SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 5, NO. 3, DECEMBER 2022
DOI: 10.35377/saucis.05.03.1197119

Research Article

A Deep Transfer Learning-Based Comparative Study for Detection of Malaria Disease

Emel Soylu[1]

[1]Corresponding Author; Samsun University, Faculty of Engineering, Department of Software Engineering, Samsun/TURKEY; emel.soylu@samsun.edu.tr

Received 31 October 2022; Revised 8 November 2022; Accepted 30 November 2022; Published online 31 December 2022

Abstract

Malaria is a disease caused by a parasite. The parasite is transmitted to humans through the bite of infected mosquitoes. Thousands of people die every year due to malaria. When this disease is diagnosed early, it can be fully treated with medication. Diagnosis of malaria can be made according to the presence of parasites in the blood taken from the patient. In this study, malaria detection and diagnosis study were performed using The Malaria dataset containing a total of 27,558 cell images with samples of equally parasitized and uninfected cells from thin blood smear slide images of segmented cells. It is possible to detect malaria from microscopic blood smear images via modern deep learning techniques. In this study, 5 of the popular convolutional neural network architectures for malaria detection from cell images were retrained to find the best combination of architecture and learning algorithm. AlexNet, GoogLeNet, ResNet-50, MobileNet-v2, VGG-16 architectures from pre-trained networks were used, their hyperparameters were adjusted and their performances were compared. In this study, a maximum 96.53% accuracy rate was achieved with MobileNet-v2 architecture using the adam learning algorithm.

Keywords: malaria detection, deep transfer learning, Matlab, Convolutional Neural Network

1. Introduction

Malaria; is a disease that is transmitted to humans by the bite of a mosquito that carries parasites, can be fatal if not treated in time, and causes fever and chills in seizures. Anemia and jaundice may develop in cases where diagnosis and treatment are delayed. In some types of parasites that cause malaria, if treatment is not started within 24 hours, it can progress and lead to death. Malaria is a disease that can be treated with drugs. If the disease is diagnosed early and treated appropriately, patients can fully recover [1].

According to The World Health Organization's report, globally 229 million malaria cases were estimated. It is more common, especially in the Africa region. In 2019, 409 thousand people died from malaria disease. In the 2000s, this number was 736 thousand. Between 2000 and 2019, there were 1.5 billion cases of malaria globally, and 7.6 million people died from this cause [2]. Because malaria causes so much illness and death, the disease is a huge burden for many national economies. Most of the countries with malaria are already among the poor countries, and the economy of these countries is badly affected by the disease. The malaria parasite resides in the red blood cells of an infected person. Malaria can also be transmitted through blood transfusions, organ transplants, or the shared use of needles or syringes. Malaria can also be transmitted to an unborn baby [3].

Artificial intelligence techniques are effective methods that can produce solutions to optimization, prediction, fault diagnosis, image processing problems [4]–[6]. Artificial intelligence has entered a new phase with the start of running multi-layer neural networks on graphics cards. Thus, it became easier to solve problems such as image processing where too much data is processed. Multilayer deep convolutional neural networks produce very successful results for image processing problems [7]–[10]. In classical image processing, feature extraction is required to determine representation from the image. In contrast to this situation, raw pixel values are used in the CNN (Convolutional Neural Network) model. Deep learning techniques are also successfully used in the diagnosis of diseases in the field of health [11].

In the last decades, great advances have been made in the diagnosis of disease from medical images. Researchers have developed techniques that produce highly accurate results with various image processing operations. Examples of these techniques are artificial neural networks, machine learning, and deep learning techniques. According to the type of disease, images can be obtained and analyzed from sources such as microscopic, x-ray, MR, and ECG, and computer-assisted disease diagnosis can be made [12]–[17].

Training a deep network from scratch takes a lot of time. Retraining a previously trained model saves time. Re-training of a previously trained class network with a new data set is called DTL (Deep Transfer Learning). Using DTL provides a great advantage. There are many deep network models currently developed. These networks have been obtained as a result of days-long training on very powerful computers using millions of data [18]. Fig. 1. shows how DTL works [19].

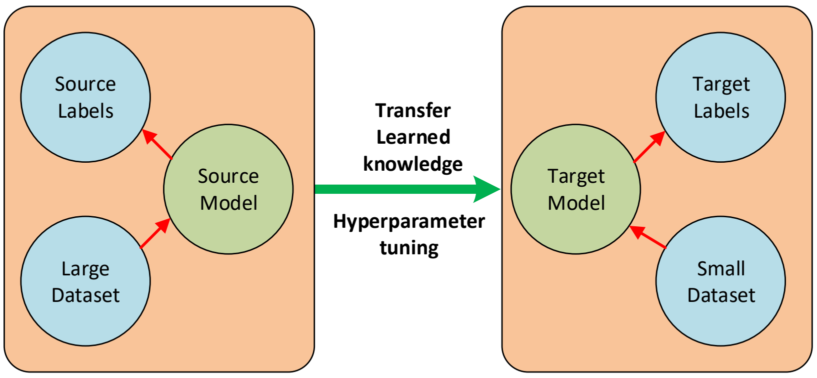


Figure 1 Concept of DTL

In this study, five of the deep network models available in the Matlab environment were retrained for the malaria data set. These are AlexNet, GoogLeNet, ResNet-50, MobileNet-v2, VGG-16 models. When previous studies were examined, no study was found to compare these five architectures. In this study, it has been seen that high performance can be achieved with DTL when hyperparameters are adjusted appropriately.

Information about the data set and deep learning architectures used in the rest of the study was given. After the hyperparameter settings are made, the training and comparison results are given.

2. Relevant Work

Scientists have done many studies around the world on the detection of the deadly malaria disease. With the technological improvements in computer hardware, software styles based on running parallel programs on graphics cards, great progress has been made in image processing studies. Image processing techniques have become frequently used in the field of health sciences, especially in disease detection. Deep artificial neural network techniques with a high number of neurons and layers are the technique with the highest performance today. Unlike machine learning, deep learning architectures with millions of parameters do not have feature extraction. The fact that high-performance results without a laborious process such as feature extraction increases the preference rate of convolutional neural networks by people.

Existing deep learning approaches applied to Malaria detection are given in Table 1. Vijayalakshmi et al. Retrained the Visual Geometry Group (VGG) by replacing some of its layers with the Support Vector Machine (SVM) and achieved 93.1% accuracy in malaria detection [20]. Dong et al. In their study on Identification of Malaria Infected Cells, they used transfer-based deep learning techniques and achieved a success rate of over 95% [21]. Yang et al. created a dataset with 1819 thick smear images collected from 150 patients. They reached a maximum of 94.33% accuracy in their malaria detection study using deep learning techniques on this data set [22]. According to Pan et al., It demonstrated that the deep convolutional network based on LeNet-5 can achieve very high classification accuracies for automatic malaria diagnosis. They analyzed the performance effect of the dataset by running their method on datasets containing different numbers of images [23]. Reddy et al. reached a 95.91% accuracy rate with ResNet50 architecture for malaria detection. They used a dataset containing 27558 images [24]. Fuhad et al. used a variety of techniques including knowledge distillation, data augmentation, autoencoder, feature extraction by a CNN model, and classified by Support Vector Machine (SVM) or K-Nearest Neighbors (KNN). They reached a 99.23% accuracy rate for malaria detection problems with training the network using 32 × 32 images.

Table 1 Existing deep learning approaches applied to Malaria detection

Authors	**Methods**	**Year**	**Images**	**Best Accuracy rate (%)**
Reddy et al. [24]	ResNet50	2019	27558	95,91
Fuhad et al. [25]	CNN-SVM, CNN-KNN	2020	26161	99,23
Vijayalakshmi et al. [20]	Visual Geometry Group (VGG) network and Support Vector Machine (SVM)	2019	2550	93,1
Dong et al. [21]	LeNet, AlexNet and GoogLeNet	2017	2565	98,13
Yang et al. [22]	ResNet50, VGG19, AlexNet, CNN	2020	1443	93,46
Pan et al. [23]	LeNet-5	2018	800	99

In this study, different from the others, 5 types of methods were compared according to the learning algorithm. High performance has been achieved without applying pre-image processing techniques in the dataset. The data set was applied directly to the input of the networks. Each architecture has been tested for 3 types of learning algorithms for two different initial learning rates (lr). The effect of learning algorithm selection on success was investigated. After 2x15 re-network training, it was observed that the success rates were between 50% and 96.53%.

3. Materials and Methods

The technical features of the computer used in this study are as follows:

- GPU: NVIDIA GeForce GTX 1070 8GB
- CPU: Intel I7 3.4 GHz
- Ram: 12 GB
- Operating System: Windows 10, 64 bit

The entire study has been done in the Matlab development environment using Deep Network Designer application. The last layer of the models is updated according to the number of categories, parameter settings are made and training of the network is carried out.

One of the most popular types of deep neural networks is the Convolutional Neural Network (CNN). CNN has a very good performance especially when a lot of images need to be processed. CNN has more than one layer. These are the convolutional layer, non-linearity layer, pooling layer, fully connected layer [26].

The first layer is the convolutional layer. In this layer, the image is passed through more than one parallel convolutional filter. These filters act as feature extractors. The output of the filters is a feature map [27]. The nonlinear transform layer normalizes between nearby feature maps [28]. With the pooling layer, the number of parameters and dimensions of the network is reduced. In the fully connected layer, data from previous layers are combined by weighting, and thanks to a loss function, the optimal weight to be given to neurons during training is found. Usually, the softmax activation function is used in this layer, and classification is made as probabilistic [29].

Three different optimizers are available in Matlab deep network designer tool as Stochastic Gradient Descent with Momentum(sgdm), Adaptive Moment Estimation (adam), and Root Mean Square Propagation (rmsprop). In this study their performance according to network models are compared. Sgdm is the preferred optimization method for many large-scale learning problems. Adam is a form of optimization that can be used instead of stochastic gradient descent. Quickly achieving good results on net weights makes this method popular. Rmsprop divides the learning rate by an exponentially decreasing square gradient average. Rmsprop produces effective results for online and unstable problems [30].

