 13 January 2024
Accepted : 16 February 2024

Keywords:

Image Classification, DenseNet, NASNetMobile, Optimization Algorithm, CNN

ABSTRACT

Convolutional neural networks, inspired by the workings of biological neural networks, have proven highly successful in tasks like image data recognition, classification, and feature extraction. Yet, designing and implementing these networks pose certain challenges. One such challenge involves optimizing hyperparameters tailored to the specific model, dataset, and hardware. This study delved into how various hyperparameters impact the classification performance of convolutional neural network models. The investigation focused on parameters like the number of epochs, neurons, batch size, activation functions, optimization algorithms, and learning rate. Using the Keras library, experiments were conducted using NASNetMobile and DenseNet201 models—highlighted for their superior performance on the dataset. As a result of the studies, the accuracy rate of the NASNetMobile model increased by 6.1% from 0.617 to 0.678, and the accuracy rate of the DenseNet201 model increased by 11.55% from 0.668 to 0.786.

MAKALE BİLGİSİ

Makale Tarihleri

Gönderim : 13 Ocak 2024
Kabul : 16 Şubat 2024

Anahtar Kelimeler:

Görüntü Sınıflandırma, DenseNet, NASNetMobile, Optimizasyon Algoritmaları, ESA

ÖZET

Biyolojik sinir ağlarının işleyişinden esinlenen evrişimli sinir ağlarının görüntü verisi tanıma, sınıflandırma ve özellik çıkarma gibi görevlerde oldukça başarılı olduğu kanıtlanmıştır. Yine de, bu ağların tasarlanması ve uygulanması bazı zorluklar ortaya çıkarmaktadır. Bu zorluklardan biri, belirli model, veri kümesi ve donanıma göre uyarlanmış hiperparametrelerin optimize edilmesidir. Bu çalışmada, çeşitli hiperparametrelerin evrişimli sinir ağı modellerinin sınıflandırma performansını nasıl etkilediği araştırılmıştır. Araştırma epok sayısı, nöronlar, yığın boyutu, aktivasyon fonksiyonları, optimizasyon algoritmaları ve öğrenme oranı gibi parametrelere odaklanmıştır. Keras kütüphanesi kullanılarak NASNetMobile ve DenseNet201 modelleri (veri kümesindeki üstün performansları nedeniyle vurgulanmıştır) kullanılarak deneyler yapılmıştır. Yapılan çalışmalar neticesinde NASNetMobile modelinde 0,617 olan doğruluk oranı 0,678'ye kadar yükselerek % 6,1 oranında, DenseNet201 modelinde ise 0,668 olan doğruluk oranı 0,786'ya yükselerek %11,55 oranında artış sağlanmıştır.

ORCID ID: [1]0000-0001-7708-8731
[2]0000-0002-3752-7354

1. INTRODUCTION

In what can be described as the information age, information is used not only by humans but also by computers through machine learning and artificial intelligence methods. Artificial neural networks are generally used in the learning process, which is one of the basic elements of artificial intelligence. Artificial neural networks (ANN), inspired by the working principles of the human brain, enable the development of systems that can solve complex problems, recognize patterns and make predictions [1].

Deep learning is one of the most widely used methods in pattern recognition, object detection, object tracking and classification. The basis of deep learning is multilayer artificial neural networks. In classical machine learning methods, attribute extraction and selection are performed by researchers, whereas in deep learning methods, features and their weights are determined by the model throughout the layers. Deep learning methods have gained popularity by relieving researchers from the burden of feature extraction and obtaining more useful features thanks to interconnected layers [2]. In deep learning models, it is usual to obtain different success performances depending on variables such as the problem itself, hardware, dataset, variety and size of the dataset. For this reason, there are specialized methods in different areas according to the problems in deep learning. Convolutional neural networks (CNN) are the most successful and most preferred deep method in image processing. CNN has a multilayer structure inspired by the functioning of the biological visual system. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was won by the CNN model Alexnet algorithm in 2012, and its success has gradually increased in the following years. CNN has become the most preferred deep learning model in many fields thanks to its high performance in image recognition and classification problems. When the literature is examined, it is observed that CNN is used not only in image processing but also in other data types such as natural language processing and time series [3].

In addition to this rise and success of CNN, the creation of a neural network model suitable for the problem and the selection of the best hyperparameters for this model is a very complex issue. In this study, it is aimed to observe the effect of different values of hyperparameters in CNN models on the classification performance of the model.

1.1. Related Works

Thanks to the successful developments in object detection, studies are carried out in many fields such as health, production and defense industry, security systems, electronics, design and architecture. Studies on CNNs and their parameters, which are the main elements of these studies, also make important contributions to the field.

