Web-Based Application for Malaria Parasite Detection Using Thin-Blood Smear Images

Windra Swastika [1,*], Benedictus J. Pradana [1], Romy B. Widodo [1], Rehmadanta Sitepu [2], and Gregorius G. Putra [1]

[1] Informatics Engineering, Universitas Ma Chung, Malang, Indonesia
[2] Pharmaceutical Science and Biomedicine, Universitas Ma Chung, Malang, Indonesia;
Email: rehmadanta.sitepu@machung.ac.id (R.S.)
*Correspondence: windra.swastika@machung.ac.id. (W.S.)

***Abstract*—Malaria is an infectious disease caused by the Plasmodium parasite. In 2019, there were 229 million cases of malaria with a death toll of 400.900. Malaria cases increased in 2020 to 241 million people with the death toll reaching 627,000. Malaria diagnosis which is carried out by observing the patient's blood sample requires experts and if it is not done correctly, misdiagnosis can occur. Deep Learning can be used to help diagnose Malaria by classifying thin blood smear images. In this study, transfer learning techniques were used on the Convolutional Neural Network to speed up the model training process and get high accuracy. The architecture used for Transfer Learning is EfficientNetB0. The training model is embedded in a python-based web application which is then deployed on the Google App Engine platform. This is done so that it can be used by experts to help diagnose. The training model has a training accuracy of 0.9664, a training loss of 0.0937, a validation accuracy of 0.9734, and a validation loss of 0.0816. Prediction results on test data have an accuracy of 96.8% and an F1-score value of 0.968.**

***Keywords*—malaria detection, Convolution Neural Network (CNN), transfer learning**

I. Introduction

Based on the WHO report, in 2019 there were 229 million malaria cases worldwide and 400.900 deaths from malaria. The group most vulnerable to contracting malaria are children under the age of five with a percentage of 67% of total deaths (274,000). The distribution of malaria in 2019 was still centered in Africa with a case percentage of 94% of total cases [1]. Indonesia ranks second for the highest number of malaria cases in Southeast Asia. Malaria cases in Indonesia had decreased in 2010–2014 and became stagnant in 2014–2019 [1]. Based on data from the Ministry of Health in 2019 there were 250.644 malaria cases, the majority of cases were in Papua with a percentage of 86% (216,380) followed by East Nusa Tenggara Province with 12,909 cases. Provinces that are free of malaria or have 0 cases include DKI Jakarta Province, East Java Province, and Bali Province [2].

During the last two decades, WHO has made recommendations on how to prevent malaria, namely vector control, preventive chemotherapy, and malaria vaccination. Vector control is an effort to prevent malaria using Insecticide-Treated Nets (ITNs) or a type of net made specifically to physically repel mosquitoes, preventive chemotherapies are prevention efforts using drugs or vaccination. Since early October 2021 WHO has recommended the widespread use of the RTS, S/AS01 vaccine to prevent malaria, especially in children [3].

Malaria is diagnosed by physical examination and blood tests. Blood tests to diagnose malaria include malaria Rapid Diagnostic Tests (RDT malaria) and examination of the patient's blood under a microscope. This aims to detect malaria-causing parasites found in blood cells [4]. This diagnosis is done manually by the doctor and if the experience of the doctor who diagnoses is not qualified, it can result in a misdiagnosis.

Machine Learning is part of artificial intelligence where computers can solve problems or find patterns from data without having to be programmed explicitly [5]. One method of machine learning is transfer learning; this method is applied by using a machine learning model that already made previously as a starting point for a similar problem. Machine learning can be used to classify a set of images to detect the image class.

There are two objectives of this research. First, utilizing transfer learning by testing various parameters to get a model with optimal accuracy. Second, developing a web-based application using the model that has been obtained.