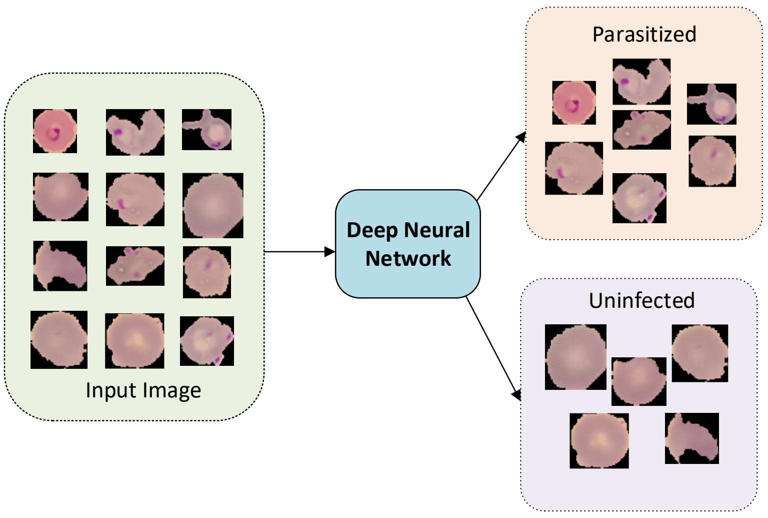


Figure 2 Block Diagram of the System

In this study, pre-trained networks are used to detect malaria disease. In deep learning, retraining a previously trained model for another problem is called transfer-based deep learning. Transfer learning is an approach in deep learning where knowledge is transferred from one model to another. The information obtained from the pre-trained model that was previously trained with a large-scale data can be used in a new model. Using a pre-trained network, especially for applications where the number of data is low, produces satisfactory results in the field of deep learning. When transfer-based learning is used, it is necessary to make changes in the last layers according to the number of classes to be classified, and to retrain the model after making the hyperparameter settings [31].

The block diagram of the system is given in Fig. 2. Deep network classifies input image as infected from parasite or not. The used networks and their properties are given in Table 2. In this study there are two classes, so the last classification layers of network models are modified to classify new images. For comparison when setting training parameters, batch size and epoch number set the same.

Table 2 Properties of network models

Network	Year of development	Input Image Size	Depth	Number of parameters	Number of categories
AlexNet	2012	224x224x3	8	61 million	1000
GoogLeNet	2014	224x224x3	22	7 million	1000
VGG-16	2014	224x224x3	16	138 million	1000
ResNet-50	2015	224x224x3	50	25,6 million	1000
MobileNet-v2	2018	224x224x3	53	3,5 million	1000

3.1. AlexNet

AlexNet is a convolutional neural network with 8-layer deep CNN. It has been trained with 1.2 million images in the ImageNet dataset and can classify 1000 objects [32]. AlexNet is designed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton [33]. The detailed configuration of AlexNet model for this study is given in Table 3.

Table 3 The architecture of AlexNet model

No	Layer	Properties	No	Layer	Properties
1	Image Input	227×227×3	14	2-D Grouped Conv.	2 groups of 128 3×3×192
2	2-D Conv.	96 11×11×3	15	ReLU	ReLU
3	ReLU	ReLU	16	2-D Max Pooling	3×3
4	Cross Channel Norm.	5 channels per element	17	Fully Connected	4096
5	2-D Max Pooling	3×3	18	ReLU	ReLU
6	2-D Grouped Conv.	2 groups of 128 5×5×48	19	Dropout	50% dropout
7	ReLU	ReLU	20	Fully Connected	4096
8	Cross Channel Norm.	5 channels per element	21	ReLU	ReLU
9	2-D Max Pooling	3×3	22	Dropout	50% dropout
10	2-D Conv.	384 3×3×256	23	Fully Connected	2
11	ReLU	ReLU	24	Softmax	softmax
12	2-D Grouped Conv.	2 groups of 192 3×3×192	25	Classification Output	2 classes
13	ReLU	ReLU			

3.2. GoogLeNet

GoogLeNet is developed by researchers working at Google. GoogLeNet was the winner of ILSVRC 2014 competition [34]. GoogLeNet's other name is Inception block. It has a 22-layer deep CNN and 7 million parameters. Pre-trained network can classify 1000 objects. The detailed configuration of GoogLeNet model for this study is given in Table 4.

Table 4 The architecture of GoogLeNet model

No	Layer	Properties	No	Layer	Properties	No	Layer	Properties
1	Image Input	224×224×3	49	2-D Convolution	48 5×5×16	97	2-D Convolution	256 1×1×528
2	2-D Convolution	64 7×7×3	50	ReLU	ReLU	98	ReLU	ReLU
3	ReLU	ReLU	51	2-D Max Pooling	3×3	99	2-D Convolution	160 1×1×528
4	2-D Max Pooling	3×3	52	2-D Convolution	64 1×1×480	10 0	ReLU	ReLU
5	Cross Channel Norm.	channels per element	53	ReLU	ReLU	10 1	2-D Convolution	320 3×3×160
6	2-D Convolution	64 1×1×64	54	Depth concatenatio n	4 inputs	10 2	ReLU	ReLU
7	ReLU	ReLU	55	2-D Convolution	160 1×1×512	10 3	2-D Convolution	32 1×1×528
8	2-D Convolution	192 3×3×64	56	ReLU	ReLU	10 4	ReLU	ReLU
9	ReLU	ReLU	57	2-D Convolution	112 1×1×512	10 5	2-D Convolution	128 5×5×32
10	Cross Channel Norm.	5 channels per element	58	ReLU	ReLU	10 6	ReLU	ReLU
11	2-D Max Pooling	3×3	59	2-D Convolution	224 3×3×112	10 7	2-D Max Pooling	3×3

12	2-D Convolution	64 1×1×192	60	ReLU	ReLU	108	2-D Convolution	128 1×1×528
13	ReLU	ReLU	61	2-D Convolution	24 1×1×512	109	ReLU	ReLU
14	2-D Convolution	96 1×1×192	62	ReLU	ReLU	110	Depth concatenation	4 inputs
15	ReLU	ReLU	63	2-D Convolution	64 5×5×24	111	2-D Max Pooling	3×3
16	2-D Convolution	128 3×3×96	64	ReLU	ReLU	112	2-D Convolution	256 1×1×832
17	ReLU	ReLU	65	2-D Max Pooling	3×3	113	ReLU	ReLU
18	2-D Convolution	16 1×1×192	66	2-D Convolution	64 1×1×512	114	2-D Convolution	160 1×1×832
19	ReLU	ReLU	67	ReLU	ReLU	115	ReLU	ReLU
20	2-D Convolution	32 5×5×16	68	Depth concatenation	4 inputs	116	2-D Convolution	320 3×3×160
21	ReLU	ReLU	69	2-D Convolution	128 1×1×512	117	ReLU	ReLU
22	2-D Max Pooling	3×3	70	ReLU	ReLU	118	2-D Convolution	32 1×1×832
23	2-D Convolution	32 1×1×192	71	2-D Convolution	128 1×1×512	119	ReLU	ReLU
24	ReLU	ReLU	72	ReLU	ReLU	120	2-D Convolution	128 5×5×32
25	Depth concatenation	4 inputs	73	2-D Convolution	256 3×3×128	121	ReLU	ReLU
26	2-D Convolution	128 1×1×256	74	ReLU	ReLU	122	2-D Max Pooling	3×3
27	ReLU	ReLU	75	2-D Convolution	24 1×1×512	123	2-D Convolution	128 1×1×832
28	2-D Convolution	128 1×1×256	76	ReLU	ReLU	124	ReLU	ReLU
29	ReLU	ReLU	77	2-D Convolution	64 5×5×24	125	Depth concatenation	4 inputs
30	2-D Convolution	192 3×3×128	78	ReLU	ReLU	126	2-D Convolution	384 1×1×832
31	ReLU	ReLU	79	2-D Max Pooling	3×3	127	ReLU	ReLU
32	2-D Convolution	32 1×1×256	80	2-D Convolution	64 1×1×512	128	2-D Convolution	192 1×1×832
33	ReLU	ReLU	81	ReLU	ReLU	129	ReLU	ReLU
34	2-D Convolution	96 5×5×32	82	Depth concatenation	4 inputs	130	2-D Convolution	384 3×3×192
35	ReLU	ReLU	83	2-D Convolution	112 1×1×512	131	ReLU	ReLU
36	2-D Max Pooling	3×3	84	ReLU	ReLU	132	2-D Convolution	48 1×1×832
37	2-D Convolution	64 1×1×256	85	2-D Convolution	144 1×1×512	133	ReLU	ReLU
38	ReLU	ReLU	86	ReLU	ReLU	134	2-D Convolution	128 5×5×48

39	Depth concatenation	4 inputs	87	2-D Convolution	288 3×3×144	135	ReLU	ReLU
40	2-D Max Pooling	3×3	88	ReLU	ReLU	136	2-D Max Pooling	3×3
41	2-D Convolution	192 1×1×480	89	2-D Convolution	32 1×1×512	137	2-D Convolution	128 1×1×832
42	ReLU	ReLU	90	ReLU	ReLU	138	ReLU	ReLU
43	2-D Convolution	96 1×1×480	91	2-D Convolution	64 5×5×32	139	Depth concatenation	4 inputs
44	ReLU	ReLU	92	ReLU	ReLU	140	2-D Global Avg. Pooling	2-D
45	2-D Convolution	208 3×3×96	93	2-D Max Pooling	3×3	141	Dropout	40% dropout
46	ReLU	ReLU	94	2-D Convolution	64 1×1×512	142	Fully Connected	2
47	2-D Convolution	16 1×1×480	95	ReLU	ReLU	143	Softmax	softmax
48	ReLU	ReLU	96	Depth concatenation	4 inputs	144	Classification Output	2 classes

3.3. ResNet-50

ResNet stands for Residual Network introduced in the 2015 p by He Kaiming et. al.[35] ResNet50 is a CNN architecture with 50-layer deep CNN. Pre-trained network can classify into 1000 categories. The architecture has 25.6 million parameters. The detailed configuration of ResNet-50 model for this study is given in Table 5.