In a study by Smith [4] on learning rates in deep neural networks, a cyclical approach was proposed to determine learning rates. In the study, this approach, called iterative learning rates, was tested with CIFAR-10 and CIFAR-100 datasets in ResNet and DenseNet models, and with ImageNet dataset in Alexnet and GoogLeNet models. Smith argued that this approach would be a practical method for people who train neural networks by reducing the assumptions in determining ideal learning rates. Bircanoğlu and Arıca [5] compared the effects of commonly used Linear, Sigmoid, Tanh, Hard Sigmoid, Softsign, ReLU, Softplus, ELU, SeLU, Swish activation functions and Square activation function on classification performance. As a result of experimental studies with different datasets, it was stated that the best performing activation function varied according to the dataset, but generally good results were obtained with ReLU activation function in all datasets. In the study titled "Investigation of Hyper-Parameter Optimization Methods in Convolutional Neural Networks" published by Gülcü and Kuş [6], they examined genetic algorithm, particle swarm optimization, differential evolution and Bayesian optimization methods on CNN. In their study, as a result of their tests with different datasets, they stated that it is not possible to make a generalization to choose the best hyperparameter set and that hyperparameter optimization may vary according to the problem and dataset. In the study published by Seyyarer et al [7], the performance of optimization algorithms in classification success was compared using Caltech 101 and Caltech 256 datasets. In the study, the success rates of the optimization algorithms in classification with 64x64 image size, ReLU activation function and cross entropy error function were found as adadelta 86.88%, adagrad 71.25%, adam 92.31%, momentum 85.56%, rmsprop 40.26% and sgd 64.5%, It was also suggested to use gardient descent-based optimization algorithms (SGD, Momentum, Adam) to minimize the error in large datasets. Adem [8] investigated the effects of P+ FELU activation function which is a combination of FELU, ELU and ReLU activation functions. He made comparisons by working with MNIST, CIFAR-10 and CIFAR-100 datasets. It is stated that the P+ FELU activation function with flexible properties can effectively prevent the vanishing gradient. It is concluded that the proposed activation function outperforms ELU, SELU, MPELU, TReLU, ReLU and FELU activation functions. Liashchynskyi and Liashchynskyi [9] compared grid search, random search and genetic algorithm methods for hyperparameter optimization. As a result of the study, they stated that the grid search method takes too long and is costly, the random search method is faster, but this method cannot guarantee the best results, and the genetic algorithm method takes a long time to run, but this situation can be controlled with some variables. They concluded that the genetic algorithm method is preferable when there are more parameter options.

In this study, CNN models from the Keras deep learning library are used. In the selected model, 53 different trainings were performed to observe the effects of batch size, number of training epochs, optimization algorithms, activation functions, learning rate and number of neurons hyperparameters on model performance. The study is

expected to contribute to researchers in the selection and optimization of hyperparameters to be used in CNN models.

2. MATERIAL VE METHODS

2.1. Datasets

Two different datasets were used in this study. The datasets were obtained from the Kaggle platform, which has a lot of content in various fields. No pre-processing or editing was done on the datasets used. The first dataset is the dataset published on Kaggle under the name "Dogs & Cats Images" and contains images of dogs and cats taken from different angles [10]. The dataset consists of two classes, "cats" and "dogs", and a total of 10000 images. 8000 of the images are allocated to the training dataset and 2000 of the images are allocated to the test dataset with equal distribution to the two classes. The second dataset is a dataset published on kaggle.com under the name "pizza classification data" [11]. It consists of images of pizza and non-pizza dishes taken from different angles. The images are of different sizes with a maximum edge length of 512 pixels. The dataset consists of two classes, "pizza" and "not_pizza". A total of 1966 images, 983 each, are equally distributed in the two classes. Of the images in the dataset, 1600 images are allocated to the training set and 366 images are allocated to the test dataset.

2.2. Deep Learning and Libraries

Deep learning is a more advanced subtype of multilayer artificial neural networks. Deep learning models are composed of many layers, and each layer aims to extract features by taking the output of its predecessors. Since the first feed-forward multilayer deep learning model developed by Ivakhnenko and Lapa [12], there has been significant progress in deep learning models [13]. Since designing and training deep learning models is a difficult and costly task, open source deep learning libraries have been developed to help users in this sense. These libraries provide users with significant facilities to create learning algorithms [14]. In this study, TensorFlow is used as a sub-library and Keras library is used as a high-level library due to its ease of use and up-to-date nature.

2.3. Convolutional Neural Networks and Models Used

Inspired by the functioning of biological neural networks, CNN is one of the most successful and most preferred deep learning methods in image processing. Alexnet, an CNN model, won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 and its success gradually increased in the following years. CNN has become the most preferred deep learning model in many fields thanks to its high performance in image recognition and classification problems. Designers continue to improve these models and introduce new models every day. However, there are also libraries where popular models are brought together and made available for use. These libraries, which are easier to use than designing a new model, are widely used in CNN studies, especially in areas such as health, agriculture, production and industry. In this sense, the Keras library is a library that offers very up-to-date and popular CNN models. In this study, experimental studies were conducted on 27 CNN models in the Keras library. In these models, trainings were performed with two datasets and the models with the highest accuracy rates, DenseNet201 and NASNetMobile, are given in the subsections.

2.3.1. DenseNet

The basis of the DenseNet architecture is the dense blocks in the convolutional layers, which give the model its name. Unlike traditional CNN models that have a hierarchical structure, DenseNet is that each layer is fed with the outputs of all previous layers instead of only the outputs of the previous layer [15]. The layer structure of the DenseNet architecture is shown in Figure 1.

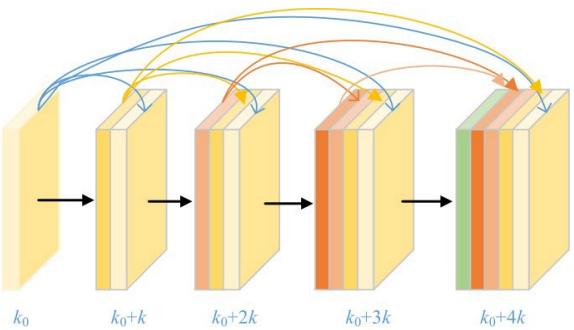

Figure 1. DenseNet layer structure [16].

2.3.2. NASNetMobile

NASNetMobile is an artificial neural network model designed by Google in 2017 through deep learning based on the Neural Architecture Search (NAS) method [17]. NASNet is composed of cells that can be improved by

reinforcement learning. In these cells, several convolution and pooling operations are performed and repeated many times according to the capacity of the network. The architecture of the NASNetMobile model is given in Figure 2.