II. Literature Review

A. Convolution Neural Network

Convolutional Neural Network (CNN) has been widely used in computer vision tasks such as image classification, image analysis, and image recognition. The main goal in computer vision is to make machines work like humans in understanding an image or recognizing a pattern. Development of a computer vision algorithm that is part of deep learning called CNN. CNN works similar as the

Manuscript received April 24, 2023; revised May 25, 2023; accepted June 30, 2023.

visual cortex, which connects every neuron in the brain to recognize an image or pattern. CNN is designed to study spatial hierarchical data that has an adaptive and automatic grid pattern [6].

The CNN architecture consists of three layers, namely the input layer, hidden layer, and output layer. The CNN input is in the form of a matrix representing pixel values in the range 0–255 in a digital image. The hidden layer consists of convolution layer and a pooling layer, which is followed by at least one fully connected layer. The convolution layer performs feature extraction using an input layer with a 3×3 or 5×5 filter by calculating the values for each kernel and adding them to get a feature map consisting of dense one-depth channels. Padding is used on convolutional layers to keep the feature map size from shrinking by adding 0 pixels to each side of the input entered. The resulting output will be converted to non-linear using an activation function which is a mathematical representation of biological neuron behavior, one of the functions that is often used is the Rectified Linear Unit (RELU) by calculating the function (x)=max (0.x). Pooling layer serves to reduce the spatial size of the feature map that has gone through the convolutional layer, thereby reducing computation. There are two types of pooling, namely Max Pooling and Average Pooling. Max Pooling selects the highest value from the kernel while Average Pooling averages all values in the kernel. The flatten layer serves to reduce the dimensions of the previously three-dimensional input vector into one dimension which is then channeled into a fully connected layer for classification. In the fully connected layer, the activation functions used are softmax, softmax will ensure that the total number of CNN output values is one. Softmax serves to scale the output values into probabilities. Fig. 1 shows the architecture of CNN starting with the input layer, followed by the convolution layer, pooling layer, fully connected, and output layer.

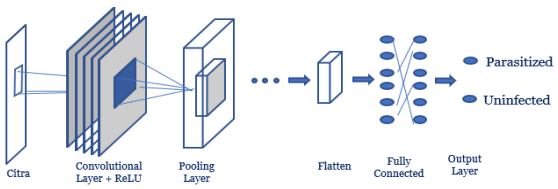

Figure 1. General CNN architecture consists of input image, convolution layer, pooling layer and output layer.

CNN has performed in various fields in computer vision. Radiological tasks such as x-ray images can be solved with CNN such as Yadav's study [7] conducted image classification to detect pneumonia using the VGG16, InceptionV3, and Capsule Neural Network architectures resulting in accuracy of 0.923, 0.869, and 0.824. The architecture of VGG166 is not too complex so that it is overfitting compared to InceptionV3 which is experiencing overfitting. Object detection using CNN for mask detection results in up to 99% accuracy on the MobileNetV2 Architecture with Adam Optimizer [8]. Combining Long Short Term Memory (LSTM) with Architectural Inception CNN is able to perform American Sign Language Recognition with performance accuracy approaching 90% [9]. A similar study using CNN-LSTM to detect natural frequencies of different beams in modal frequency detection tax was able to produce the highest accuracy of 96.6% on aluminum-long [10].

B. EfficientNet Architecture

EfficientNet architecture is a CNN architecture developed by Google's Brain Team to increase efficiency in scaling and improve performance accuracy [11]. Scaling is done on the channel width of the layer, the height of the number of layers, and the resolution of the digital image. EfficientNet automates scaling up using AutoML from the MobileNet model architecture. EfficientNet has 8 different models with different number of parameters named sequentially from B0 to B7 [12]. EfficientNet architecture requires faster computation time by applying the concept of inverted residual block [13]. The EfficientNetB0 architecture is shown in Fig. 2.