Table 5 The architecture of ResNet-50 model

No	Layer	Properties	No	Layer	Properties	No	Layer	Properties
1	Image Input	224×224×3	60	2-D Conv.	128 1×1×512	119	Batch Norm.	1024 channels
2	2-D Conv.	64 7×7×3	61	Batch Norm.	128 channels	120	Addition	2 inputs
3	Batch Norm.	64 channels	62	ReLU	ReLU	121	ReLU	ReLU
4	ReLU	ReLU	63	2-D Conv.	128 3×3×128	122	2-D Conv.	256 1×1×1024
5	2-D Max Pooling	3×3	64	Batch Norm.	128 channels	123	Batch Norm.	256 channels
6	2-D Conv.	64 1×1×64	65	ReLU	ReLU	124	ReLU	ReLU
7	Batch Norm.	64 channels	66	2-D Conv.	512 1×1×128	125	2-D Conv.	256 3×3×256
8	ReLU	ReLU	67	Batch Norm.	512 channels	126	Batch Norm.	256 channels
9	2-D Conv.	64 3×3×64	68	Addition	2 inputs	127	ReLU	ReLU
10	Batch Norm.	64 channels	69	ReLU	ReLU	128	2-D Conv.	1024 1×1×256
11	ReLU	ReLU	70	2-D Conv.	128 1×1×512	129	Batch Norm.	1024 channels
12	2-D Conv.	256 1×1×64	71	Batch Norm.	128 channels	130	Addition	2 inputs
13	2-D Conv.	256 1×1×64	72	ReLU	ReLU	131	ReLU	ReLU

14	Batch Norm.	256 channels	73	2-D Conv.	128 3×3×128	132	2-D Conv.	256 1×1×1024
15	Batch Norm.	256 channels	74	Batch Norm.	128 channels	133	Batch Norm.	256 channels
16	Addition	2 inputs	75	ReLU	ReLU	134	ReLU	ReLU
17	ReLU	ReLU	76	2-D Conv.	512 1×1×128	135	2-D Conv.	256 3×3×256
18	2-D Conv.	64 1×1×256	77	Batch Norm.	512 channels	136	Batch Norm.	256 channels
19	Batch Norm.	64 channels	78	Addition	2 inputs	137	ReLU	ReLU
20	ReLU	ReLU	79	ReLU	ReLU	138	2-D Conv.	1024 1×1×256
21	2-D Conv.	64 3×3×64	80	2-D Conv.	256 1×1×512	139	Batch Norm.	1024 channels
22	Batch Norm.	64 channels	81	Batch Norm.	256 channels	140	Addition	2 inputs
23	ReLU	ReLU	82	ReLU	ReLU	141	ReLU	ReLU
24	2-D Conv.	256 1×1×64	83	2-D Conv.	256 3×3×256	142	2-D Conv.	512 1×1×1024
25	Batch Norm.	256 channels	84	Batch Norm.	256 channels	143	Batch Norm.	512 channels
26	Addition	2 inputs	85	ReLU	ReLU	144	ReLU	ReLU
27	ReLU	ReLU	86	2-D Conv.	1024 1×1×256	145	2-D Conv.	512 3×3×512
28	2-D Conv.	64 1×1×256	87	2-D Conv.	1024 1×1×512	146	Batch Norm.	512 channels
29	Batch Norm.	64 channels	88	Batch Norm.	1024 channels	147	ReLU	ReLU
30	ReLU	ReLU	89	Batch Norm.	1024 channels	148	2-D Conv.	2048 1×1×512
31	2-D Conv.	64 3×3×64	90	Addition	2 inputs	149	2-D Conv.	2048 1×1×1024
32	Batch Norm.	64 channels	91	ReLU	ReLU	150	Batch Norm.	2048 channels
33	ReLU	ReLU	92	2-D Conv.	256 1×1×1024	151	Batch Norm.	2048 channels
34	2-D Conv.	256 1×1×64	93	Batch Norm.	256 channels	152	Addition	2 inputs
35	Batch Norm.	256 channels	94	ReLU	ReLU	153	ReLU	ReLU
36	Addition	2 inputs	95	2-D Conv.	256 3×3×256	154	2-D Conv.	512 1×1×2048
37	ReLU	ReLU	96	Batch Norm.	256 channels	155	Batch Norm.	512 channels
38	2-D Conv.	128 1×1×256	97	ReLU	ReLU	156	ReLU	ReLU
39	Batch Norm.	128 channels	98	2-D Conv.	1024 1×1×256	157	2-D Conv.	512 3×3×512
40	ReLU	ReLU	99	Batch Norm.	1024 channels	158	Batch Norm.	512 channels
41	2-D Conv.	128 3×3×128	100	Addition	2 inputs	159	ReLU	ReLU
42	Batch Norm.	128 channels	101	ReLU	ReLU	160	2-D Conv.	2048 1×1×512
43	ReLU	ReLU	102	2-D Conv.	256 1×1×1024	161	Batch Norm.	2048 channels

44	2-D Conv.	512 1×1×128	103	Batch Norm.	256 channels	162	Addition	2 inputs
45	2-D Conv.	512 1×1×256	104	ReLU	ReLU	163	ReLU	ReLU
46	Batch Norm.	512 channels	105	2-D Conv.	256 3×3×256	164	2-D Conv.	512 1×1×2048
47	Batch Norm.	512 channels	106	Batch Norm.	256 channels	165	Batch Norm.	512 channels
48	Addition	2 inputs	107	ReLU	ReLU	166	ReLU	ReLU
49	ReLU	ReLU	108	2-D Conv.	1024 1×1×256	167	2-D Conv.	512 3×3×512
50	2-D Conv.	128 1×1×512	109	Batch Norm.	1024 channels	168	Batch Norm.	512 channels
51	Batch Norm.	128 channels	110	Addition	2 inputs	169	ReLU	ReLU
52	ReLU	ReLU	111	ReLU	ReLU	170	2-D Conv.	2048 1×1×512
53	2-D Conv.	128 3×3×128	112	2-D Conv.	256 1×1×1024	171	Batch Norm.	2048 channels
54	Batch Norm.	128 channels	113	Batch Norm.	256 channels	172	Addition	2 inputs
55	ReLU	ReLU	114	ReLU	ReLU	173	ReLU	ReLU
56	2-D Conv.	512 1×1×128	115	2-D Conv.	256 3×3×256	174	2-D Global Average Pooling	2-D
57	Batch Norm.	512 channels	116	Batch Norm.	256 channels	175	Fully Connected	Fully Connected
58	Addition	2 inputs	117	ReLU	ReLU	176	Softmax	softmax
59	ReLU	ReLU	118	2-D Conv.	1024 1×1×256	177	Classification Output	2 classes

3.4. MobileNet-v2

MobileNet-v2 has an architecture designed to be used mostly on mobile devices. With 3.5 million parameters, it has fewer parameters than other architectures. It has 53-layer deep CNN. It is trained with over a million data from the ImageNet dataset. The pre-trained network can classify into 1000 categories. The low number of parameters also reduces the training time. The detailed configuration of MobileNet-v2 model for this study is given in Table 6.

Table 6 The architecture of MobileNet-v2 model

Layer	Properties	No	Layer	Properties	No	Layer	Properties
Image Input	224×224×3	53	2-D Conv.	192 1×1×32	105	2-D Conv.	576 1×1×96
2-D Conv.	32 3×3×3	54	Batch Norm.	192 channels	106	Batch Norm.	576 channels
Batch Norm.	32 channels	55	Clipped ReLU	ceiling 6	107	Clipped ReLU	ceiling 6
Clipped ReLU	ceiling 6	56	2-D Grouped Conv.	192 groups of 1 3×3×1	108	2-D Grouped Conv.	576 groups
2-D Grouped Conv.	32 groups	57	Batch Norm.	192 channels	109	Batch Norm.	576 channels
Batch Norm.	32 channels	58	Clipped ReLU	ceiling 6	110	Clipped ReLU	ceiling 6
Clipped ReLU	ceiling 6	59	2-D Conv.	64 1×1×192	111	2-D Conv.	96 1×1×576
2-D Conv.	16 1×1×32	60	Batch Norm.	64 channels	112	Batch Norm.	96 channels