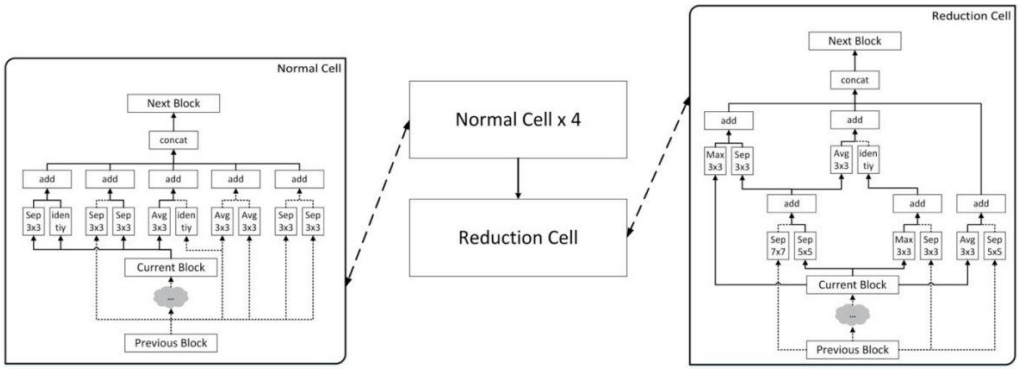

Figure 2. NASNetMobile layer structure [18].

2.4. Hyperparameters in Convolutional Neural Networks

Hyperparameters are predetermined control parameters of the model used in training [19]. Some of the parameters used in CNN models are layer-specific and some are considered as common parameters of the model. These parameters and their grouping according to layers are shown in Table 1.

Table 1. Hyperparameters that can be optimized [6]

Convolution	Pooling	Fully-connected	General
• *Filter size*	• *Filter size*	• *Number of neurons*	• *Optimization method*
• *Number of filters*	• *Stride*	• *Number of layers*	• *Batch size*
• *Stride*	• *Centering detection.*	• *Initial number of layers*	• *Learning rate*
• *Padding*	• *Number of layers*		• *Initial learning rate*
• *Activation function*			• *Dropout rate*
• *Number of layers*			• *Dropout active*
• *Initial number of layers*			• *Regularization method*
			• *Regularization rate*
			• *Weight initializer*
			• *Weight reducer*
			• *Weight multiplier*
			• *Weight normalization*
			• *Weight penalty value*
			• *Number of iterations*
			• *Momentum*
			• *Bias active (bias*
			• *Bias inception*
			• *Bias onset rate*
			• *Gauss ratio*

2.4.1. Number of Epoch

When training a neural network model, the training dataset is usually passed through the network multiple times. The parameter that determines how many times the entire training dataset is run in the network's learning algorithm is called the number of training rounds (Epoch) [20]. The number of epochs can take a value between one and infinity. In the literature, epoch numbers are usually chosen from high values such as 100, 200, 500 [21]. However, a higher number of epochs does not necessarily mean that the network will be trained better or achieve better success values. However, the higher the number of epochs, the more time it will take to train the model [22].

2.4.2. Number of Neurons

In neural networks, the number of neurons is one of the parameters that have a significant impact on the complexity and learning ability of the network. Neurons are connected to the layers with the weights of the data determined in the network and the output value of each neuron is used as the input data of the next neuron [23]. As the number of neurons increases, the learning capacity of the network is expected to increase, but this is not always the case. Using too many neurons in the architecture of the network may lead to overfitting and weakening of the generalization ability [24]. In addition, a large number of neurons increases the size and computational load of the network. If the hardware capacity is low, this has a negative impactin terms of time. If the number of neurons is less than the required number, the model may not be able to represent the dataset [25].

2.4.3. Batch Size

When training a neural network model, it is not a preferred method to pass all the training data through the network at the same time. Instead, the dataset is passed through the network step by step in forward and backward propagation. The amount of data in these chunks or the number of samples taken from the dataset in each training round is called the batch size [26]. In the literature, it is seen that the batch size is usually composed of 32, 64 and 128 samples [21].

2.4.4. Activation Functions

In CNN models, activation functions are used to transmit the output values produced by neurons in one layer to the next layer. Activation functions are used to determine the threshold values of the output values in the layers and to decide whether they will be transmitted to the next layer, and therefore whether the artificial neural cell will be active [27]. Since CNN is generally used for nonlinear classifications, activation functions are also chosen from nonlinear functions. These functions usually produce output values in the range [-1,1] or [0,1]. The most commonly used activation functions and their mathematical formulas in the Keras library are given in Table-2.

Table 2. Activation Functions [28]

Activation Function	Mathematical Formula
ReLU	$\max(0, x)$
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
Tanh	$\tanh(x)$
ELU	$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$
Softplus	$\ln(1 + e^x)$

Although researchers generally use the popular activation functions available in the Keras library, there are also studies using activation functions that are newly introduced or created by combining different functions. Kılıçarslan et al. used new activation functions such as RSigELU and Superior Exponential (SupEx) and obtained better results compared to popular functions [29-30].

2.4.5. Optimization Methods

In artificial neural networks, optimization methods are used to find the best difference between the output value given by the network and the actual value. In the literature, these optimization methods are also called gradient descent [7]. In CNN applications, the goal is often not to minimize the error rate but to make the best generalization. It is known that the choice of optimization algorithm used in the CNN model will be effective in making this generalization. However, no algorithm guarantees the best solution, but some methods are used to help choose the best optimization algorithm for the model [31]. The most commonly used algorithms as optimization methods in deep learning are SGD, Adagrad, RMSProp, Adadelta, Adam and Adamax [32].

2.4.6. Learning Rate

Learning rate is a coefficient used in convergence calculations in optimization algorithms. Learning rate is one of the most important parameters in the training of neural networks [4]. It is known that setting this ratio too large will fail to achieve the convergence goal and even cause divergence, while setting it too small will provide a better convergence with small steps [33]. However, as the learning rate becomes smaller, the training time will increase as the optimization algorithm will proceed in smaller steps. It is very difficult to precisely adjust the learning rate in optimization algorithms [26].