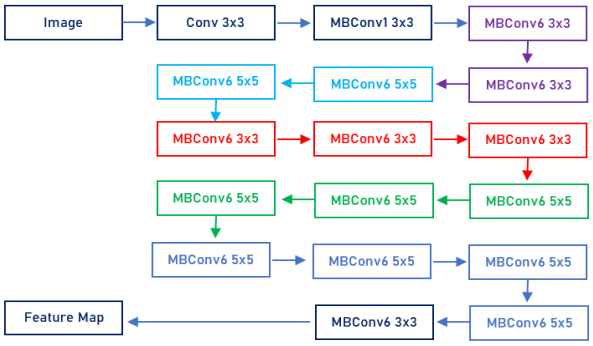

Figure 2. EfficientNetB0 architecture.

EfficientNetB0 architecture is a CNN architecture designed to make it easier to scale and generalize. EfficientNetB0 has been trained using 14 million digital images with 1000 different classes from the ImageNet database. From Fig. 2, the EfficientNetB0 architecture requires an image input with a size of 224×244. Then the concept of inverted residuals structure is applied to the MBConv block derived from the MobileNetV3 architecture. In this concept, it uses a narrow to wide approach and then to a narrow approach which is inverted in the architecture. MBConv performs multi-objective optimization with the objective function of Pareto optimal solutions that seek the highest accuracy and speed using a hierarchical search [14]. The last MBConv layer has a 1×1 filter to add output to the classification process.

EfficientNet Architecture conducted a study to detect diabetic retinopathy capable of detecting five levels of disease with an accuracy rate of 79.8% [15]. Detecting medical images on X-Ray images to detect Covid-19 EfficientNetB0 is able to detect very well with an average accuracy of 92.93% [16]. EfficientNetB0 can also be used to detect diseases in plants, study [17] also uses the InceptionV3, InceptionResNetV2, MobileNetV2 architecture to detect 14 different plant species with 38 categories of disease classes and healthy leaves. The results of his study stated that EfficientNetB0 has the best

accuracy of 99.56% and MobileNetV2 is the lowest architecture with an accuracy of 97.02%.

C. Transfer Learning

Deep Learning architectural models that have been designed will be trained to solve specific problems. When solving similar problems with different datasets, the trained model cannot be used directly due to low capability. Transfer Learning is a machine learning technique that is used to improve accuracy by utilizing a previously trained model as a starting point for different but still relevant problems [18]. By using transfer learning, problem solving can be done with a small number of samples [19]. The model that has been previously trained and then reused as a starting point to create a new model is called a pre-trained model [20]. The pretrained model uses a trained architecture such as InceptionV2, EfficientNet, MobileNet, and VGG16. In Transfer Learning model can be trained in two ways. The first way only trains the output layer and preserves the pretrained model weights (freeze the pretrained model layer). The second way is to train all the layer to increase the model performance, this process is called fine tuning.

Pretrained model as feature extraction is done by freezing the convolutional layer. The early layer in the convolutional layer is used to identify the image in a simple form, and the later layer is used to detect complex images that produce vector values. This method does not require model retraining because the model freezes in the early and later layers, so that the resulting output is still a vector. This vector value will be classified using flatten, fully connected, and softmax. This vector value can be classified using deep learning and machine learning [21]. The number of classes in the output does not have to be the same as the number of classes in the pre-trained model. Pretrained models such as MobileNetV2, EfficientNet, and InceptionV3 models.

Fine Tuning is a method that adjusts the architecture so that it becomes optimal in solving problems. Fine Tuning performs retraining of the model by adding several methods and parameters. The frozen layer is the layer adjacent to the input layer while the other layers will be tuned. The fine-tuning process is carried out carefully so as not to drop the performance of the pretrained model. There are many methods for fine tuning, one of them is by adding Batch Normalization or pooling layer to the classification process. Fine-tuning method can be done adaptively using AdaFilter which is better than standard fine-tuning [22].