Batch Norm.	16 channels	61	2-D Conv.	384 1×1×64	113	Addition	2 inputs
2-D Conv.	96 1×1×16	62	Batch Norm.	384 channels	114	2-D Conv.	576 1×1×96
Batch Norm.	96 channels	63	Clipped ReLU	ceiling 6	115	Batch Norm.	576 channels
Clipped ReLU	ceiling 6	64	2-D Grouped Conv.	384 groups	116	Clipped ReLU	ceiling 6
2-D Grouped Conv.	96 groups of 1 3×3×1	65	Batch Norm.	384 channels	117	2-D Grouped Conv.	576 groups of 1 3×3×1
Batch Norm.	96 channels	66	Clipped ReLU	ceiling 6	118	Batch Norm.	576 channels
Clipped ReLU	ceiling 6	67	2-D Conv.	64 1×1×384	119	Clipped ReLU	ceiling 6
2-D Conv.	24 1×1×96	68	Batch Norm.	64 channels	120	2-D Conv.	160 1×1×576
Batch Norm.	24 channels	69	Addition	2 inputs	121	Batch Norm.	160 channels
2-D Conv.	144 1×1×24	70	2-D Conv.	384 1×1×64	122	2-D Conv.	960 1×1×160
Batch Norm.	144 channels	71	Batch Norm.	384 channels	123	Batch Norm.	960 channels
Clipped ReLU	ceiling 6	72	Clipped ReLU	ceiling 6	124	Clipped ReLU	ceiling 6
2-D Grouped Conv.	144 groups	73	2-D Grouped Conv.	384 groups	125	2-D Grouped Conv.	960 groups
Batch Norm.	144 channels	74	Batch Norm.	384 channels	126	Batch Norm.	960 channels
Clipped ReLU	ceiling 6	75	Clipped ReLU	ceiling 6	127	Clipped ReLU	ceiling 6
2-D Conv.	24 1×1×144	76	2-D Conv.	64 1×1×384	128	2-D Conv.	160 1×1×960
Batch Norm.	24 channels	77	Batch Norm.	64 channels	129	Batch Norm.	160 channels
Addition	2 inputs	78	Addition	2 inputs	130	Addition	2 inputs
2-D Conv.	144 1×1×24	79	2-D Conv.	384 1×1×64	131	2-D Conv.	960 1×1×160
Batch Norm.	144 channels	80	Batch Norm.	384 channels	132	Batch Norm.	960 channels
Clipped ReLU	ceiling 6	81	Clipped ReLU	ceiling 6	133	Clipped ReLU	ceiling 6
2-D Grouped Conv.	144 groups of 1 3×3×1	82	2-D Grouped Conv.	384 groups	134	2-D Grouped Conv.	960 groups
Batch Norm.	144 channels	83	Batch Norm.	384 channels	135	Batch Norm.	960 channels
Clipped ReLU	ceiling 6	84	Clipped ReLU	ceiling 6	136	Clipped ReLU	ceiling 6
2-D Conv.	32 1×1×144	85	2-D Conv.	64 1×1×384	137	2-D Conv.	160 1×1×960
Batch Norm.	32 channels	86	Batch Norm.	64 channels	138	Batch Norm.	160 channels
2-D Conv.	192 1×1×32	87	Addition	2 inputs	139	Addition	2 inputs
Batch Norm.	192 channels	88	2-D Conv.	384 1×1×64	140	2-D Conv.	960 1×1×160

Clipped ReLU	ceiling 6	89	Batch Norm.	384 channels	141	Batch Norm.	960 channels
2-D Grouped Conv.	192 groups	90	Clipped ReLU	ceiling 6	142	Clipped ReLU	ceiling 6
Batch Norm.	192 channels	91	2-D Grouped Conv.	384 groups	143	2-D Grouped Conv.	960 groups
Clipped ReLU	ceiling 6	92	Batch Norm.	384 channels	144	Batch Norm.	960 channels
2-D Conv.	32 1×1×192	93	Clipped ReLU	ceiling 6	145	Clipped ReLU	ceiling 6
Batch Norm.	32 channels	94	2-D Conv.	96 1×1×384	146	2-D Conv.	320 1×1×960
Addition	2 inputs	95	Batch Norm.	96 channels	147	Batch Norm.	320 channels
2-D Conv.	192 1×1×32	96	2-D Conv.	576 1×1×96	148	2-D Conv.	1280 1×1×320
Batch Norm.	192 channels	97	Batch Norm.	576 channels	149	Batch Norm.	1280 channels
Clipped ReLU	ceiling 6	98	Clipped ReLU	ceiling 6	150	Clipped ReLU	ceiling 6
2-D Grouped Conv.	192 groups	99	2-D Grouped Conv.	576 groups	151	2-D Global Average Pooling	2-D global average pooling
Batch Norm.	192 channels	100	Batch Norm.	576 channels	152	Fully Connected	Fully connected
Clipped ReLU	ceiling 6	101	Clipped ReLU	ceiling 6	153	Softmax	softmax
2-D Conv.	32 1×1×192	102	2-D Conv.	96 1×1×576	154	Classification Output	2 classes
Batch Norm.	32 channels	103	Batch Norm.	96 channels			
Addition	2 inputs	104	Addition	2 inputs			

3.5. VGG-16

It is trained with more than 14 million data in the VGG-16 ImageNet dataset. It’s training took weeks. It has 41 layers. With 138 million parameters, it is the architecture with the most parameters among those used in this study. The pre-trained network can classify into 1000 categories. The detailed configuration of VGG-16 model for this study is given in Table 7.

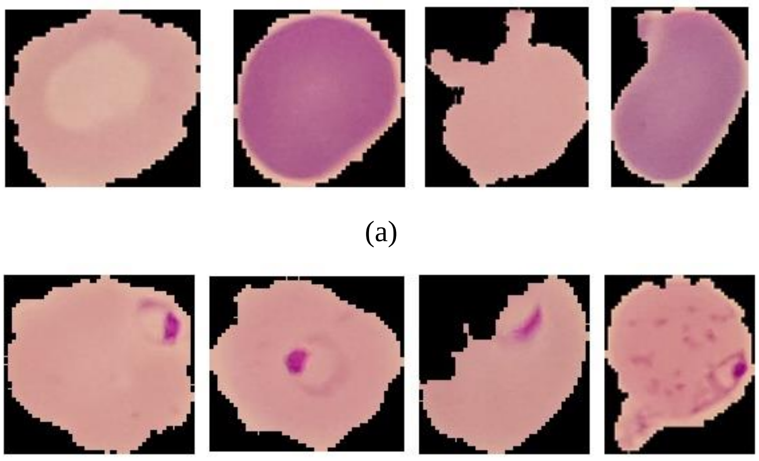
Table 7 The architecture of VGG-16 model

No	**Layer**	**Properties**	**No**	**Layer**	**Properties**	**No**	**Layer**	**Properties**
1	Image Input	224x224x3	15	ReLU	ReLU	29	ReLU	ReLU
2	Convolution	64 3x3x3	16	Convolution	256 3x3x256	30	Convolution	512 3x3x512
3	ReLU	ReLU	17	ReLU	ReLU	31	ReLU	ReLU
4	Convolution	64 3x3x64	18	Max Pooling	2x2	32	Max Pooling	2x2
5	ReLU	ReLU	19	Convolution	512 3x3x256	33	Fully Connected	4096
6	Max Pooling	2x2	20	ReLU	ReLU	34	ReLU	ReLU
7	Convolution	128 3x3x64	21	Convolution	512 3x3x512	35	Dropout	50% dropout
8	ReLU	ReLU	22	ReLU	ReLU	36	Fully Connected	4096

9	Convolution	128 3x3x128	23	Convolution	512 3x3x512	37	ReLU	ReLU
10	ReLU	ReLU	24	ReLU	ReLU	38	Dropout	50% dropout
11	Max Pooling	2x2	25	Max Pooling	2x2	39	Fully Connected	2
12	Convolution	256 3x3x128	26	Convolution	512 3x3x512	40	Softmax	softmax
13	ReLU	ReLU	27	ReLU	ReLU	41	Classification Output	2 classes
14	Convolution	256 3x3x256	28	Convolution	512 3x3x512			

3.6. Dataset

The data set used in this study was created with a mobile application developed to take microscopic images and samples taken from patients and non-sick individuals in Mahidol-Oxford Tropical Medicine Research Unit in Bangkok [36]. The data set was shared on the internet available to researchers. It is possible to reach the data set from many different links. In this study, the data obtained from the Kaggle platform was used [37]. The Malaria dataset contains a total of 27,558 cell images with samples of equally parasitized and uninfected cells from thin blood smear slide images of segmented cells. Sample images from dataset is given in Figure 3. Parasitized cells contain Plasmodium in different sizes and shapes.


(a)

(b)

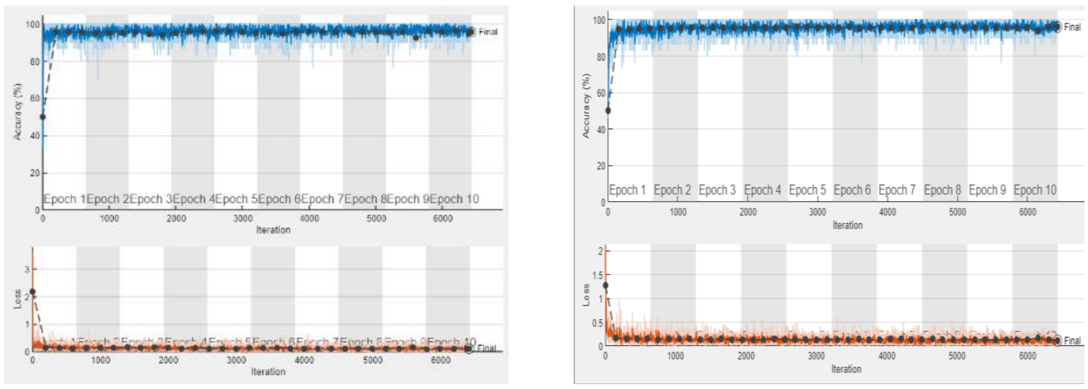
Figure 3 Sample dataset images (a) uninfected (b) parasitized

The image sizes in the data set are resized according to the input sizes of the network to be used. In this study data augmentation was not applied. 70% of the data were used as training data and 30% as test data. Number of images for training is 19290 and number of images for validation is 8268. Number of parasitized and uninfected images are equal.