3. RESULTS

The Google Colab platform was used to design the CNN models used in the study and to implement the training tasks. The testing of the transfer learning models and the applications with the second dataset were carried out with 12.7 GB RAM, 15 GB GPU and 72.8 GB disk space hardware provided with the standard Colab membership.The applications with the first dataset were run with 85 GB RAM, 40 GB Tesla A100 GPU and 170 GB disk space provided with Colab Pro membership.

In order to determine the model to be used in the study, performance tests were performed on 27 models in the Keras library with our datasets. In order to examine the effects of the parameters, the models were used without weights pre-trained with the ImageNet dataset. The parameter values used in these trainings are as follows;

Number of training rounds (Epoch) = 10

Package size = 32

Activation function = ReLU,

Optimization algorithm = Adam
Learning rate = 0.001

Table 3. Performance results of CNN models in Keras

First Dataset Results					*Second Dataset Results*				
No	**Model**	**Accuracy**	**Epochs**	**Time (sec.)**	**No**	**Model**	**Accuracy**	**Epochs**	**Time (sec.)**
1	NASNetMobile	0.6175	10	273.46	**1**	DenseNet201	0.6687	10	66.10
2	ResNet101V2	0.6167	10	267.62	**2**	InceptionResNetV2	0.6675	10	85.16
3	DenseNet169	0.6162	10	278.22	**3**	NASNetMobile	0.6667	10	70.95
4	ResNet50	0.6150	10	257.73	**4**	DenseNet121	0.6562	10	70.56
5	DenseNet201	0.6113	10	280.25	**5**	ResNet152V2	0.6438	10	95.95
6	ResNet152V2	0.6100	10	279.85	**6**	ResNet101V2	0.6375	10	68.71
7	DenseNet121	0.6087	10	282.77	**7**	DenseNet169	0.6313	10	68.42
8	ResNet50V2	0.6012	10	258.21	**8**	ResNet50	0.6187	10	63.61
9	InceptionResNetV2	0.5900	10	286.43	**9**	ResNet50V2	0.6125	10	62.77
10	VGG16	0.5838	10	256.08	**10**	InceptionV3	0.5250	10	61.22
11	VGG19	0.5038	10	264.16	**11**	EfficientNetB6	0.4750	10	72.89
12	ResNet152	0.5038	10	285.43	**12**	EfficientNetB3	0.4750	10	70.71
13	ResNet101	0.5038	10	274.85	**13**	VGG19	0.4750	10	54.21
14	Xception	0.5038	10	258.15	**14**	VGG16	0.4750	10	63.87
15	EfficientNetB7	0.5038	10	306.26	**15**	EfficientNetB0	0.4750	10	68.57
16	EfficientNetB2	0.5038	10	268.53	**16**	EfficientNetB1	0.4750	10	67.11
17	EfficientNetB1	0.5038	10	283.10	**17**	EfficientNetB2	0.4750	10	59.88
18	EfficientNetB0	0.5038	10	261.37	**18**	ResNet152	0.4750	10	92.00
19	MobileNet	0.4963	10	251.58	**19**	ResNet101	0.4750	10	79.33
20	MobileNetV3Large	0.4963	10	270.40	**20**	EfficientNetB7	0.4750	10	75.78
21	MobileNetV3Small	0.4963	10	251.78	**21**	EfficientNetB4	0.4750	10	77.36
22	MobileNetV2	0.4963	10	261.31	**22**	MobileNetV3Small	0.4750	10	58.07
23	EfficientNetB6	0.4963	10	278.30	**23**	MobileNetV3Large	0.4750	10	63.32
24	EfficientNetB5	0.4963	10	292.48	**24**	MobileNet	0.4750	10	51.30
25	EfficientNetB4	0.4963	10	293.12	**25**	MobileNetV2	0.4750	10	47.62
26	EfficientNetB3	0.4963	10	271.77	**26**	EfficientNetB5	0.4750	10	74.51
27	InceptionV3	0.4963	10	261.88	**27**	Xception	0.4750	10	64.10

In the applications with the first dataset, the best classification performance was obtained with the "NASNetMobile" model. In the applications with the second dataset, the best classification performance was obtained with the "DenseNet201" model. We continued our study with the models with the best performance in the two datasets.
In this study, experimental studies were conducted on the number of epochs, batch size, activation function, optimization methods, learning rate and number of neurons among the parameters given in Table 1. These studies were carried out separately for the models where the best classification performance was obtained for both datasets. Firstly, the number of epoch's parameter was varied with 5, 10, 25 and 50 values and the results are given in Table 3.

Table 4. Training results by number of epochs

First Dataset						**Second Dataset**					
No	**Model**	**Epochs**	**Loss**	**Accuracy**	**Time (sec.)**	**No**	**Model**	**Epochs**	**Loss**	**Accuracy**	**Time (sec.)**
1	NASNetMobile	5	0.6936	0.5615	359	5	DenseNet201	5	0.5441	0.6639	73
2	NASNetMobile	10	0.6407	0.62	528	6	DenseNet201	10	0.5221	0.7459	174
3	NASNetMobile	25	0.6337	0.627	1296	7	DenseNet201	25	0.5268	0.724	350
4	NASNetMobile	50	0.6534	0.6005	2536	8	DenseNet201	50	0.5673	0.7022	776

When Table is examined, it is seen that the highest accuracy rate was obtained at 25 epochs in the training of NASNetMobile model with the first dataset, and the highest accuracy rate was obtained with 10 epochs in the training of DenseNet201 model with the second dataset. In the number of neurons parameter, experiments were performed on 32, 64, 128, 256 and 512 values.
As can be seen in Table 5, the number of neurons providing the highest accuracy rate for both models was 256. As the number of neurons increases, the learning capacity of the network is expected to increase, but using more neurons than necessary may lead to overfitting and weakening of the generalization ability.
Batch size is one of the important parameters that should be adjusted according to the dataset. In this study, the batch size was changed to 8, 16, 32 and 64 and both datasets were trained with the specified models and the results are given in Tables 6 and 7.