Some studies use transfer learning with pretrained model efficientnetb0. A study conducted by Pangkasidhi [23] carried out transfer learning using the EfficientNet model versions B0. B1, and B2 resulting in an accuracy of 92.54%, 93.19%, and 92.03%. In Ref. [24], the EfficientNetB0 architecture performed transfer learning and hyperparameter tuning on Dense, Batch Size, and Learning Rate resulting in the highest accuracy of 95.17% on the ADAM optimizer. Transfer Learning using EfficientNetB0 produces 98.33% accuracy and MobileNetV3 provides 98.75% accuracy in detecting puppets [25].

III. Materials and Methods

A. Data Collection

The dataset obtained comes from the Lister Hill National Center for Biomedical Communication and was first introduced by Rajaraman *et al.* [26]. The dataset is a collection of Blood Smear images which have two classes, namely uninfected and parasitized. The total number of datasets is 27,558, which is divided into two classes with the same number of different image sizes. Different image sizes will be equated according to the needs of the required CNN architecture. The image in the parasitized class has a pale bluish purple with black spots from infected blood, while in the uninfected class it does not have a purple colour. The dataset is divided into three parts, i.e., data train, validation, and test data. Ratio between data train, validation, and data test is 70%: 15%: 15%, respectively. The sample images of two classes can be seen in Fig. 3.

B. Model Training

In this study, 18 experiments were carried out to get the best results. Before the training process starts, the image will be resized to 128×128 and image augmentation will be applied. Image augmentation that will be used are random flip, random height, random width, random zoom, and random rotation each with value 0.2. Since there are five augmentations, each image will be augmented five times during training process. Table I shows the hyper-parameter used during the training process.

TABLE I. Experiments and Hyper-Parameters for Training

No.	Experiments	Optimizer	Learning Rate	Epochs	Fine Tune
1	EfficientNetB0_SGD1	SGD	0.001	20	No
2	EfficientNetB0_SGD2	SGD	0.00001	10	Yes
3	EfficientNetB0_SGD3	SGD	0.0001	20	No
4	EfficientNetB0_SGD4	SGD	0.00001	10	Yes
5	EfficientNetB0_SGD5	SGD	0.0001	30	No
6	EfficientNetB0_SGD6	SGD	0.00001	15	Yes
7	EfficientNetB0_Adam1	Adam	0.001	20	No
8	EfficientNetB0_Adam2	Adam	0.00001	10	Yes
9	EfficientNetB0_Adam3	Adam	0.0001	20	No
10	EfficientNetB0_Adam4	Adam	0.00001	10	Yes
11	EfficientNetB0_Adam5	Adam	0.0001	30	No
12	EfficientNetB0_Adam6	Adam	0.00001	15	Yes
13	EfficientNetB0_RMSprop1	RMSprop	0.001	20	No
14	EfficientNetB0_RMSprop2	RMSprop	0.00001	10	Yes
15	EfficientNetB0_RMSprop3	RMSprop	0.0001	20	No
16	EfficientNetB0_RMSprop4	RMSprop	0.00001	10	Yes
17	EfficientNetB0_RMSprop5	RMSprop	0.0001	30	No
18	EfficientNetB0_RMSprop6	RMSprop	0.00001	15	Yes

As shown in Table I, there are 18 experiments using three different optimizers, namely SGD, Adam and RMSProp. Each optimizer will be tested with two different learning rates (0.001 and 0.0001) and four different epochs (10, 15, 20, and 30). The last parameter for the experiment is fine-tune. Fine-tune consists of unfreezing the entire model obtained before and re-training on the new data with low learning rate (0.0001). The fine-tune parameter is expected to have better accuracy compared to those without fine-tune. All experiments using the same EfficientNetB0 architecture.

C. Evaluation

Model from each experiment will be evaluated on test dataset using F1-Score and Confusion Matrix (CM). CM is a standard tool to evaluate model performance on test data (data that has never been seen before). CM has corresponding rows and columns representing ground truth class and predicted class. CM consist of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). TP is positive class that predicted as positive, FP is non-positive class that predicted as positive, TN is negative class that predicted as negative, and FN is non-negative class that predicted as negative. F1-Score is weighted average of precision and recall.