4. Training of models

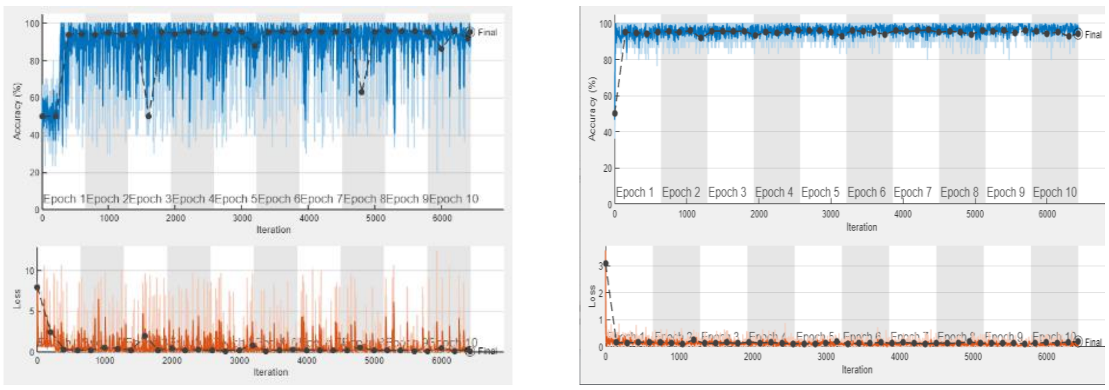
Learning curves that showing the progress over the experience during the training of a machine learning models are just a mathematical representation of the learning process. We observe accuracy and loss performances from plots according to validation data. In this section training progress of models are given.

Screenshots of the training window for AlexNet are given in Fig. 4, Fig. 5, and Fig.6 respectively. Accuracy and loss rates according to iteration are shown in these graphs. The validation accuracy is obtained 95,9% with sgdm optimizer, 50% with adam optimizer, 95,22% with rmsprop optimizer at learning rate of 0.001 and 96.08% with sgdm optimizer, 94.19% with adam optimizer, 95.85% with rmsprop optimizer learning rate of 0.0001.



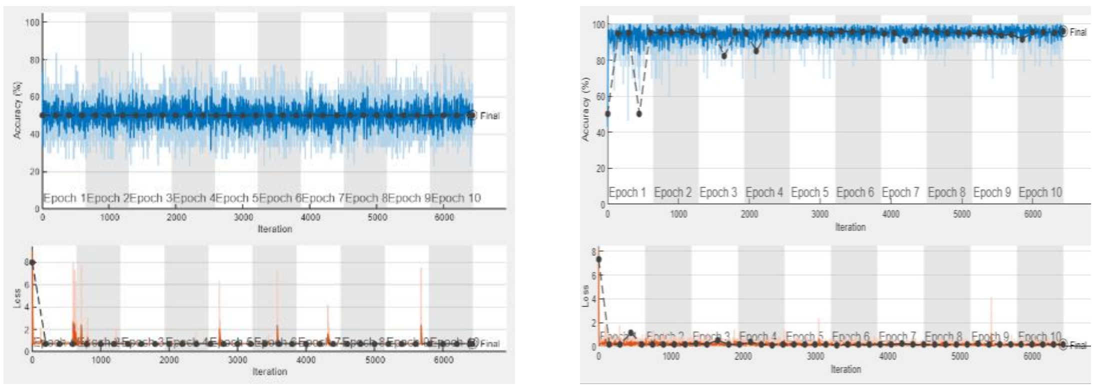
(a) (b)

Figure 4 Re-training of AlexNet Network Model with sgdm Optimizer (a)lr=0.001 (b) lr=0.0001



(a) (b)

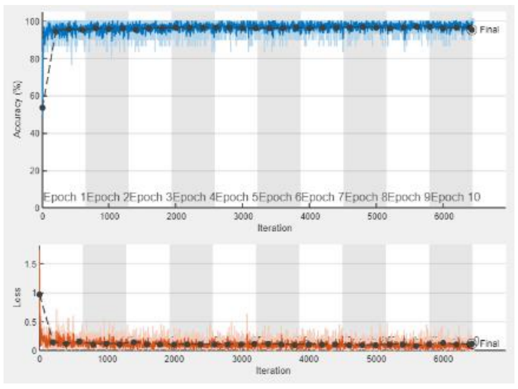
Figure 5 Re-training of AlexNet Network Model with adam Optimizer (a)lr=0.001 (b) lr=0.0001



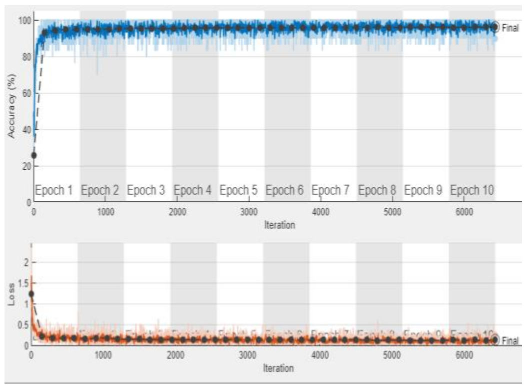
(a) (b)

Figure 6 Re-training of AlexNet Network Model with rmsprop Optimizer (a)lr=0.001 (b) lr=0.0001

Screenshots of the training window for GoogLeNet are given in Fig. 7, Fig. 8, and Fig.9 respectively. Accuracy and loss rates according to iteration are shown in these graphs. Accuracy and loss rates according to iteration are shown in these graphs. The validation accuracy is obtained 95,22% with sgdm optimizer, 95,46% with adam optimizer, 95,54% with rmsprop optimizer and 96.07% with sgdm optimizer, 96.26% with adam optimizer, 96.73% with rmsprop optimizer learning rate of 0.0001.

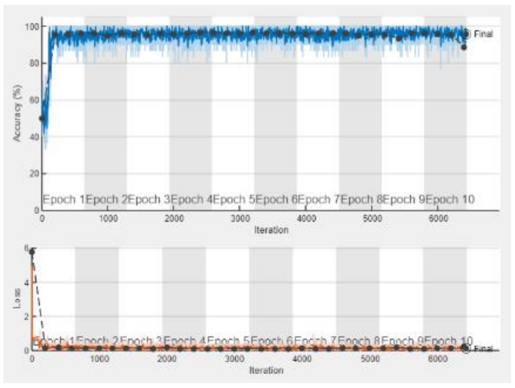


(a)

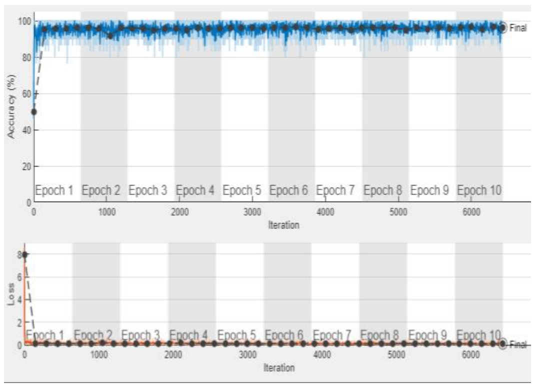


(b)

Figure 7 Re-training of GoogLeNet Network Model with sgdm Optimizer (a)lr=0.001 (b) lr=0.0001

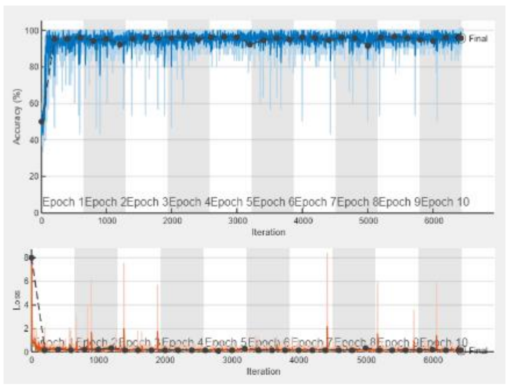


(a)

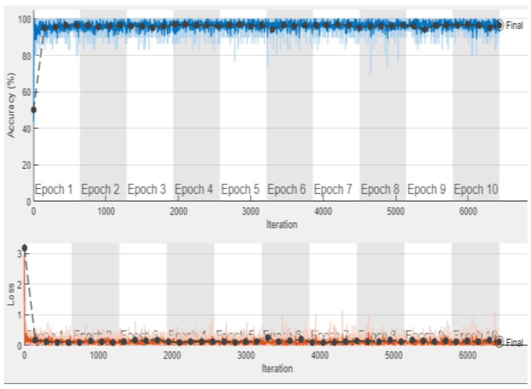


(b)

Figure 8 Re-training of GoogLeNet Network Model with adam Optimizer (a)lr=0.001 (b) lr=0.0001



(a)



(b)

Figure 9 Re-training of GoogLeNet Network Model with rmsprop Optimizer (a)lr=0.001 (b) lr=0.0001

Screenshots of the training window for ResNet-50 are given in Fig. 10, Fig. 11, and Fig.12 respectively. Accuracy and loss rates according to iteration are shown in these graphs. The validation accuracy is obtained 95,57% with sgdm optimizer, 95,66% with adam optimizer, 95,05% with rmsprop optimizer at learning rate of 0.001 and 95.65% with sgdm optimizer, 96.76% with adam optimizer, 96.07% with rmsprop optimizer learning rate of 0.0001.