Table 5. Training results by neuron number

First Dataset							Second Dataset						
No	Model	Epoch	Neuron	Loss	Accuracy	Time	No	Model	Epoch	Neuron	Loss	Accuracy	Time
9	NASNetMobile	25	32	0.5255	0.6315	808	14	DenseNet201	10	32	0.5255	0.6967	112
10	NASNetMobile	25	64	0.5047	0.639	806	15	DenseNet201	10	64	0.5047	0.6749	104
11	NASNetMobile	25	128	0.4897	0.6355	805	16	DenseNet201	10	128	0.4897	0.7486	104
12	NASNetMobile	25	256	0.5244	0.6535	804	17	DenseNet201	10	256	0.5244	0.776	103
13	NASNetMobile	25	512	0.5126	0.6525	806	18	DenseNet201	10	512	0.5124	0.7541	104

Table 6. Batch size training results with the first dataset

No	Model	Epochs	Neuron	Batch Size	Loss	Accuracy	Time (sec)
19	NASNetMobile	25	256	8	0.5986	0.6825	1287
20	NASNetMobile	25	256	16	0.6342	0.625	1206
21	NASNetMobile	25	256	32	0.6532	0.6015	1083
22	NASNetMobile	25	256	64	0.6369	0.6355	1181

Table 7. Batch size training results with the second dataset

No	Model	Epochs	Neuron	Batch Size	Loss	Accuracy	Time (sec)
23	DenseNet201	10	256	8	0.5017	0.7787	194
24	DenseNet201	10	256	16	0.5165	0.7568	178
25	DenseNet201	10	256	32	0.5205	0.7541	180
26	DenseNet201	10	256	64	0.5603	0.735	196

The smaller the batch size, the more number of times the network is trained. It is observed that this leads to a certain increase in the performance of the model. However, Table 6 shows that as the batch size decreases, the training time increases due to the increase in the number of iterations. It should be considered that this may be a disadvantage in studies with larger datasets. In this study, the highest performance among the two models was achieved with a batch size of 8.

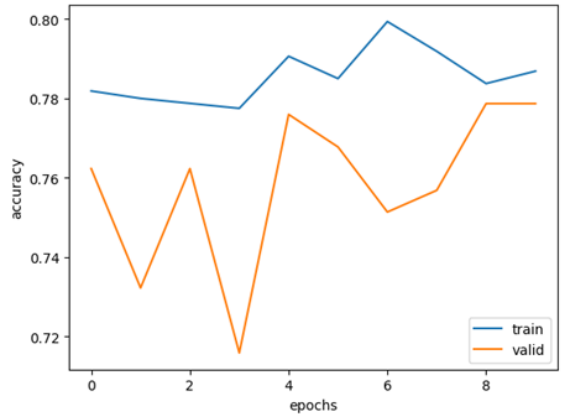

Figure 3. Accuracy graph of 8 batch size training.

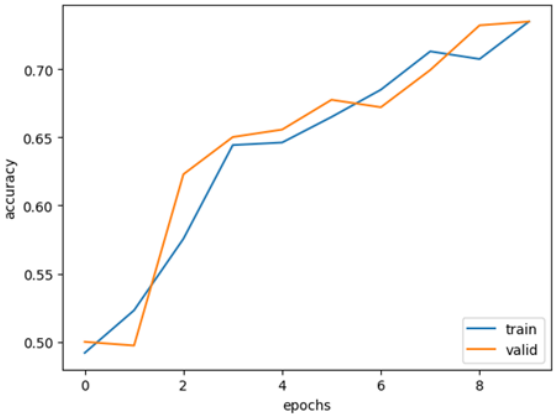

Figure 4. Accuracy graph of 64 batch size training.

As can be seen in Figure 3, the fact that the training accuracy rate is more consistent while the test accuracy rate shows a fluctuating graph can be attributed to the small batch size. This can be explained by the fact that when the batch size given for testing is small, the number of samples with high or low discriminability is more likely to be unevenly distributed.

As for the activation functions, six different activation functions were used in the intermediate layers: ELU, ReLU, Sigmoid, Softmax, Softplus, Tanh. The results obtained according to the models in these studies are shown in Table 8.

As seen in Table 8, the best classification performances were obtained with the ELU activation function in the DenseNet201 model, and with the Softplus and ReLU activation functions in the NASNetMobile model.

According to the results, although good results are obtained with the ReLU activation function for both models, it should be tested with more models and datasets in order to make generalizations. Considering the properties of activation functions, different functions should be tested and the most appropriate one for the dataset and model should be decided.

One of the parameters examined in the study is optimization algorithms. Seven different optimization algorithms,

Table 8. Results according to activation functions

No	Model	Epochs	Neuron	Batch Size	Activation Functions	Accuracy	Time (sec)
27	DenseNet201	10	256	8	ELU	0.7842	139.74
28	DenseNet201	10	256	8	ReLU	0.7705	142.23
29	DenseNet201	10	256	8	Tanh	0.7623	129.66
30	DenseNet201	10	256	8	Softplus	0.7322	131.92
31	DenseNet201	10	256	8	Sigmoid	0.6940	137.09
32	DenseNet201	10	256	8	Softmax	0.5000	140.85
33	NASNetMobile	25	256	8	Softplus	0.6480	1018
34	NASNetMobile	25	256	8	ReLU	0.6480	1127
35	NASNetMobile	25	256	8	Sigmoid	0.6365	1024
36	NASNetMobile	25	256	8	ELU	0.6070	1129
37	NASNetMobile	25	256	8	Tanh	0.5000	1078
38	NASNetMobile	25	256	8	Softmax	0.5000	1013

namely Adadelta, Adagrad, Adam, Adamax, Nadam, SGD and RMSprop, were used for model training and their classification performances are presented in Table 9.