IV. Result and Discussion

The result of experiments is split into two parts. First is the result of model training and second is the result of prediction on test dataset. The time required for training the model without fine tune (experiment labeled "b") is 15 minutes, meanwhile model with fine tune require 22.5 minutes. Fine tuning is done to the previous model (experiment labeled "a").

A. Training Results

The results of accuracy and loss using training data and validation data can be seen in Table II.

TABLE II. Training Results

No.	Experiments	Training Loss	Training Accuracy	Val Loss	Val Accuracy
1	EfficientNetB0_SGD1a	0.2333	0.9107	0.1932	0.9284
2	EfficientNetB0_SGD1b	0.2094	0.9213	0.1633	0.9395
3	EfficientNetB0_SGD2a	0.3185	0.8765	0.2714	0.9006
4	EfficientNetB0_SGD2b	0.2557	0.9053	0.2082	0.9272
5	EfficientNetB0_SGD3a	0.3057	0.8804	0.2664	0.9001
6	EfficientNetB0_SGD3b	0.2427	0.9081	0.2009	0.9286
7	EfficientNetB0_Adam1a	0.1912	0.9287	0.1627	0.9448
8	EfficientNetB0_Adam1b	0.1015	0.9638	0.0823	0.9695
9	EfficientNetB0_Adam2a	0.2027	0.9242	0.1629	0.9422
10	EfficientNetB0_Adam2b	0.1004	0.9632	0.086	0.9702
11	EfficientNetB0_Adam3a	0.201	0.9244	0.1571	0.9456
12	EfficientNetB0_Adam3b	0.0899	0.9678	0.0834	0.9698
13	EfficientNetB0_RMSprop1a	0.1917	0.928	0.1471	0.9482
14	EfficientNetB0_RMSprop1b	0.1041	0.963	0.0979	0.9693
15	EfficientNetB0_RMSprop2a	0.2083	0.9226	0.1652	0.9417
16	EfficientNetB0_RMSprop2b	0.1026	0.9633	0.0861	0.9724
17	EfficientNetB0_RMSprop3a	0.2029	0.925	0.1596	0.9432
18	EfficientNetB0_RMSprop3b	0.0937	0.9664	0.0816	0.9734

Based on the experiment in Table II, the overall model does not have a significant difference in term of accuracy. We can say that all models are good at generalizing on the given data. The EfficientNetB0_RMSprop3b experiment is the experiment that produces the highest accuracy performance (0.9734). Fine Tune on EfficientNetB0 results in better model performance than without fine-tuning. The SGD optimizer used in EfficientNetB0 has lower performance than using Adam and RMSprop with the same learning-rate and epoch hyperparameters.

B. Prediction Resutls

Table III shows the results of the model predictions on the test dataset, and Fig. 3 shows the F1-Score comparison and the prediction accuracy comparison.