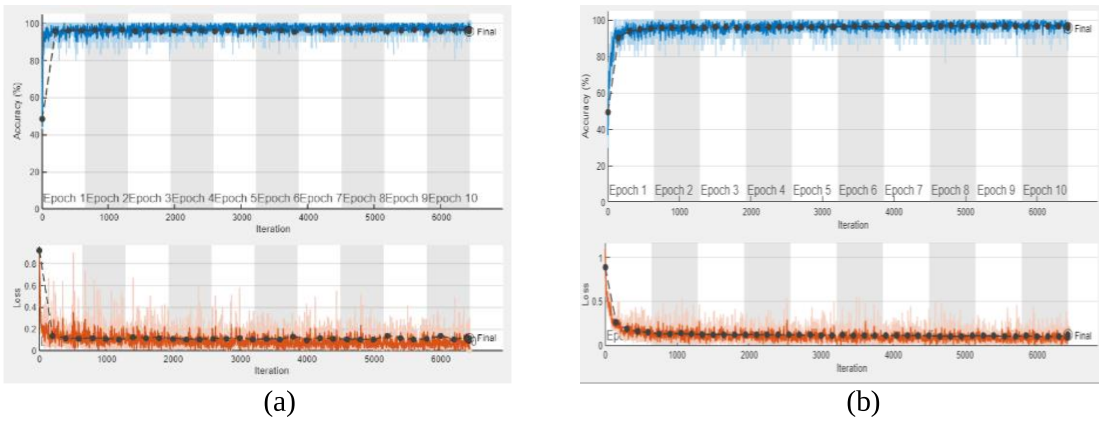


Figure 10 Re-training of ResNet-50 Network Model with sgdm Optimizer (a)lr=0.001 (b) lr=0.0001

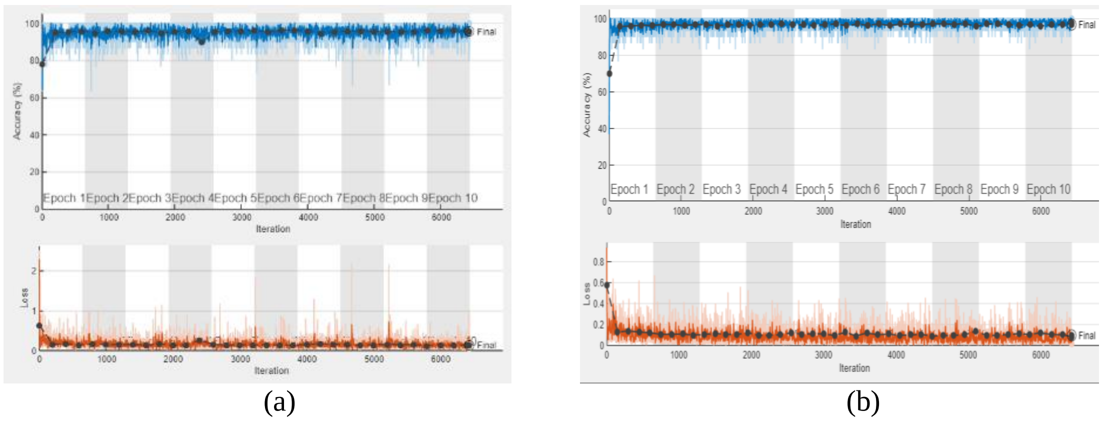


Figure 3 Re-training of ResNet-50 Network Model with adam Optimizer (a)lr=0.001 (b) lr=0.0001

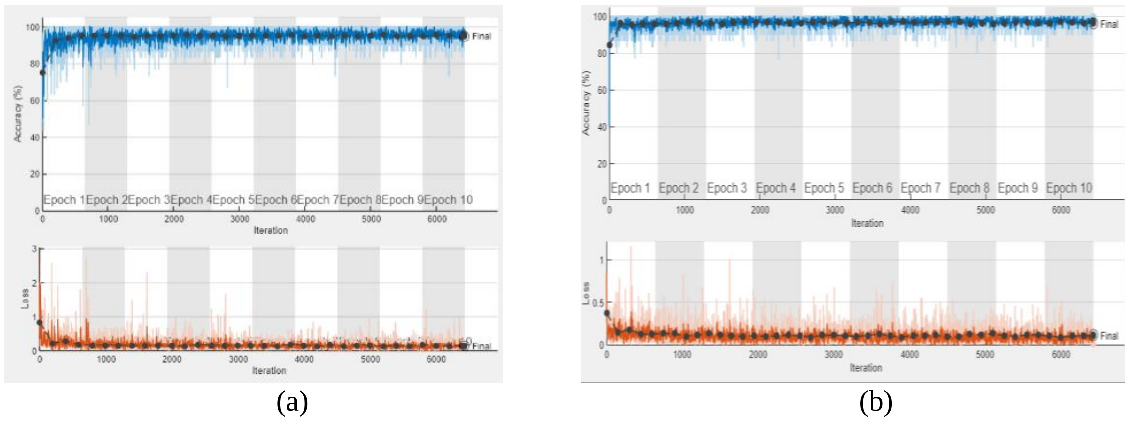
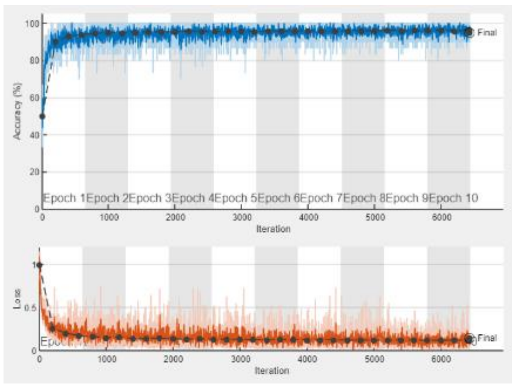
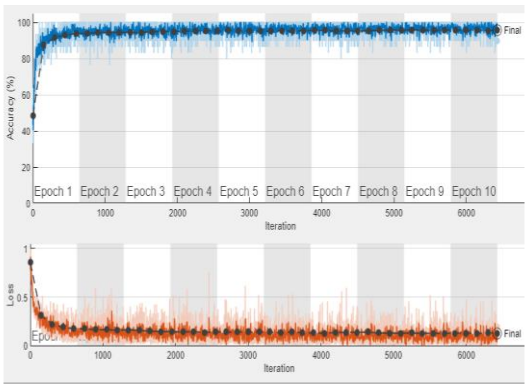


Figure 4 Re-training of ResNet-50 Network Model with rmsprop Optimizer (a)lr=0.001 (b) lr=0.0001

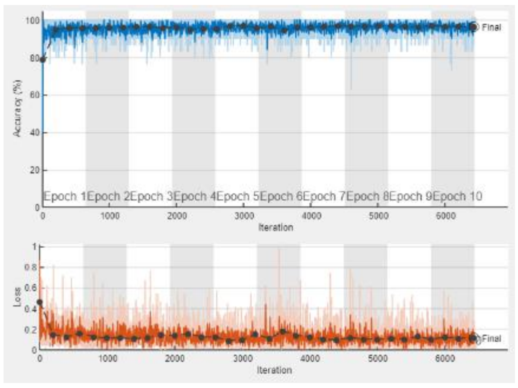
Screenshots of the training window for MobileNet-v2 are given in Fig. 13, Fig. 14, and Fig.15 respectively. Accuracy and loss rates according to iteration are shown in these graphs. The validation accuracy is obtained 95,09% with sgdm optimizer, 96,53% with adam optimizer, 96,31% with rmsprop optimizer at learning rate of 0.001 and 95.72% with sgdm optimizer, 95.63% with adam optimizer, 96.13% with rmsprop optimizer learning rate of 0.0001.

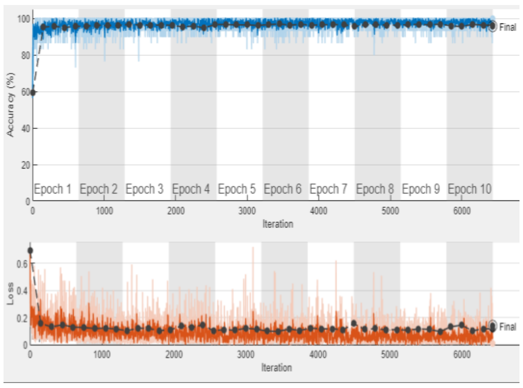




(a) (b)

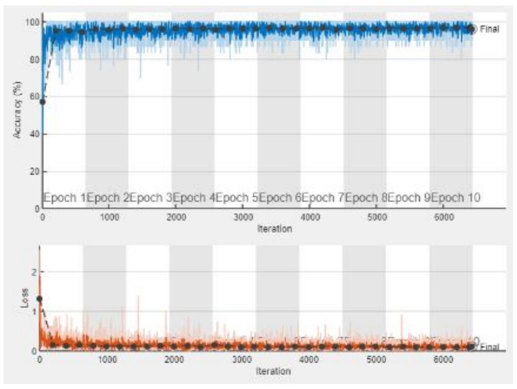
Figure 5 Re-training of MobileNet-v2 Network Model with sgdm Optimizer (a)lr=0.001 (b) lr=0.0001

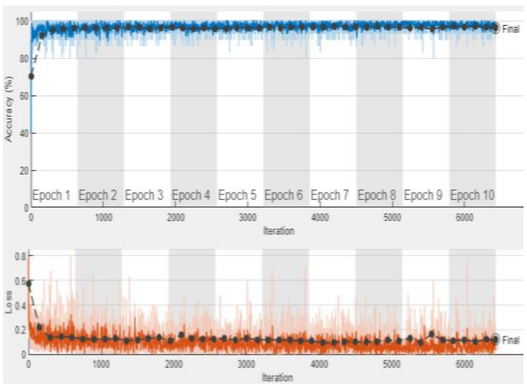




(a) (b)

Figure 6 Re-training of MobileNet-v2 Network Model with adam Optimizer (a)lr=0.001 (b) lr=0.0001

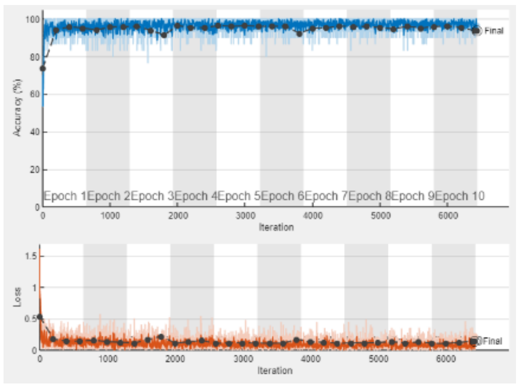


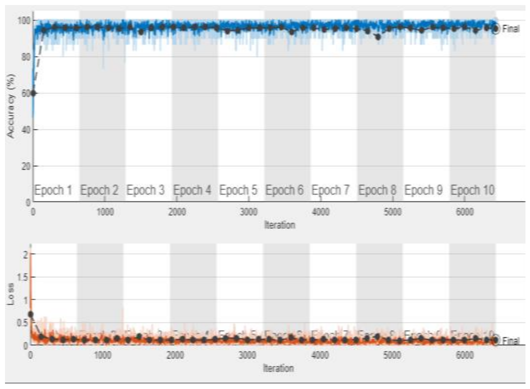


(a) (b)