Table 9. Result performance of optimization functions

No	Model	Epochs	Neuron	Batch Size	Activation Functions	Optimization Algorithms	Accuracy	Time (sec)
34	DenseNet201	10	256	8	ELU	RMSprop	0.7732	127.90
35	DenseNet201	10	256	8	ELU	Adamax	0.7377	116.71
36	DenseNet201	10	256	8	ELU	Nadam	0.7322	128.04
37	DenseNet201	10	256	8	ELU	Adam	0.7319	116.34
38	DenseNet201	10	256	8	ELU	Adagrad	0.5956	115.91
39	DenseNet201	10	256	8	ELU	Adadelta	0.5000	115.12
40	DenseNet201	10	256	8	ELU	SGD	0.5000	116.27
41	NASNetMobile	25	256	8	Softplus	Adam	0.6525	1029.47
42	NASNetMobile	25	256	16	Softplus	Adagrad	0.6295	1026.14
43	NASNetMobile	25	256	32	Softplus	SGD	0.6280	1034.05
44	NASNetMobile	25	256	48	Softplus	Nadam	0.6235	1041.37
45	NASNetMobile	25	256	64	Softplus	Adamax	0.6210	1025.61
46	NASNetMobile	25	256	80	Softplus	RMSprop	0.6170	1034.03
47	NASNetMobile	25	256	96	Softplus	Adadelta	0.6025	1025.97

As seen in Table 9, the highest accuracy values were obtained with the RMSprop algorithm in training with the DenseNet201 model and with the Adam algorithm in training with the NASNetMobile model. The accuracy graphs of the optimization algorithms according to the models are given in Figures 5 and 6.

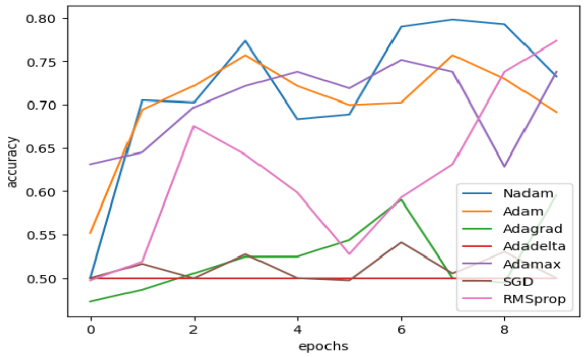

Figure 5. Accuracy graph of optimization optimizationalgorithms with DenseNet201.

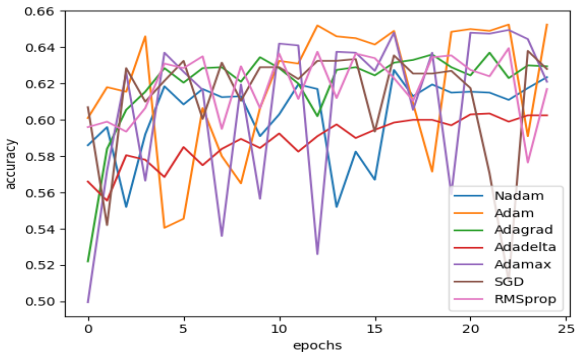

Figure 6. Accuracy graph of algorithms with NASNetMobile.

In the graph shown in Figure 5, the optimization algorithms have more distant performances from each other. This difference is thought to be due to the dataset rather than the model. Since the dataset used in the DenseNet201 model is smaller, a consistent result could not be obtained with some optimization algorithms in training. Figure 6 shows that in the NASNetMobile model, RMSprop, Adadelta and Adagrad algorithms draw a more consistent graph while Adam, Adamax, Nadam and SGD algorithms have a more fluctuating training process. On the other hand, the highest accuracy rate in the NASNetMobile model was obtained with the Adam optimization algorithm. As can be seen in the results of the training tasks performed in our study, the optimization algorithms that achieve the best performance on different models and datasets vary. Therefore, no generalization can be made about any

optimization algorithm. It is recommended to try different algorithms to find the most suitable optimization method for the datasets and models.

In CNN, the learning rate is the parameter that determines how large steps the network's weight updates will be made. In this study, training was performed with values between 0.01 and 0.0001 for the learning rate parameter. In order to better observe the effects of this parameter, the models were trained for 100 epochs unlike the previous trainings. The results obtained after these trainings are given in Table 10.

Table 10. Results of RMSprop and Adam algorithms according to learning rates

No	Model	Dataset	Epochs	Optimization Algorithms	Learning Rate	Accuracy	Time (sec)
48	DenseNet201	pizza classification data	100	RMSprop	0.01	0.6585	1251
49	DenseNet201	pizza classification data	100	RMSprop	0.001	0.7860	1269
50	DenseNet201	pizza classification data	100	RMSprop	0.0001	0.7486	1249
51	NASNetMobile	Dogs & Cats Images	100	Adam	0.01	0.6	1028
52	NASNetMobile	Dogs & Cats Images	100	Adam	0.001	0.6286	1033
53	NASNetMobile	Dogs & Cats Images	100	Adam	0.0001	0.6786	1011

As seen in Table 10, the RMSprop algorithm achieved the highest accuracy performance with a learning rate of 0.001 and the Adam algorithm achieved the highest accuracy performance with a learning rate of 0.0001. It is observed that the Adam algorithm gives better results at a lower learning rate than the RMSprop algorithm. This shows that the ideal learning rates may be different according to the optimization algorithm.

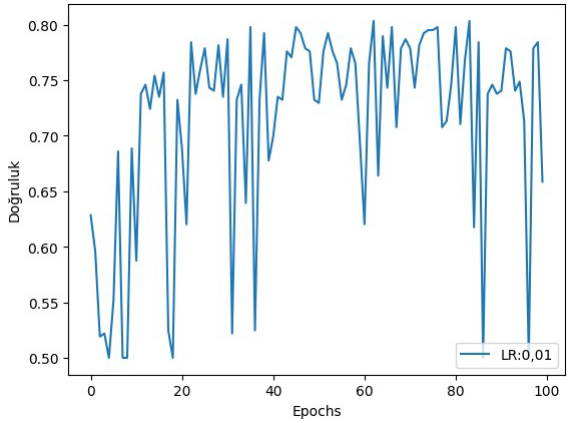

Figure 7. Accuracy of 0.01 learning rate in the RMSprop algorithm.

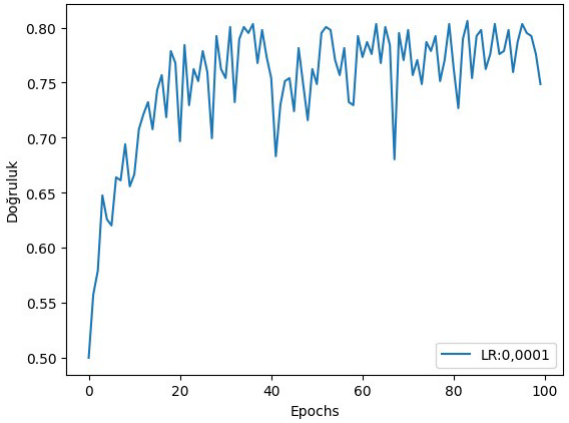

Figure 8. Accuracy of 0.0001 learning rate in the RMSprop algorithm.

Looking at the accuracy/epoch graphs given in Figures 7 and 8, it can be seen that when the learning rate is high, the training progresses in a more fluctuating structure, while as the learning rate decreases, the fluctuation decreases and it becomes more consistent. In summary, in this study, the NASNetMobile model achieved the highest performance with 10 epochs, 256 number of neurons, 8 batch sizes, Softplus activation function, Adam optimization algorithm and 0.0001 learning rate, while the DenseNet201 model achieved the highest performance with 25 epochs, 256 number of neurons, 8 batch sizes, ELU activation function, RMSprop optimization algorithm and 0.001 learning rate.

4. CONCLUSION

In order to examine the effect of the hyperparameters used in CNN on the classification performance of the model, 58 trainings were performed on NASNetMobile and DenseNet201 models with two different datasets. As a result of the trainings for the number of training rounds, it was seen that the ideal values of this parameter can be reached with different numbers according to the model and dataset. Considering the datasets used in the study, it can be said that larger datasets should be trained with higher epoch values. In the experiments on the number of neurons, it was observed that increasing the number of neurons in both models increased the learning capacity of the network and thus the classification performance. In the training studies on batch size, it was observed that the accuracy rate increased as the batch size decreased for both models. However, it should be noted that when the batch size is small, the number of iterations will be larger and the training time will be longer. As a result of the experiments with activation functions and optimization algorithms, no characterization could be made according to the models or dataset. Considering the properties of these parameters, different values should be tried and the most suitable one for the dataset and the model should be decided. Finally, it was observed that training was more unstable when the learning rates took large values, while training was more consistent with small learning rates. However, it should be kept in mind that small learning rates may cause the training to slow down, especially for large datasets. As a result of the parameter changes made on the CNN models in the study; in the NASNetMobilemodel, the

accuracy rate increased from 61.75% to 68.25%, an increase of 6.5%. In the DenseNet201 model, the accuracy increased from 66.87% to 78.42%, an increase of 11.55%. Although these findings show that hyperparameters have an impact on the classification performance of CNN models, our study also has some limitations. The values of only some of the hyperparameters used in the CNN models were selected and analyzedin certain ranges. Experiments were conducted for only a part of the probability space consisting of all values that all hyperparameters can take. In addition to the hyperparameters, the dataset used in the model also has an impacton performance. Training the models with two datasets is another limitation of the study. In the continuation of the study, experiments can be conducted on CNN models with different value ranges of different hyperparameters and more datasets. In addition, the examination of the parameters in Vision Transformers architectures, which is a new and impressive approach in the field of image processing, can be added to the continuation of the study.

Author Contributions

All stages of the study were done by the authors.

Statement of Conflict of Interest

Authors have declared no conflict of interest.

REFERENCES

[1] E. Öztemel "Yapay sinir ağları", Papatya Yayıncılık, İstanbul, 2003.

[2] S. Aktürk and K. Serbest, "Nesne Tespiti İçin Derin Öğrenme Kütüphanelerinin İncelenmesi", Journal of Smart Systems Research, vol. 3, no. 2, pp. 97-119, 2022.

[3] A. Onan, "Evrişimli sinir ağı mimarilerine dayalı Türkçe duygu analizi", Avrupa Bilim ve Teknoloji Dergisi, pp. 374-380, 2020.

[4] L.N. Smith, "Cyclical learning rates for training neural networks", IEEE winter conference on applications of computer vision (WACV), pp. 464-472, 2017.

[5] C. Bircanoğlu and N. Arıca, "Yapay Sinir Ağlarında Aktivasyon Fonksiyonlarının Karşılaştırılması", in 2018 26th signal processing and communications applications conference (SIU). IEEE, pp. 1-4, İzmir, 2018.

[6] A. Gülcü and Z. Kuş, "Konvolüsyonel sinir ağlarında hiper-parametre optimizasyonu yöntemlerinin incelenmesi", Gazi University Journal of Science Part C: Design and Technology, pp. 503-522, 2019.

[7] E. Seyyarer, F. Ayata, T. Uçkan and A. Karci, "Derin öğrenmede kullanılan optimizasyon algoritmalarının uygulanması ve kıyaslanması", Computer Science, vol. 5, no. 2, pp. 90-98, 2020.

[8] K. Adem, "P+ FELU: Flexible and trainable fast exponential linear unit for deep learning architectures", Neural Computing and Applications, vol. 34, no. 24, pp. 21729-21740, 2022.

[9] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for NAS", arXiv preprint arXiv:1912.06059, 2019.

[10] Kaggle, "Dogs & Cats Images", url: https://www.kaggle.com/datasets/chetankv/dogs-cats-images, (Access Date: 01/01/2024).

[11] Kaggle, "pizza classification data", url: https://www.kaggle.com/datasets/projectshs/pizza-classification-data, (Access Date: 01/01/2024).

[12] A.G. Ivakhnenko and V.G. Lapa, "Cybernetic predicting devices", 1966.

[13] A. Şeker, B. Diri and H. Balık, "Derin öğrenme yöntemleri ve uygulamaları hakkında bir inceleme", Gazi Mühendislik Bilimleri Dergisi, vol. 3, no. 3, pp. 47-64, 2017.

[14] Ö. İnik and E. Ülker, "Derin Öğrenmede Kullanılan Veri Setleri ve Yazılım Kütüphaneleri", International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) 2017, Tokat, 2017.

[15] F.D. Adhinata, D.P. Rakhmadani, M. Wibowo and A. Jayadi, "A deep learning using DenseNet201 to detect masked or non-masked face", JUITA: Jurnal Informatika, vol. 9, no. 1, pp. 115-121, 2021.

[16] S.H. Wang and Y.D. Zhang, "DenseNet-201-based deep neural network with composite learning factor and precomputation for multiple sclerosis classification", ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), vol. 16, no. 2, pp. 1-19, 2020.

[17] A.O. Adedoja, P.A. Owolawi, T. Mapayi and C. Tu, "Intelligent Mobile Plant Disease Diagnostic System Using NASNet-Mobile Deep Learning", IAENG International Journal of Computer Science, vol. 49, no. 1, pp. 216-231, 2022.

[18] A. Daşgın, "Covid19 Yayılımını Azaltmak İçin Yüz Maskesinin Evrişimsel Sinir Ağı Modelleri İle Tespiti", Master's thesis, Sosyal Bilimler Enstitüsü, Aksaray Üniversitesi, Aksaray, 2023.

[19] C. Doğan, "İstatistiksel ve Makine Öğrenme ile Derin Sinir Ağlarında Hiper-Parametre Seçimi İçin Melez Yaklaşım", Master's thesis, Fen Bilimleri Enstitüsü, Hacettepe Üniversitesi, Ankara, 2021.

[20] F. Kurt, "Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi", Master's thesis, Fen Bilimleri Enstitüsü, Hacettepe Üniversitesi, Ankara, 2018.

[21] J. Brownlee, "What is the Difference Between a Batch and an Epoch in a Neural Network", Machine Learning Mastery, vol. 20, 2018.

[22] W. Hastomo, A.S.B. Karno, N. Kalbuana, and A. Meiriki, "Characteristic parameters of epoch deep learning to predict Covid-19 data in Indonesia", Journal of Physics: Conference Series, vol. 1933, no. 1, pp. 1-4, 2021.

[23] M.A. Bülbül and C. Öztürk, "Optimization, modeling and implementation of plant water consumption control using genetic algorithm and artificial neural network in a hybrid structure", Arabian Journal for Science and Engineering, vol. 47, no. 2, pp. 2329-2343, 2022.

[24] A. Zhang, Z.C. Lipton, M. Li and A.J. Smola, "Dive into deep learning", arXiv preprint arXiv:2106.11342, 2021.

[25] T. Ünal, Ü. Çiftçi and N.N. Urgan, "Bir Gizli Katmanlı Yapay Sinir Ağlarında Optimal Nöron Sayısının İncelenmesi", Süleyman Demirel Üniversitesi Fen Edebiyat Fakültesi Fen Dergisi, vol. 17, no. 2, pp. 303-325, 2022.

[26] L.N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1-learning rate, batch size, momentum, and weight decay", arXiv preprint arXiv:1803.09820, 2018.

[27] G. Ser and C.T. Bati, "Derin sinir ağları ile en iyi modelin belirlenmesi: mantar verileri üzerine Keras uygulaması", Yuzuncu Yıl University Journal of Agricultural Sciences, vol. 29, no. 3, pp. 406-417, 2019.

[28] D. Kulshrestha, "Activation Functions in Machine Learning", url: https://iq.opengenus.org/activation-functions-ml/, (Access Date: 03.12.2023).

[29] S. Kılıçarslan, C. Közkurt, S. Baş and A. Elen, "Detection and classification of pneumonia using novel Superior Exponential (SupEx) activation function in convolutional neural networks", Expert Systems with Applications, vol. 217, no. 119503, 2023.

[30] S. Kılıçarslan and M. Çelik, "RSigELU: A nonlinear activation function for deep neural networks", Expert Systems with Applications, vol. 174, no. 114805, 2021.

[31] M.A. Bülbül, "Optimization of artificial neural network structure and hyperparameters in hybrid model by genetic algorithm: iOS–android application for breast cancer diagnosis/prediction", The Journal of Supercomputing, pp. 1-21, 2023.

[32] R. Sun, "Optimization for deep learning: theory and algorithms", arXiv preprint arXiv:1912.08957, 2019.

[33] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures", Neural Networks: Tricks of the Trade: Second Edition, pp. 437-478, 2012.