TABLE III. Prediction Results

No.	Experiments	Training Loss	Training Accuracy	Val Loss	Val Accuracy
1	EfficientNetB0_SGD1a	0.925	0.911	0.918	0.918
2	EfficientNetB0_SGD1b	0.931	0.93	0.9305	0.9305
3	EfficientNetB0_SGD2a	0.887	0.882	0.8845	0.8845
4	EfficientNetB0_SGD2b	0.915	0.913	0.914	0.914
5	EfficientNetB0_SGD3a	0.905	0.866	0.8855	0.8855
6	EfficientNetB0_SGD3b	0.929	0.908	0.9185	0.9185
7	EfficientNetB0_Adam1a	0.888	0.967	0.9275	0.9275
8	EfficientNetB0_Adam1b	0.956	0.973	0.9645	0.9645
9	EfficientNetB0_Adam2a	0.918	0.948	0.933	0.933
10	EfficientNetB0_Adam2b	0.961	0.972	0.9665	0.9665
11	EfficientNetB0_Adam3a	0.913	0.957	0.935	0.935
12	EfficientNetB0_Adam3b	0.944	0.99	0.967	0.967
13	EfficientNetB0_RMSprop1a	0.919	0.958	0.9385	0.9385
14	EfficientNetB0_RMSprop1b	0.966	0.956	0.961	0.961
15	EfficientNetB0_RMSprop2a	0.92	0.939	0.9295	0.9295
16	EfficientNetB0_RMSprop2b	0.961	0.973	0.967	0.967
17	EfficientNetB0_RMSprop3a	0.92	0.948	0.934	0.934
18	EfficientNetB0_RMSprop3b	0.965	0.971	0.968	0.968

In Figs. 3 and 4 there are six models with highest performance based on F1-Score and Prediction accuracy, those are EfficientNetAdam1b, EfficientNetAdam2b, EfficientNetAdam3b, EfficientNetRMSprop1b, EfficientNetRMSprop2b, and EfficientNetRMSprop3b. Table IV shown the confusion matrix of those models.

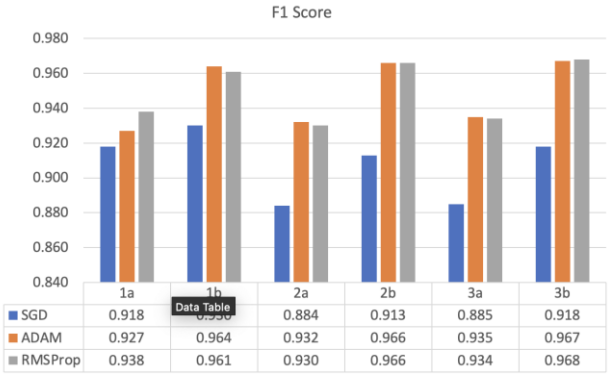

Figure 3. F1-Score graph.

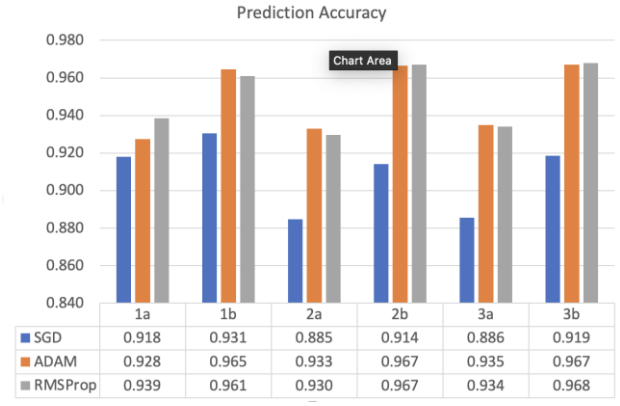

Figure 4. Prediction accuracy graph.

TABLE IV. CONFUSION MATRIX RESULTS

Experiments	Ground Truth	Predicted Class	
		Parasitized	Uninfected
EfficientNetB0_Adam1b	Parasitized	1976 (95.6%)	91 (4.4)%
	Uninfected	56 (2.7%)	2011(97.3%)
EfficientNetB0_Adam2b	Parasitized	1987 (96.1%)	80 (3.9%)
	Uninfected	58 (2.8%)	2009 (97.2%)
EfficientNetB0_Adam3b	Parasitized	1952 (94.4%)	115 (5.6%)
	Uninfected	21 (1.0 %)	2046 (99.0%)
EfficientNetB0_RMSprop1b	Parasitized	1997 (96.6%)	70 (3.4%)
	Uninfected	91 (4.4%)	1976 (95.6%)
EfficientNetB0_RMSprop2b	Parasitized	1986 (96.1%)	81 (3.9%)
	Uninfected	56 (2.7%)	2011 (97.3%)
EfficientNetB0_RMSprop3b	Parasitized	1995 (96.5%)	72 (3.5%)
	Uninfected	60 (2.9%)	2007 (97.1%)