Figure 7 Re-training of MobileNet-v2 Network Model with rmsprop Optimizer (a)lr=0.001 (b) lr=0.0001

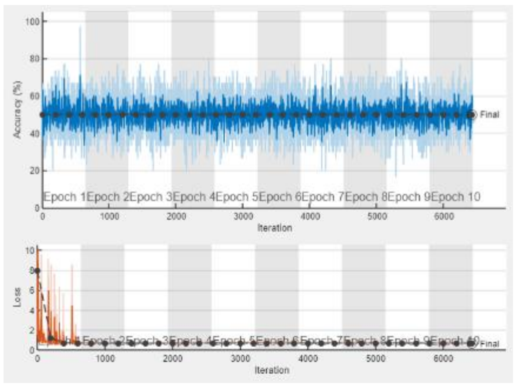
A large number of parameters also affects the retraining speed. Among the architectures used in this study, the longest training period belongs to this architecture. Screenshots of training window for VGG-16 are given in Fig. 16, Fig. 17, and Fig.18 respectively. The validation accuracy is obtained 93,89% with sgdm optimizer, 50% with adam optimizer, 50% with rmsprop optimizer at learning rate of 0.001 and 95.42% with sgdm optimizer, 96.46% with adam optimizer, 95.52% with rmsprop optimizer learning rate of 0.0001.

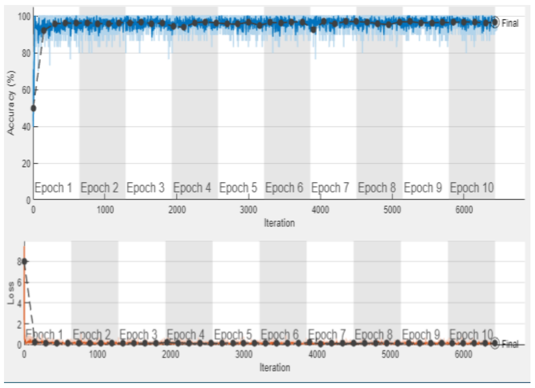




(a) (b)

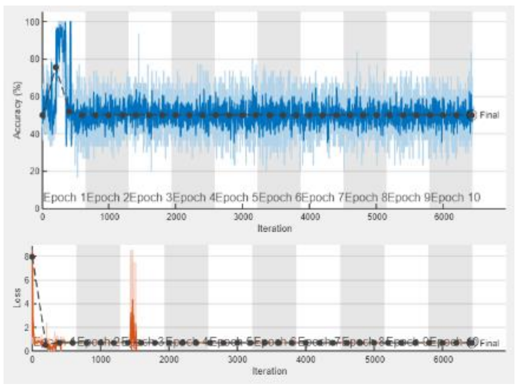
Figure 8 Re-training of the VGG-16 Network Model with sgdm Optimizer (a)lr=0.001 (b) lr=0.0001

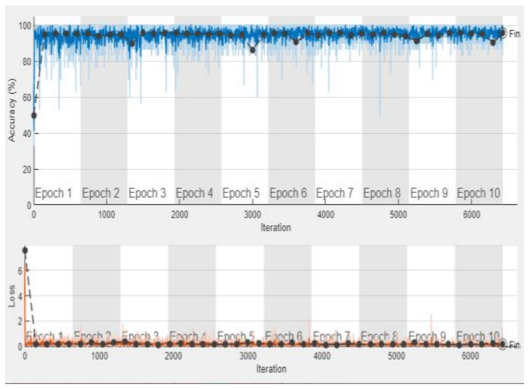




(a) (b)

Figure 9 Re-training of the VGG-16 Network Model with adam Optimizer (a)lr=0.001 (b) lr=0.0001





(a) (b)

Figure 10 Re-Training of The VGG-16 Network Model with Rmsprop Optimizer (a)lr=0.001 (b) lr=0.0001

5. Results

Table 8. represents the entire training results for 0.001 initial learning rate, 30 batch size, and 10 epoch. The most successful results were obtained when the MobileNet-v2 network was trained using the adam optimizer. The network reached a 96,53% validation accuracy rate.

Table 8 Re-training results of network models at 0.001 learning rate

No	Architecture	Learning Algorithm	Learning Rate	Batch Size	Validation Accuracy
1	AlexNet	sgdm	0.001	30	95.9
2	AlexNet	adam	0.001	30	50
3	AlexNet	rmsprop	0.001	30	95.22
4	GoogLeNet	sgdm	0.001	30	95.46
5	GoogLeNet	adam	0.001	30	95.75
6	GoogLeNet	rmsprop	0.001	30	95.54
7	ResNet-50	sgdm	0.001	30	95.57
8	ResNet-50	adam	0.001	30	95.66
9	ResNet-50	rmsprop	0.001	30	95.05
10	MobileNet-v2	sgdm	0.001	30	95.09
11	MobileNet-v2	adam	0.001	30	96.53
12	MobileNet-v2	rmsprop	0.001	30	96.31
13	VGG-16	sgdm	0.001	30	93.89
14	VGG-16	adam	0.001	30	50
15	VGG-16	rmsprop	0.001	30	50

Performance rates from highest to lowest at 0.001 learning rate are given in Figure 19. According to the experimental results, the best results were obtained from the combination of MobileNet-v2 architecture, adam learning algorithm. Goodfits are obtained except three experiments. Combinations VGA16 -sgdm, VGA16-adam, AlexNet-adam failed with this problem.

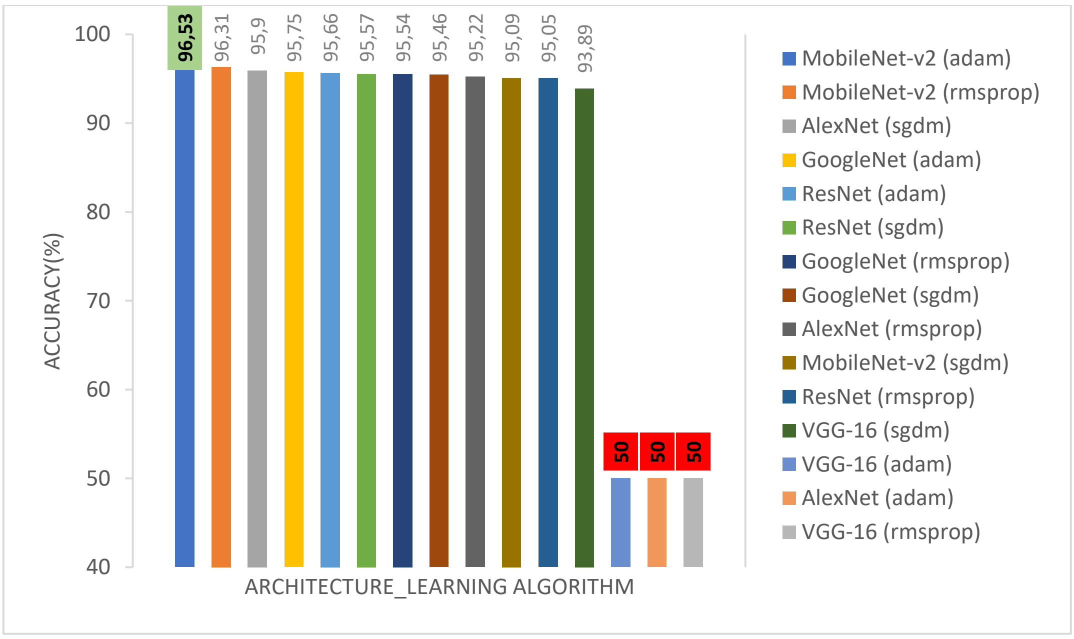


Figure 11 Success Rates of Models at 0.001 learning rate

Table 9. represents the entire training results for 0.0001 initial learning rate, 30 batch size, and 10 epoch. The most successful results were obtained when the ResNet-50 network was trained using the adam optimizer. The network reached a 96,76 % validation accuracy rate. The optimizer type setting is important when using a low learning rate.

Table 9 Re-training results of network models at 0.0001 learning rate

No	Architecture	Learning Algorithm	Learning Rate	Batch Size	Validation Accuracy
1	AlexNet	sgdm	0.0001	30	96.08
2	AlexNet	adam	0.0001	30	94.19
3	AlexNet	rmsprop	0.0001	30	95.85
4	GoogleNet	sgdm	0.0001	30	96.07
5	GoogleNet	adam	0.0001	30	96.26
6	GoogleNet	rmsprop	0.0001	30	96.73
7	ResNet-50	sgdm	0.0001	30	95.65
8	ResNet-50	adam	0.0001	30	96.76
9	ResNet-50	rmsprop	0.0001	30	96.07
10	MobileNet-v2	sgdm	0.0001	30	95.72
11	MobileNet-v2	adam	0.0001	30	95.63
12	MobileNet-v2	rmsprop	0.0001	30	96.13
13	VGG-16	sgdm	0.0001	30	95.42
14	VGG-16	adam	0.0001	30	96.46
15	VGG-16	rmsprop	0.0001	30	95.52

Performance rates from highest to lowest at 0.0001 learning rate are given in Figure 20. According to the experimental results, the best results were obtained from the combination of ResNet architecture, adam learning algorithm.