Based on the confusion matrix in Table IV, the six best models have no significant difference when predicting the Parasitized and Uninfected classes. The best model for predicting the Parasitized class is the EfficientNetB0_RMSprop3b model, while the uninfected class prediction is the best model for EfficientNetB0_Adam3b. Of the two models that have the best ability to make predictions, EfficientNetB0_RMSprop3b achieve 0.968 accuracy. Hence, the best model EfficientNetB0_RMSprop3b will be implemented into a web application.

C. Deployment

The model chosen for deployment is EfficientNetB0_RMSprop3b and the platform for deployment is Google App Engine (GAE). The model is embedded on web app built with Streamlit library. Before being deployed, the web app is dockerized first and then deploy the web app on GAE. The model needs to be deployed so it can be used as a malaria screener. The web app can be used by uploading cell image and then click predict button to show the prediction result of the image. The interface of the web app can be seen on Figure 5 below.

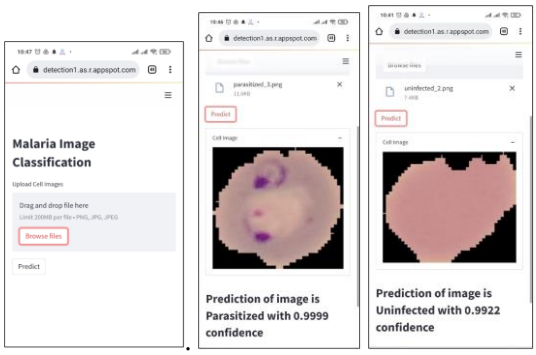

Figure 5. Interface of web-based application malaria image classification.

V. CONCLUSION

Model that has best performance is EfficientNetB0_RMSprop3b, with training result in *training loss*=0.0937, *validation loss*=0.0816, *training accuracy*=0.9664, *validation accuracy*=0.9734. Prediction result on test dataset is accuracy=96.8% and F1-score=0.968. The required time to train the model before *fine tuning* (experiment labeled "a") is 15 minutes and during *fine tuning* (experiment labeled "b") is 22.25 minutes. The increase of model accuracy by fine tuned the model is almost 5%.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Windra Swastika: Designed and developed the machine learning algorithm for malaria parasite detection using thin-blood smear images; Benedictus J. Pradana: Collected and prepared the thin-blood smear images used in the study; Romy B. Widodo: Designed and developed the web-based application for malaria parasite detection using the machine learning algorithm; Rehmadanta Sitepu: Conducted user testing of the web-based application with medical professionals and provided feedback on the usability and functionality of the application.; Gregorius G.

Putra: Conducted a literature review on existing methods for malaria parasite detection using thin-blood smear images. All authors had approved the final version

FUNDING

This work was supported by the Ministry of Science and Ministry of Education, Culture, Research, and Technology (Grant no. 023/SP2H/PT-L/LL7/2022).

REFERENCES

[1] World Health Organization, "World malaria report," 2020.
[2] Kementrian Republik Indonesia, "Profil Kesehatan Indonesia," 2019.
[3] World Health Organization. (2022). Malaria Fact Sheets. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/malaria
[4] Alodokter. Malaria-Gejala, penyebab, dan mengobati—Alodokter. [Online]. Available: https://www.alodokter.com/malaria
[5] IBM. (15 July, 2020). What is machine learning? [Online]. Available: https://www.ibm.com/cloud/learn/machine-learning
[6] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights Imaging*, vol. 9, pp. 611–629, 2018.
[7] S. S. Yadav and S. M. Jadhav, "Deep convolutional neural network based medical image classification for disease diagnosis," *Journal of Big Data*, vol. 6, no. 113, 2019.
[8] N. S. Rashmi and N. Manohar, "Computer-vision based face mask detection using CNN," in *Proc. 2021 6th International Conference on Communication and Electronics Systems (ICCES)*, Coimbatre, India, 2021. doi: 10.1109/ICCES51350.2021.9489098
[9] K. Bantupalli and Y. Xie, "American sign language recognition using deep learning and computer vision," in *Proc. 2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2018. doi: 10.1109/BigData.2018.8622141
[10] R. Yang, S. K. Singh, M. Tavakkoli, N. Amiri, Y. Yang, M. A. Karimi, and R. Rai, "CNN-LSTM deep learning architecture for computer vision-based modal frequency detection," *Mechanical System and Signal Processing*, vol. 144, 106885, 2020.
[11] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. International Conference on Machine Learning*, 2019, pp. 6105–6114.
[12] R. H. Hridoy, F. Akter, and A. Rakshit, "Computer vision based skin disorder recognition using EfficientNet: A transfer learning approach," in *Proc. 2021 International Conference on Information Technology (ICIT)*, 2021.
[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018.
[14] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.
[15] S. Rizal, N. Ibrahim, N. K. C. Pratiwi, S. Saidah, and R. Y. N. Fuadah, "Deep learning for classification of diabetic retinopathy uses the efficientnet model," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, dan Teknik Elektronika*, vol. 8, no. 3, pp. 693–705, 2020. (in Indonesian)
[16] L. Gaur, U. Bhatia, N. Z. Jhanjhi, G. Muhammad, and M. Masud, "Medical image based detection of Covid-19 using deep learning convolutional neural networks," *Multimed Syst.*, vol. 29, no. 3, pp. 1729–1738, 2021, doi: 10.1007/s00530-021-00794-6
[17] S. M. Hassan, A. K. Maji, M. Jasinski, Z. Leonowicz, and E. Jasinska, "Identification of plant-leaf diseases using CNN and transfer-learning approach," *Electronics*, vol. 10, no. 1388, 2021.
[18] J. Brownlee. (2017). A gentle introduction to transfer learning for deep learning. [Online]. Available: https://machinelearningmastery.com/transfer-learning-for-deep-learning/
[19] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *IEEE*, 2021, vol. 109, no. 1, pp. 43–76.
[20] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo and J. Qiu, "Pre-trained models: Past, present and future," arXiv:2106.07139 [cs.AI], 2021.
[21] P. Marchelino. (2018). Transfer learning form pre-trained models. [Online]. 8. Available: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751
[22] Y. Guo, Y. Li, L. Wang, and T. Rosing, "AdaFilter: Adaptive fitler fine-tuning for deep transfer learning," in *Proc. AAAI Conference on Artificial Intelligence*, 2020.
[23] A. K. Pangkasidhi, H. N. Palit, and A. N. Tjondrowiguno, "An application that supports the diagnosis of Covid-19 that analyzes X-ray results of the lungs using the efficientnet model," *Jurnal INFRA*, vol. 9, no. 2, 2021. (in Indonesian)
[24] G. Y. Alhafis, Jasril, S. Suwanto, F. Syafria, and E. Budianita, "Image classification of beef and pork using feature extraction and convolutional neural networks," *JURIKOM: Jurnal Riset Komputer*, vol. 9, no. 3, 2022. (in Indonesian)
[25] A. Mustafid, M. M. Pamuji, and S. Helmiyah, "A comparative study of transfer learning and fine-tuning on deep learning models for wayang dataset classification," *IJID: International Journal on Informatics for Development*, vol. 9, no. 2, pp. 100–110, 2020.
[26] S. Rajaraman, S. K. Antani, M. Poostchi, K. Silamut, M. A. Hossain, R. J. Maude, S. Jaeger, and G. R. Thoma, "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, vol. 6, no. e4568, 2018, doi: https://doi.org/10.7717/peerj.4568