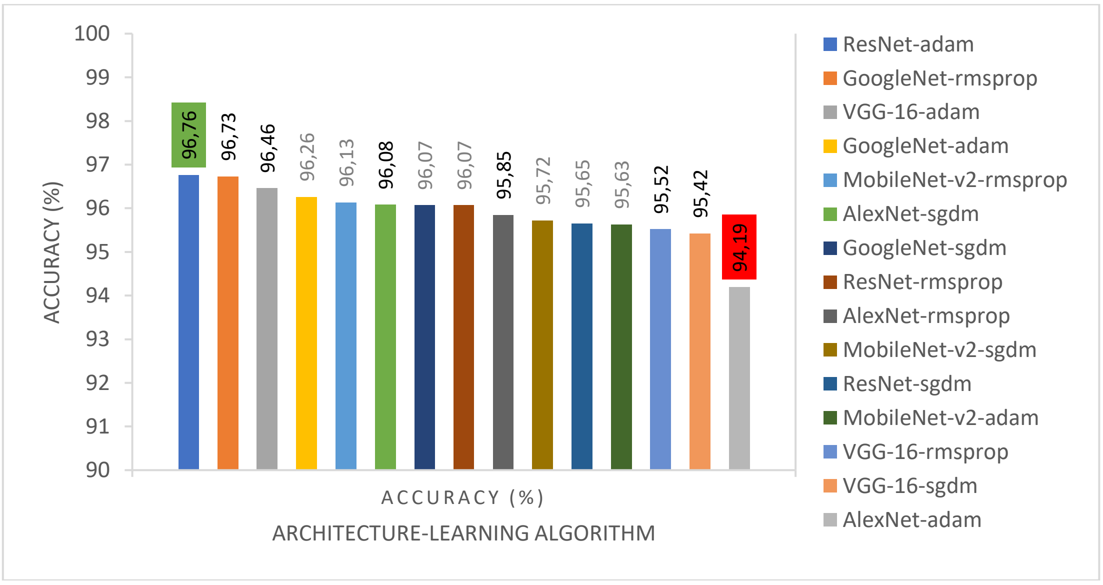


Figure 20 Success Rates of Models at 0.0001 learning rate

In general, the training results were good when the learning rate value was set to 0.0001. Goodfits have been obtained. There is not much difference between the performance rates of the models. For situations where the performance ratios are close to each other, it would be logical to choose the architecture with less number of parameters. In this way, the processing load is less and the result is calculated faster.

6. Conclusions

Malaria is a type of disease that kills when left untreated. Thousands of people die each year due to this disease. When treated, there is full recovery. Malaria can be diagnosed by looking for the malaria parasite in the red blood cell. Deep learning techniques are frequently used in disease detection. Deep learning techniques are very successful in classification problems. Using transfer-based deep learning techniques provides fast and high-performance solutions in image classification. Pre-trained networks are trained using millions of data sets and have proven architectures. In this study, the effect of 3 types of learning algorithms on the performance of 5 types of pre-trained networks at two different learning rate values was investigated. The disease was diagnosed by classifying the red blood cells as having or not having malaria parasites. The duration of the re-trainings, the success rates, and the effects of the learning algorithm on the success was interpreted. When the learning value is set to 0.0001 with the ResNet-50 model and adam optimizer, the maximum success rate of 96.76% has been reached.

References

[1] "Sıtma." [Online]. Available: https://hsgm.saglik.gov.tr/tr/zoonotikvektorel-sitma/detay.html.

[2] WHO, "World malaria report 2020- WHO," 2020. [Online]. Available: https://www.who.int/publications/i/item/9789240015791.

[3] "What is malaria?," *Global Health, Division of Parasitic Diseases and Malaria*, 2021. [Online]. Available: https://www.cdc.gov/.

[4] E. Soylu, T. Soylu, and R. Bayir, "Design and implementation of SOC prediction for a Li-Ion battery pack in an electric car with an embedded system," *Entropy*, vol. 19, no. 4, 2017.

[5] Y. Karabacak and A. Uysal, "Fuzzy logic controlled brushless direct current motor drive design and application for regenerative braking," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, 2017, pp. 1–7.

[6] A. Uysal, S. Gokay, E. Soylu, T. Soylu, and S. Çaşka, "Fuzzy proportional-integral speed control of switched reluctance motor with MATLAB/Simulink and programmable logic controller communication," *Meas. Control (United Kingdom)*, vol. 52, no. 7–8, 2019.

[7] L. V. Selby, W. R. Narain, A. Russo, V. E. Strong, and P. Stetson, "Autonomous detection, grading, and reporting of postoperative complications using natural language processing," *Surg. (United States)*, vol. 164, no. 6, pp. 1300–1305, 2018.

[8] A. Shustanov and P. Yakimov, "CNN Design for Real-Time Traffic Sign Recognition," *Procedia Eng.*, vol. 201, pp. 718–725, 2017.

[9] Y. LeCun *et al.*, "Comparison of learning algorithms for handwritten digit recognition," in *International conference on artificial neural networks*, 1995, vol. 60, pp. 53–60.

[10] Philipp Seeböck, "Deep Learning in Medical Image Analysis," *Vienna University of Technology Faculty of Informatics, Master Thesis* , 2015.

[11] U. Kaya, A. Yılmaz, and Y. Dikmen, "Sağlık Alanında Kullanılan Derin Öğrenme Yöntemleri," *Eur. J. Sci. Technol.*, no. 16, pp. 792–808, 2019.

[12] V. B. Kumar, S. S. Kumar, and V. Saboo, "Dermatological Disease Detection Using Image Processing and Machine Learning," *2016 3rd Int. Conf. Artif. Intell. Pattern Recognition, AIPR 2016*, pp. 88–93, 2016.

[13] S. Jain, V. Jagtap, and N. Pise, "Computer aided melanoma skin cancer detection using image processing," in *Procedia Computer Science, International Conference on Intelligent Computing, Communication & Convergence (ICCC-2015)*, 2015, vol. 48, no. C, pp. 735–740.

[14] A. Chaudhary and S. S. Singh, "Lung cancer detection on CT images by using image processing," *Proc. Turing 100 - Int. Conf. Comput. Sci. ICCS 2012*, pp. 142–146, 2012.

[15] P. Kumar Mallick, S. H. Ryu, S. K. Satapathy, S. Mishra, G. N. Nguyen, and P. Tiwari, "Brain MRI Image Classification for Cancer Detection Using Deep Wavelet Autoencoder-Based Deep Neural Network," *IEEE Access*, vol. 7, pp. 46278–46287, 2019.

[16] M. J. Horry *et al.*, "COVID-19 Detection through Transfer Learning Using Multimodal Imaging Data," *IEEE Access*, vol. 8, pp. 149808–149824, 2020.

[17] M. Toğaçar, B. Ergen, and Z. Cömert, "Tumor type detection in brain MR images of the deep model developed using hypercolumn technique, attention modules, and residual blocks," *Med. Biol. Eng. Comput.*, vol. 59, no. 1, pp. 57–70, 2021.

[18] A. A. Abbasi *et al.*, "Detecting prostate cancer using deep learning convolution neural network with transfer learning approach," *Cogn. Neurodyn.*, vol. 14, no. 4, pp. 523–533, 2020.

[19] T. Rahman *et al.*, "Transfer learning with deep Convolutional Neural Network (CNN) for pneumonia detection using chest X-ray," *Appl. Sci.*, vol. 10, no. 9, 2020.

[20] Vijayalakshmi A and Rajesh Kanna B, "Deep learning approach to detect malaria from microscopic images," *Multimed. Tools Appl.*, vol. 79, no. 21–22, pp. 15297–15317, 2020.

[21] Y. Dong *et al.*, "Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells," *2017 IEEE EMBS Int. Conf. Biomed. Heal. Informatics, BHI 2017*, pp. 101–104, 2017.

[22] F. Yang *et al.*, "Deep Learning for Smartphone-Based Malaria Parasite Detection in Thick Blood Smears," *IEEE J. Biomed. Heal. Informatics*, vol. 24, no. 5, pp. 1427–1438, 2020.

[23] W. D. Pan, Y. Dong, and D. Wu, "Classification of Malaria-Infected Cells Using Deep Convolutional Neural Networks," in *Machine Learning - Advanced Techniques and Emerging Applications*, 2018, pp. 159–173.

[24] A. Sai Bharadwaj Reddy and D. Sujitha Juliet, "Transfer learning with RESNET-50 for malaria cell-image classification," *Proc. 2019 IEEE Int. Conf. Commun. Signal Process. ICCSP 2019*, pp. 945–949, 2019.

[25] K. M. F. Fuhad, J. F. Tuba, M. R. A. Sarker, S. Momen, N. Mohammed, and T. Rahman, "Deep learning based automatic malaria parasite detection from blood smear and its smartphone based application," *Diagnostics*, vol. 10, no. 5, 2020.

[26] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018-Janua, pp. 1–6, 2018.

[27] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," *IH MMSec 2016 - Proc. 2016 ACM Inf. Hiding Multimed. Secur. Work.*, pp. 5–10, 2016.

[28] D. Miao, W. Pedrycz, D. Ślezak, G. Peters, Q. Hu, and R. Wang, "Mixed Pooling for Convolutional Neural Networks," in *International Conference on Rough Sets and Knowledge Technology*, 2014, vol. 8818, pp. 364–375.

[29] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, "Learning Pooling for Convolutional Neural Network," *Neurocomputing*, vol. 224, no. April 2016, pp. 96–104, 2017.

[30] S. Postalcıoğlu, "Performance Analysis of Different Optimizers for Deep Learning-Based Image Recognition," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 34, no. 2, 2020.

[31] H. Chen *et al.*, "Deep Transfer Learning for Person Re-Identification," *2018 IEEE 4th Int. Conf. Multimed. Big Data, BigMM 2018*, 2018.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "2012 AlexNet," *Adv. Neural Inf. Process. Syst.*, 2012. [Online]. Available: https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[34] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[36] S. Rajaraman *et al.*, "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, vol. 6, p. e4568, 2018.

[37] "Malaria Cell Images Dataset." [Online]. Available: https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria.