Received: July 6, 2020. Revised: August 6, 2020.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Custom Convolutional Neural Network with Data Augmentation and Bayesian Optimization for Gram-Negative Bacteria Classification

Budi Dwi Satoto[1,2]* **Mohammad Imam Utoyo[3]*** **Riries Rulaningtyas[4]**
Eko Budi Koendhori[5]

[1]*Faculty of sains and Technology, University of Airlangga Surabaya, Indonesia*
[2]*Department of Informatics Engineering, University of Trunojoyo, Madura, Indonesia*
[3]*Department of Mathematics, University of Airlangga Surabaya, Indonesia*
[4]*Department of Physics, University of Airlangga Surabaya, Indonesia*
[5]*Department of Microbiology, Faculty of Medical, University of Airlangga Surabaya, Indonesia*
* Corresponding author's Email: m.i.utoyo@fst.unair.ac.id, budids@trunojoyo.ac.id

Abstract: One of the newest methods used in image classification is the Convolutional Neural Network. This method uses a large number of hidden layers to process data so that the resulting accuracy is excellent. However, this affects the time of the training process used. The selection of suitable architecture also determines the results of the classification. In this research, the author tries to reduce computational time by reducing the number of layers and using optimization. Transfer learning helps in the preparation of models using pre-trained data before, while data augmentation increases data variation. Bayesian optimization helps to find out momentum values and initial learning rate. The data source of this research is the primary image of Gram-negative bacteria from pneumonia patients. Data was collected at Dr. Soetomo's Microbiology Laboratory in Surabaya, Indonesia. Data distribution includes training, validation, and testing divided by percentage and proportional distribution of the number of files. This research used four classes of Gram-negative bacteria with a total of 1,000 images. An experimental comparison was made with a comparison of the Convolutional Neural Network architecture. The test results show an increase in accuracy by using aiming layers 26-34, having an accuracy range of 99.5% to 99.8%. The computational time required for the training process is around 2 minutes 30 seconds, with a momentum value of 0.92813 and an initial learning level of 0,00022397. The best accuracy errors were obtained at MSE 0.0025, RMSE 0.05, and MAE 0.0025.

Keywords: Gram-negative bacteria, Convolutional neural network, Transfer Learning, Data augmentation, Bayes optimization.

1. Introduction

Today, the use of machine learning is to solve problems both in the field of information systems and public health. One application is pattern recognition. It is due to the ability of the image identification process. One of the most recent methods of machine learning is Convolutional Neural Network (CNN). This method is believed to be able to recognize patterns because it has deep learning abilities [1]. If, in the previous machine learning method, the parameter determination is done by the user, then in this method, the parameters are carried out by the learning process on CNN. It also results in the possibility of overfitting or too many parameters in the neural network used [2].

To overcome this has been done, there are several studies conducted by researchers. Among the studies conducted by Kenneth P. Smith, 2017 do research in the Convolutional Neural Network method is used to classify bacterial images collected from 189 glass objects. The observation process is carried out without human intervention. The research has a sensitivity of 98.4% and specificity 75.0% for Gram-positive coccus bacteria in the form of paired chain colonies. The Gram-positive cocci in a single group had 93.2% and 97.2%. The rest were Gram-negative rod groups with sensitivity and specificity 96.3% and 98.1% [3]. CongBai pad 2018 tries to optimize the Alexnet architecture by looking at three criteria. Optimize the convolution layer, modify the fully connected and reconstruction of the hidden layer. The pooling layer replaces with the max-average Pooling layer. It is a non-linear activation function. A fully connected layer uses Maxout means desire output. The goal is to get better feature maps. A hidden segment is added to map the high-dimensional features into binary code [4].

Ashraf Darwish's research in 2019, used CNN with orthogonal learning particle optimization (OLPSO) algorithm. The goal is to find the optimal value of hyperparameter classification [5]. Elnaz Jahani Harav in 2018 proposed a 23-layer architecture that has an accuracy of 99.14% and 96.63%. This figure is slightly better compared to ResNet and GoogleNet. Food101 and UECFood-256 are the datasets used for testing. The results are better than GoogleNet. Another comparison of accuracy results similar to ResNet. Its is shows that when the number of layers decreases, the amount number of parameters used also reduces dramatically [6].

The purpose of this research is to assist visual observations that are still needed in hospital microbiology laboratories. The choice of convolutional neural networks is based on data objects that are processed in the form of images and is the latest identification research method currently being developed. This method has deep learning about the object being observed and has high accuracy. The disadvantage is that in some architectures, the computational time for the training process is still quite high. GAP, with previous research, is saving computing time while maintaining accuracy. The novelty offered is the Custom convolutional layer that is equipped with auto contrast, transfer learning, data augmentation, and optimization. The study also added data augmentation methods to overcome the limited amount of data and the use of Bayes optimization techniques to obtain ideal parameters during the training process.

2. Related work

This section discusses similar work about Gram negative-bacteria classification, convolutional neural network, transfer learning, auto contrast image, data augmentation, Bayes optimization, accuracy, and objective function.

2.1 Gram-negative bacteria classification

This germ can cause diseases such as pneumonia, meningitis, gonorrhea, bacterial dysentery, cholera, gastritis. The unique shape of their cellular envelopes provides virulence and protection from various chemicals. It is still a big challenge for modern doctors and scientists in connection with these cellular mechanisms and mitigating the impact of these resistance properties [7]. Biology laboratories use visual observations to research this.

Examples of Gram-negative bacteria are Acinetobacter, Pseudomonas aerugenusa, klebsiella pneumonia, and Escherichia Colli. Pseudomonas aerugenusa is a bacterium that has innovative and oxygenic properties. It can cause infection in patients with a decrease in endurance and become an essential nosocomial pathogen. The Gram-negative bacteria are shown in Fig. 1.

There are several steps taken by laboratory staff to identify Gram-negative bacteria:

1. Prepare samples taken from the sputum of the patient by taking into account the inclusion criteria, namely pneumonia patients
2. Taking samples to be tested for pathogenicity using the Phoenix machine. Testing using biochemical methods.
3. If the sample is purely pathogenic, then a sample is taken to be cultured on McConkey media and incubated with freezer temperature for 12-18 hours.
4. Bacterial colonies are formed during the incubation process. Each was given a patient id, name, suspect, category of bacteria and resistance
5. Taking samples of culture results in glass objects to be observed in a microscope with the help of immersion oil [8].

2.2 Convolutional neural network to classify gram-negative bacteria

In general, the process of identifying bacteria using image processing is carried out according to the stages shown in Fig. 2. Beginning with preparing the bacterial database from which the data acquisition process is needed.

In the Convolutional neural network (CNN) block, there are several main layers, including the Conventional Vocational Layer, Pooling layer, Normalization layer, Softmax, and fully connected layer. Some existing architectures use the Directed Acyclic Graph (DAG) network to ease performance [9].

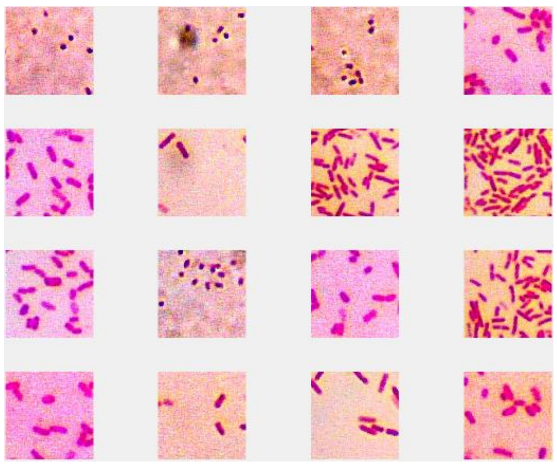


Figure. 1 Gram-negative bacteria

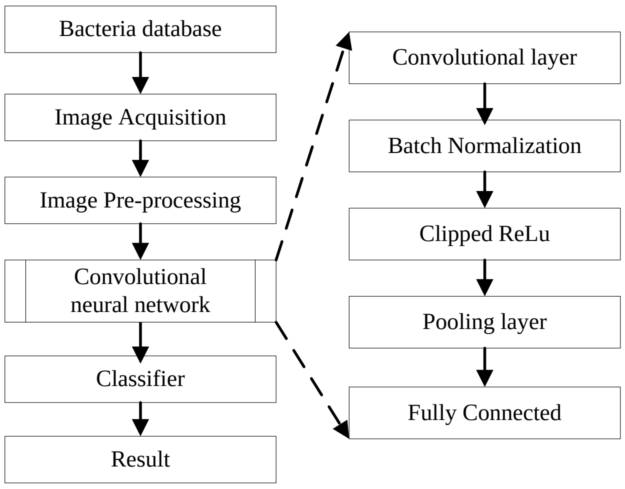


Figure. 2 Bacteria classification

2.2.1. Convolutional layer

Convolution can be generalized to several dimensions, where the features matrix of f (image) and g (filter) defined in the t set of integers. With two input dimensions $f(c, d)$, for example, models with width and height coordinates c and d. The output of convolution $h[c, d]$ can be written in analog Eq. (1) and discrete Eq. (2).

$$h(c, d) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(p, q) g(c - p, d - q) dpdq \quad (1)$$

$$h[c, d] = \sum_p \sum_q f[p, q] g[c - p, d - q] \quad (2)$$

Parameters that adjusted in the convolutional layer include the size of the kernel. It is called the filter size. The zero paddings (P) magnitude generally fits so that the spatial The desired output dimension has the same spatial input size dimensions ($P = F - 1/2$) with $F = Filter$ [10].

2.2.2. Normalization and rectifier linear unit

This layer normalizes input features through batch dimensions written in Eq. (3) and Eq. (4).

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_{ij} \text{ and } {\sigma_j}^2 = \frac{1}{m} \sum_{i=1}^{m} (x_{ij} - \mu_j)^2 \quad (3)$$

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sqrt{{\sigma_j}^2 + \epsilon}} \quad (4)$$

With x_{ij}=elementh input. μ_j=mean of input. σ_j = Standart deviation. The output of the Rectifier Linear Unit is $f(h) = max\,(0, h)$ [11]. Network studies non-linear values. ReLu formula written in Eq. (5)

$$ReLu(h) = \begin{cases} 0 & if\ h < 0 \\ x & if\ h \geq 0 \end{cases} \quad (5)$$

The Rectifier Unit (ReLu) will have an output value of 0 if the input is less than 0, but the output will be in the form of raw data if other than zero. That is, if the information is greater than 0, the production is the same as the input [12].

2.2.3. Pooling layer

The tool used to ensure the invariance of output y is the Pooling Layer. It used in combination with a convolutional layer. Network output after the pooling layer written in Eq. (6) [13].

$$n_{out} = \frac{n_{in} + 2P - F}{S} + 1 \quad (6)$$

With n_{in}=Length or High of Input, F=Length or Height of Filter, P=Zero Padding, S=Stride. The Pooling operation calculates a statistical summary of the nearest information using an arithmetic function [14].

2.3 Transfer learning

Technique or method that uses a model that previously trained in the dataset. If there are more $T1$ assignment data, features, and weights, carry out $T2$ assignments that have fewer data. [15]. Domain D is a two-element matrix that has marginal probability $P\ (X)$ and a feature space (fs) [16]. The negligible probability was written in Eq. (7)

$$P\ (X)\ with\ X = \{x_1, \ldots, x_n\},\ x_n \in \text{fs} \quad (7)$$

Here x_i represents a particular vector. Task T explained as two tuple elements of the label space

(ls), and the objective function, η. The objective function described as $P\ (ls \mid X)$ from a probabilistic point of view. [17]. $Y = \{y_1, \dots, y_n\}$, with $y_i \in ls$. The predictive function η learned from the feature vector relationship (x_i, y_i) where $x_i \epsilon fs, y_i \epsilon ls$. For each feature in the domain, η makes the appropriate label prediction $\eta(x_i) = y_i$ [18]. Task wrote in Eq. (8)

$$T = \{ls, P(Y|X)\} = \{ls, \eta\} \tag{8}$$

Given the source domain D_s which accommodates the T_s task source, as the D_T domain target and the τ_T task target. The gain information gives from D_s and T_s where $D_s \neq T_s$ or $\tau_s \neq \tau_T$. [19].

2.4 Bayes optimization

All Bayesian Optimization is a step to build a probability model using an objective function. Bayes Probability wrote in Eq. (9)

$$p(m|n) = \frac{p(m|n) * p(n)}{p(m)} \tag{9}$$

With $p(m|n)$ = probability hyperparameter that assesses the objective function. The output value can be written in Eq. (10).

$$p(a|b) = \begin{cases} l(a) & if\ b < b^* \\ g(a) & if\ b \geq b^* \end{cases} \tag{10}$$

Value of $b < b^*$ Shows the lower cost of the objective function of the threshold, labeled $l\ (a)$, and if it is larger, then it is labeled $g(a)$.

2.5 . Accuracy

The output of accuracy is a percentage comparison between the actual condition of the data and the results of predictions. TrP = True-Positive, TrN = True-Negative. It is written in Eq. (11).

$$Precision = \frac{TrP}{TrP+FaP} \quad Recall = \frac{TrP}{TrP+FaN} \tag{11}$$

Precision is a percentage ratio of correct positive predictions compared to the overall positive predicted results. Recall (Sensitivity) is a correct positive prediction ratio compared to whole accurate positive data. The F1 Score is written in Eq. (12) [20].

$$F1 - score = \frac{2x(RecallxPrecision)}{Recall+Precission} \tag{12}$$

2.6 An objective function on the CNN model

This function trains the model to be used using the stochastic gradient descent to make the objective loss function minimal. CrossLoss purpose (s) written in Eq. (13)

$$Loss = \frac{1}{N} \sum_{s \in S} (-\log(out_s[m])) \tag{13}$$

Where S is a collection of positions, N is the number of places of $out_s[m]$. It is the output value (Probability value) of node m at position s [21].

3. Methodology

The steps proposed to make improvements are shown in the Block diagram. This research uses primary data because secondary data are not available on the internet. This research used four types of Gram harmful bacteria shown in Fig. 3.

This research was carried out on Intel Core i-7 Laptop hardware with 8GB of RAM equipped with a

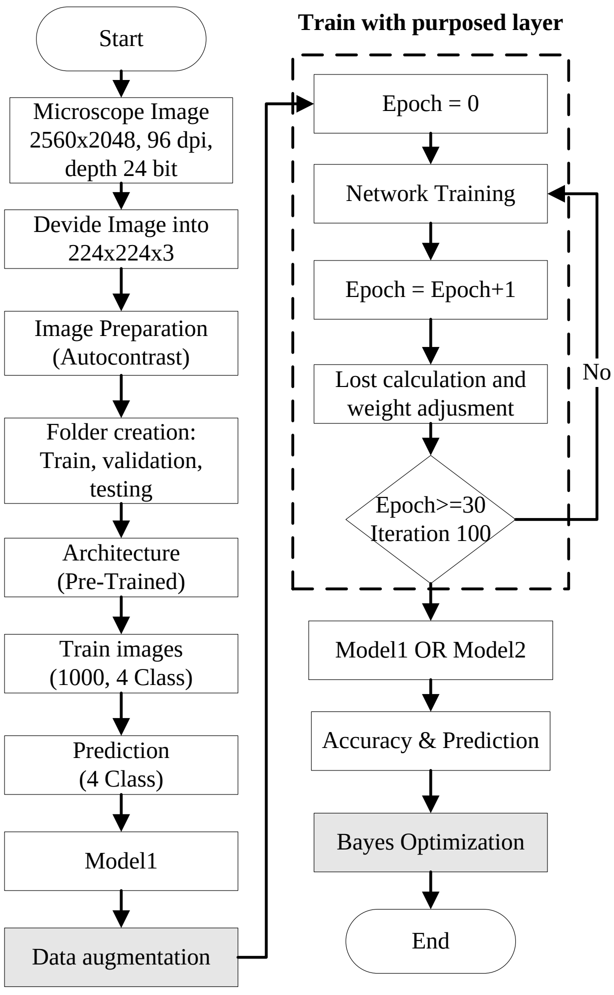

Start
Microscope Image 2560x2048, 96 dpi, depth 24 bit
Devide Image into 224x224x3
Image Preparation (Autocontrast)
Folder creation: Train, validation, testing
Architecture (Pre-Trained)
Train images (1000, 4 Class)
Prediction (4 Class)
Model1
Data augmentation
Train with purposed layer
Epoch = 0
Network Training
Epoch = Epoch+1
Lost calculation and weight adjusment
Epoch>=30 Iteration 100
No
Model1 OR Model2
Accuracy & Prediction
Bayes Optimization
End

Figure. 3 Research methodology

Received: July 6, 2020. Revised: August 6, 2020.

single GPU 4GB Nvidia GTX1050 Graphic Processing Unit. The software used is Matlab 2019a.

3.1 Data collection methods

Gram-negative bacteria image is taken from the Dr. Soetomo Hospital, Surabaya. For the period August 2018-August 2019 in the Microbiology Laboratory. The data used are primary data taken from 50 patients who were exposed to pneumonia with invitro criteria caused by Gram-negative bacteria. The size of the image produced by Optilab View is 2560x2048, with 96 dpi and 24-bit depth. The Lens magnification used 1000x magnification includes 10x the eyepiece and 100x the objective lens. The complete dataset used in the training folder shown in Table 1.

There are three folder setups, including training, validation, and test. The image divided function, namely datastore in MatLab. It is separated using several pictures and percentage portion.

3.2 Research steps

The research steps are explained as follows:

3.2.1. Preparation of data

Observation data is not used immediately because the sharpness level of the image should be improved at the beginning. In this research, auto contrast is used, the goal being that foreground and background appear more dominant. Auto Contrast formula computes the locally normalized luminescence via local mean subtraction and divides it by the local deviation. It is written in Eq. (13)

$$\hat{I}(m,n) = \frac{I(m,n)-\mu(m,n)}{\sigma(m,n)+C} \tag{13}$$

$\hat{I}(m,n)$ Normalized luminance, $\mu(m,n)$ =contrast normalization coefficient.

Suppose $I(m,n)$ domain is [0, 255], then C=1. If the field is [0, 1], then C=1/255. Mean value written in Eq. (14)

$$\mu(m,n) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} w_{k,l} I_{k,l}(m,n) \tag{14}$$

Table 1. Dataset detail in training folder

No	Bacterial name	Dimension	No. of images
1	Acinetobacter	[224 224 3]	250
2	P. Aerugenusa	[224 224 3]	250
3	Eschericia Colli	[224 224 3]	250
4	Klebsiella	[224 224 3]	250
		Total images	1000

To calculate the locally normalized luminescence, also known as mean subtracted contrast normalized (MSCN) coefficients. It is estimated the local mean. Within w is a Gaussian kernel of size (K, L) in Eq. (15)

$$\sigma(m,n) = \sqrt{\sum_{k=-K}^{K} \sum_{l=-L}^{L} w_{k,l}(I_{k,l}(m,n) - \mu(m,n))} \tag{15}$$

3.2.2. Transfer learning

Dataset creation to ensure the input layer accepts 224x224x3 image size. Split the sets of images into training and testing data. There are three primary data, including train, testing, and validation. Split can be done with a datastore image by dividing the portion of data using a randomized. Transfer Learning conducts pre-trained initials, opens the final layer fully connected, conducts training using new data, and ends by predicting and assessing network accuracy. The process is shown in Fig. 4.

Transfer learning cut the connection from google layer loss3-classifier that described as a Fully connected layer. After that proposed method connect 'pool5-drop_7x7_s1 transfer it into layer that usage in the output layer to get a new model.

The use of the number of layers affects overfitting. But keep in mind also that the more layers of learning get deeper and computing time increases. It is taken into consideration when preparing the proposed sheet, as shown in Fig. 5.

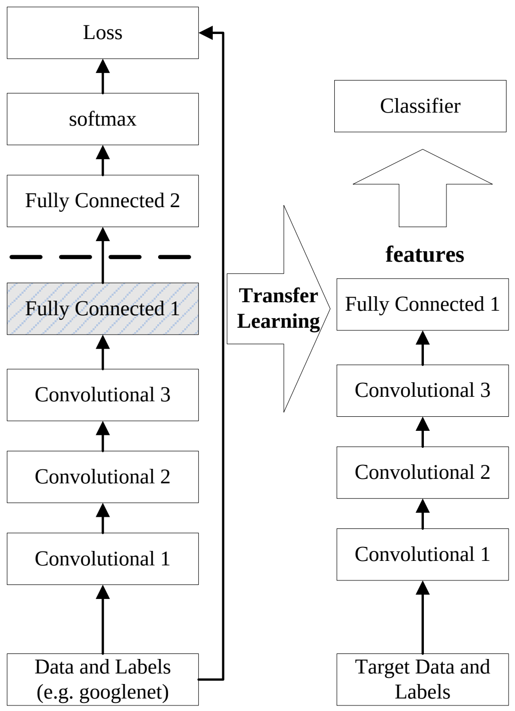


Figure. 4 Transfer learning

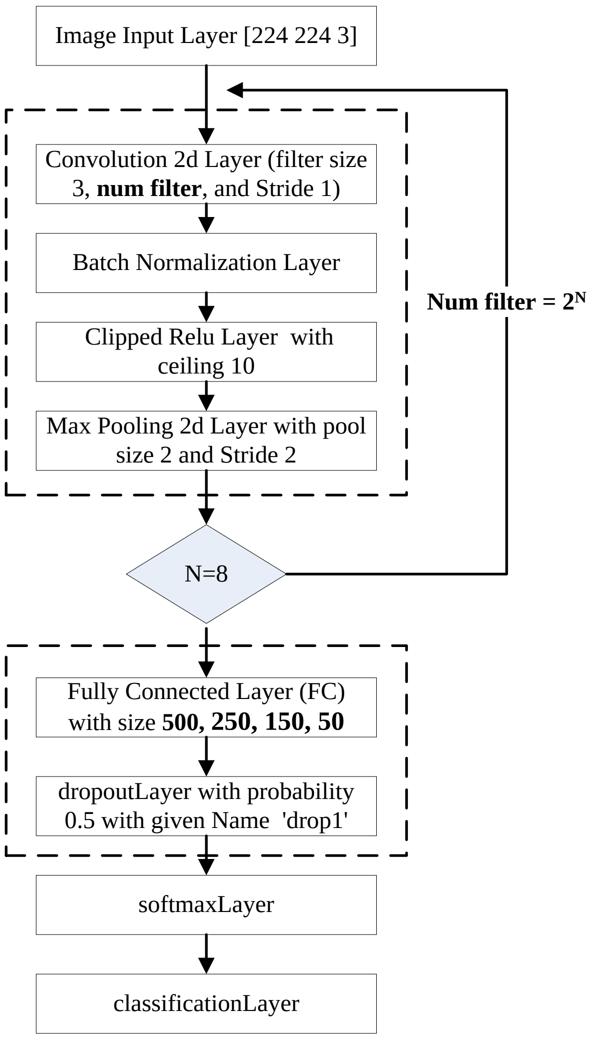


Image Input Layer [224 224 3]
Convolution 2d Layer (filter size 3, **num filter**, and Stride 1)
Batch Normalization Layer
Clipped Relu Layer with ceiling 10
Max Pooling 2d Layer with pool size 2 and Stride 2
Num filter = 2^N
N=8
Fully Connected Layer (FC) with size **500, 250, 150, 50**
dropoutLayer with probability 0.5 with given Name 'drop1'
softmaxLayer
classificationLayer

Figure. 5 Purposed layer CNN

At the convolution stage, a repetitive process is carried out with different dimensions of the image size so that the learning process is carried out more intensely. In the convolution layer, it takes several iterations between four to five times with different image dimensions so that the results of Feature maps become smoother. It affects the resulting accuracy. It refers to formula (3) that resume Neuron output will depend on the length and with of the input dimension, size of the filter, amount of padding, and several strides. The variation of this hidden layer will give feature maps and accuracy to become smooth.

3.2.3. Data augmentation

Data augmentation works by changing or modifying images so that the computer will detect that the modified model is a different image. But humans can still know that the modified image is the same picture. Rotation methods are written in Eq. (16).

$$A = \begin{pmatrix} \cos\theta & -sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{16}$$

Where θ is between 10 and 175 degrees, augmentation can improve the availability of the data that trained by CNN. The Scaling are x, y with a direction shown in Eq. (17)

$$A = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \tag{17}$$

By enlargement, the model gets additional data that can be useful for making models that can generalize better. Shears methods can do using Affine transformation shown in Eq. (18)

$$A = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \tag{18}$$

s defines the amount that $I(image)$ is sheared, and it is in the range of [0.1, 0.35]. Development performed in this research is to reverse the image horizontally, zoom in randomly, with a maximum zoom of 50% of the image size, and also rotate pictures randomly with a maximum degree of 90^o

3.2.4. Bayesian optimization

The Bayesian optimization approach is to use a Gaussian distribution. It makes this optimization better manage hyperparameter because hyperspace searching becomes more efficient, so the training process steps are less. The critical elements in the minimization using the Bayesian Optimization step. Firstly, The Bayesian procedure modifies the Gaussian process model for each new evaluation $f\ (x)$. Furthermore, the acquisition function $aq\ (x)$ will maximize to determine the next x point of evaluation. 'expected-improvement (EI),' which is the acquisition function, evaluates the number of expected improvements in the objective function. The form is shown in Eq. (19)

$$EI(x, Q) = E_Q[\max(0, \mu_Q(x_{best}) - f(x))] \tag{19}$$

With x_{best} = the location of the lowest posterior. $\mu_Q(x_{best})$ The means of the lowest value of the posterior. Bayes optimization first calculates x_{best} and $\mu_Q(x_{best})$. Bayes optimization uses probability-of-improvement (PI) by calculating the value of the new point x probability, which leads to better objective function values. Modified by the QA parameter "margin." The Formula PI has written in Eq. (20)

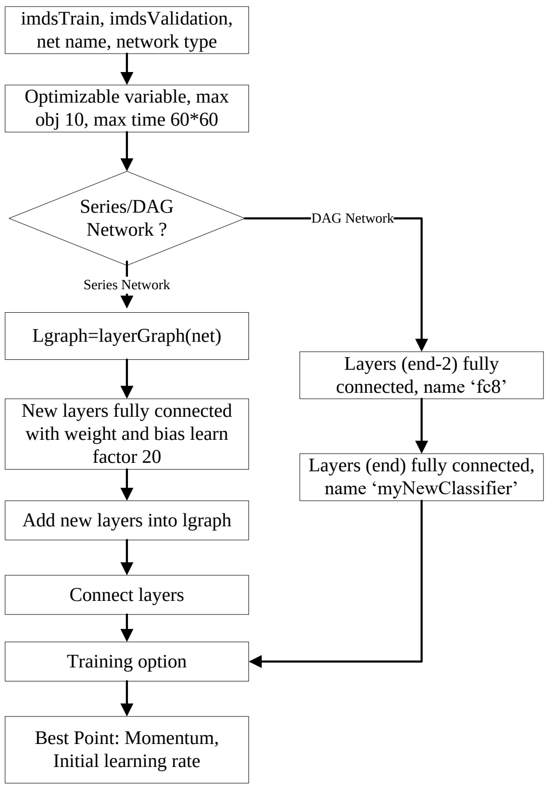

imdsTrain, imdsValidation, net name, network type
Optimizable variable, max obj 10, max time 60*60
Series/DAG Network ?
DAG Network
Series Network
Lgraph=layerGraph(net)
New layers fully connected with weight and bias learn factor 20
Add new layers into lgraph
Connect layers
Layers (end-2) fully connected, name 'fc8'
Layers (end) fully connected, name 'myNewClassifier'
Training option
Best Point: Momentum, Initial learning rate

Figure. 6 Bayesian optimization

$$PI(x,Q) = P_Q(f(x) < \mu_Q(x_{best}) - m) \quad (20)$$

Bayes optimization takes m as the estimated noise standard deviation. The process is shown in Fig. 6.

Bayesopt evaluates this probability, as written in Eq. (21)

$$PI = \Phi(v_Q(x)) \quad (21)$$

With $v_Q(x)$ written in Eq. (22)

$$v_Q(x) = \frac{\mu_Q(x_{best}) - m - \mu_Q(x)}{\sigma_Q(x)} \quad (22)$$

$\Phi(\cdot)$ = the unit normal cumulative distribution function CDF. The notation σ_Q = deviation of the Gaussian process, which is at position x. The Low Confidence Limit is obtained by looking at the function on the G curve. It has two standard deviations below the posterior average at each point written in Eq. (23)

$$G(x) = \mu_Q(x) - 2\sigma_Q(x) \quad (23)$$

$G(x)$ is the $2\sigma_Q(x)$ of the confidence envelope of the objective function model. The Bayes then maximizes the negative of G using Lower Confidence Bound (LCB) in Eq. (24)

$$LCB = 2\sigma_Q(x) - \mu_Q(x) \quad (24)$$

The stochastic gradient can be used in the objective function of the CNN model to make the objective loss function to a minimum. In cross-entropy, hidden layer neurons in movable and bound positions (s, m) of the CrossLoss purpose (s) written in Eq. (25)

$$Loss = \frac{1}{N} \sum_{s \in S} (-\log(out_s[m])) \quad (25)$$

N = the number of places, S = a collection of positions. $out_s[m]$ = output value from the probability of node m at position s [21]. Typically, the class with maximum likelihood is chosen when deciding.

3.2.5. Error calculation

Errors in the classification process can also be seen from Mean Square Error (MSE), Root Mean Square Error (RMSE), and the Mean Absolute Error (MAE). The three of them calculate misclassification using the difference between an actual label condition and a prediction label result written in Eq. (26)

$$RMSE = \sqrt{\frac{1}{N_d} \sum_{k=1}^{N_d} (y_k - \hat{y}_k)^2} \quad (26)$$

where N_d = number of data, y_k = actual label, $\hat{y}_k$= prediction label. While Mean Absolute Error (MAE) written in Eq. (27)

$$MAE = \frac{1}{n} \sum_{j=1}^{n} |y_j - \hat{y}_j| \quad (27)$$

With the absolute value of the difference, predicted results could be used. [20].

4. Result and discussion

In this section, this research emphasizes the use of CNN. The proposed method is to use a sufficient number of layers. The dataset used is a negative gram bacterial image obtained from primary data on pneumonia patients. The data size of images used 224x224 pixels, 96 dpi, and 24-bit depth (three channels). Data is processed with a single GPU GTX1050 4GB.

4.1 Pre-processing

The initial process before entering the input layer is to divide the image into three folders. These folders are train, validation, and testing. Each folder is filled with four bacterial image subfolders, namely, GNB Acinetobacter, GNB Aerugenusa, GNB Escherichia Colli, and GNB Klebsiella.

Received: July 6, 2020. Revised: August 6, 2020.

Figure. 7 bacteria Visualization in each folder

The bacterial database to be used is shown in the input layer, as shown in Fig. 7. In the train subfolder, 250 images are used so that the entire train folder contains 1000 images of bacteria.

4.2 Features map layer

A visualization feature layer can be taken on the segment that has a filter and weight. It can be seen in Fig. 8.

Convolutional layers and combined layers are part of a Feature Extraction. The feature extraction layer dimension is a 5x5x3 convolutional layer. This layer has a length of five pixels with a height of five pixels and a depth of three corresponding to the image channel. An activation map or feature map works with shift operation that works using "point" operations between input and screen values to produce output.

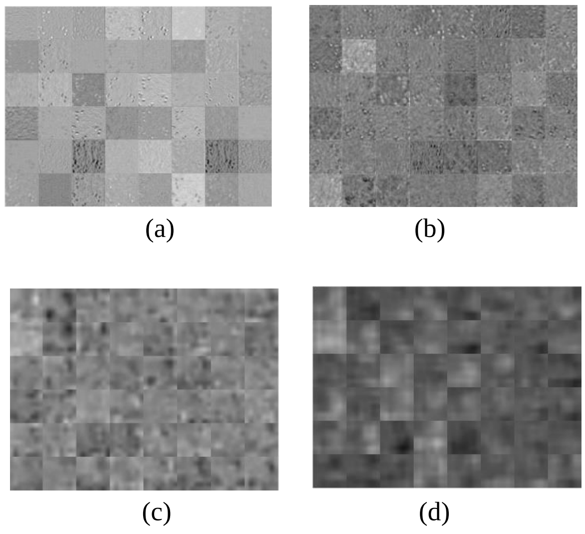


Figure. 8 Filter Visualization: (a) res2a_branch2b, (b) res3b_branch2b, (c) bn5a_branch2b, and (d) res5b_branch2b

4.3 Data augmentation

Methods to increase the amount of variation in training data, augmentation techniques are used. Additions made are scaling, rotation, shear, and reflection. The Scale formula is shown in Eq. (28)

$$A = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \quad (28)$$

A = Scaling matrix with the x or y-direction. The augmentation result is shown in Fig. 9.

Another technique uses for data augmentation is rotation. Each image I is rotated, represented by the following affine transformation. It is shown in Eq. (29)

$$A = \begin{pmatrix} \cos\theta & -sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad (29)$$

While θ is the degree, the value is between 10 and 175 degrees. The result is shown in Fig. 10.

Shears: Each image I is sheared, represented by the following affine transformation written in Eq. (30)

$$A = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \quad (30)$$

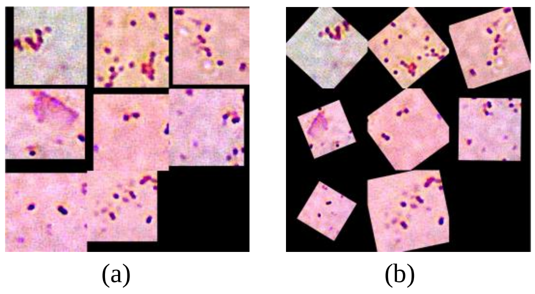


Figure. 9 Augmentation: (a) scaling and (b) rotation

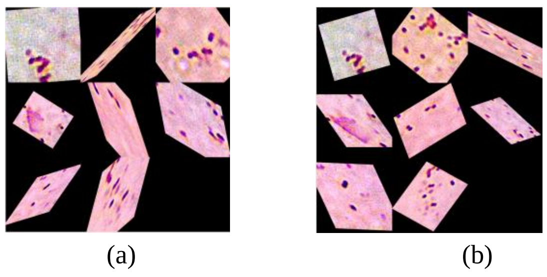
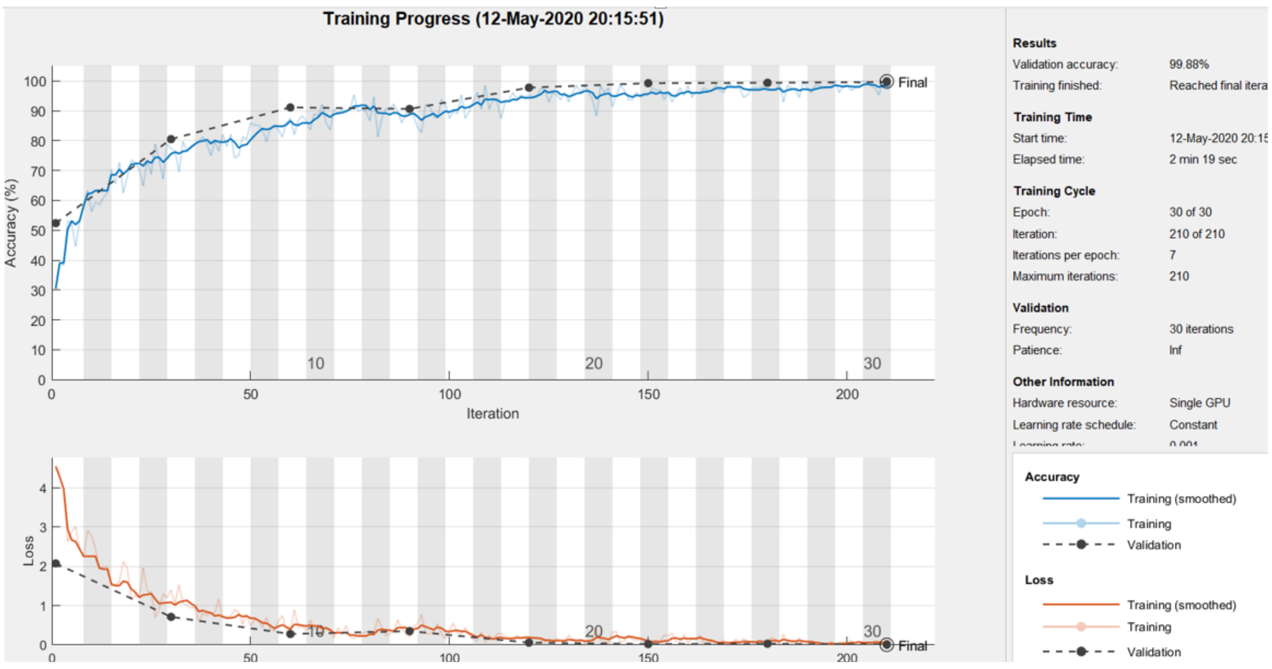


Figure. 10 Augmentation: (a) shears and (b) reflection

Received: July 6, 2020. Revised: August 6, 2020.



Results
Validation accuracy: 99.88%
Training finished: Reached final itera

Training Time
Start time: 12-May-2020 20:15
Elapsed time: 2 min 19 sec

Training Cycle
Epoch: 30 of 30
Iteration: 210 of 210
Iterations per epoch: 7
Maximum iterations: 210

Validation
Frequency: 30 iterations
Patience: Inf

Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.001

Accuracy
Training (smoothed)
Training
Validation

Loss
Training (smoothed)
Training
Validation

Figure. 11 Training process

4.4 Training process

The training process is run on 100 iterations with 30 epochs. Training can run well if we also use validation data. Several training optimizers can be used, including adaptive moment estimation (adam), Stochastic Gradient Descent with momentum (sgdm), and root means Square propagation (rmsprop). The training process setting is InitialLearnRate value of 0.001. MaxEpochs used is 30 with 100 iterations, Shuffle on each epoch, Using Validation Data with Validation Frequency 30. "verbose" to get the text train is made pure to compare the response of the signal. The Training progress results of accuracy approaching 99.8% seen in Fig. 11.

This optimization is needed so that the optimal process training results. The comparison between the training optimizer shown in Table 2. This table indicates that Adam became the most stable optimizer with the best accuracy and lowest computing time when custom using 18 layers and 34 layers.

Table 2. Training with optimizer

Custom Layer	Optimizer	Training Accuracy	Time-consuming (minute)
18 layer	Adam	99,70%	1,50
	SGDM	99,40%	1,32
	rmsprop	98,35%	2,12
26 layer	Adam	97,90%	3,75
	SGDM	97,45%	1,88
	rmsprop	95,05%	1,87
34 layer	Adam	100,00%	1,80
	SGDM	93,40%	1,18
	rmsprop	100,00%	1,15

4.5 Prediction after transfer learning

Transfer learning helps the new model training process become faster to converge. It can be seen from the prediction process produced after the merger of the initial model with the results of the training process model. The prediction is shown in Fig. 12. By taking sample images in the test data to do the test. The step is to get the value of YPred from the results of the validation data training. The YValidation value

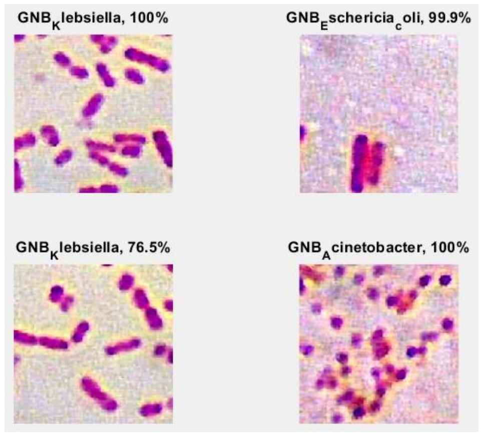


Figure. 12 Prediction result

is obtained from the label in the validation data. The accuracy calculation in the prediction process is done by getting the similarity value or similarity label to the amount of validation data used.

CNN predicts Klebsiella with a score of 100% and 76.5%. Escherichia Colli's prediction results are 99.9% and Acinetobacter 100%.

4.6 Confusion matrix

The classification results from training data define into the matrix. The confusion matrix calculates the difference of actual data with the prediction of data where the aim is to calculate the output class error. The evaluation is to compare the output class, which is not appropriate. Confusion matrix four types of bacteria shown in Fig. 13.

4.7 Bayes optimization

Bayesian optimization is done to get the optimum value of momentum and Initial learning rate after the fitting process. The training process will be repeated until the desired amount is met. Fig. 14 shows the results of the objective function.

In addition to establishing an objective function model, Bayes Optimization will calculate the Maximum Likelihood (MLE). It runs training on Convolutional neural networks using a framework to select loss functions. It is shown in Eq. (31)

$$\theta^{MLE} = argmax_\theta$$
$$= argmax_\theta \, log \prod_{i=1}^{N} p(y_i|x_i,\theta)$$
$$= argmax_\theta \sum_{i=1}^{N} \log p(y_i|x_i,\theta) \qquad (31)$$

Where $p\ (Y \mid X, \theta)$ represents the probability of an actual label on the training data, if the value of $p\ (Y \mid X, \theta)$ is 1, this shows that the model can describe the correct label. If a data train (X, Y) is provided consisting of connected N observations, the likelihood of the data train can be written as the sum of log probabilities. The two main types of loss functions are the mean squared error (MSE) and the Cross-entropy model. The MSE has written in Eq. (32)

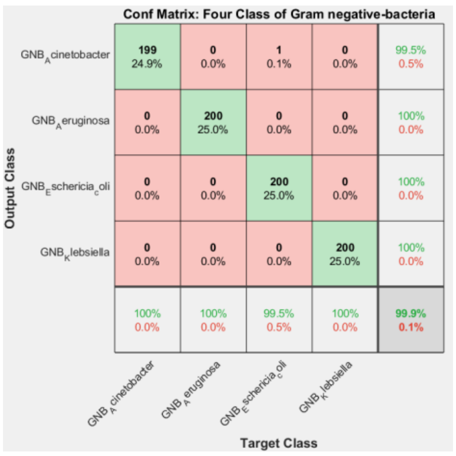
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \qquad (32)$$

with $MSE = Mean\ Square\ Error$, n=*number of data*, y_i=actual data and $\tilde{y}_i$=prediction data. CE or Cross entropy can be written in Eq. (33)

$$CE(p,q) = -\sum_x p(x) \log q(x) \qquad (33)$$

Where p = ground truth and q = network output. The candidate solution for optimization is done by selecting a series of weights called the objective function. Convolutional neural networks choose the right Loss function to overcome predictive modeling problems.

Fig. 15 shows that the third to seventh evaluation functions show the same minimum objective value, which means that at that point, the best momentum and initial learning rate results are obtained to obtain optimal results.


Conf Matrix: Four Class of Gram negative-bacteria

Output Class \ Target Class	GNB_Acinetobacter	GNB_Aeruginosa	GNB_Eschericia_coli	GNB_Klebsiella	
GNB_Acinetobacter	199 24.9%	0 0.0%	1 0.1%	0 0.0%	99.5% 0.5%
GNB_Aeruginosa	0 0.0%	200 25.0%	0 0.0%	0 0.0%	100% 0.0%
GNB_Eschericia_oli	0 0.0%	0 0.0%	200 25.0%	0 0.0%	100% 0.0%
GNB_Klebsiella	0 0.0%	0 0.0%	0 0.0%	200 25.0%	100% 0.0%
	100% 0.0%	100% 0.0%	99.5% 0.5%	100% 0.0%	99.9% 0.1%

Figure. 13 Confusion matrix

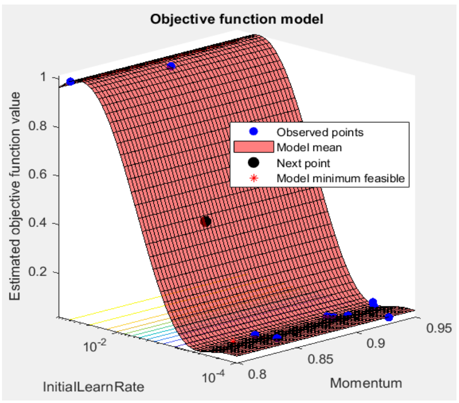


Figure. 14 Objective function model

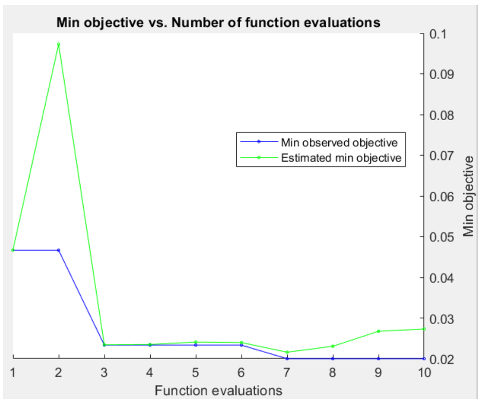
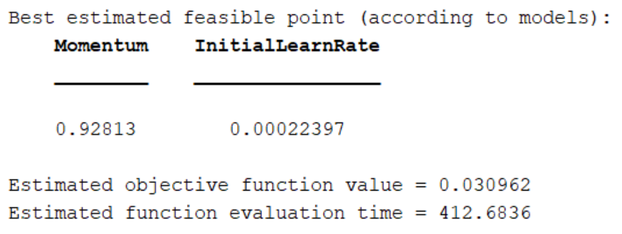


Figure. 15 Comparison of minimum objectives against the number of evaluation functions



```
Best estimated feasible point (according to models):
    Momentum     InitialLearnRate
    ________     ________________

    0.92813         0.00022397

Estimated objective function value = 0.030962
Estimated function evaluation time = 412.6836
```

Figure. 16 Momentum and Initial learning rate

The maximum likelihood selection finds the optimal value for the parameter. Momentum parameters prevent the system from converging to the local minimum and serve to stabilize the learning process, as well as the learning rate parameter to accelerate the pace of learning as written in Eq. (34)

$$w \coloneqq w - \eta \nabla Q_i(w) + \alpha \Delta w \tag{34}$$

w =parameter which minimize $Q_i(w)$, η =learning rate, α =decay factor (0< $\alpha < 1$), "Coefficient of Momentum," which is the percentage of the gradient, retained every iteration. The result of momentum for this current research shown in Fig. 16.

4.8 Comparison between CNN methods

Table 3 compares the performance of each architecture in CNN to classify Gram-negative bacteria. There are four class data of bacteria, namely Acinetobacter, pseudomonas aerugenusa, Escherichia Colli, and Klebsiella pneumonia. Several parameters to compare those become indicators are the number of layers used by CNN, Number of output Class, Training accuracy, time consumes in the training process, Precision, Recall, and F-1 Score. Another error parameter used to compare is MSE, RMSE, and MAE. The results of the comparison show that the purposed method has the specifications of the number of layers, the accuracy of the training results is above 99.88%, and the time spent on the training process ranges around 2 minutes.

4.8.1. The layers vs. time-consuming

The use of layers is intended to increase accuracy sensitivity. The more significant number of layers, the higher the accuracy value. But the consequences are the time used is getting longer.

Fig. 17 shows that Densenet 201 has the highest number of layers, 709 layers. This large number of layers affects the computational time needed, which reaches 335 minutes 2 seconds, which means about 5.6 hours to conduct training data. Purposed layers are using 34 and 18 segments. In the comparison graph, the results of the time consumption required for the process show that the more layers are used, the more time consuming it is needed. During the training process, learning is done deeper. It can be observed from the beginning of the process that with a multi-layer architecture, the initial training shows a lot of ripple or fluctuation. Still, in line with the training process, the ripple decreases because the network will continue to learn. The process will take a long time, but the resulting accuracy can always be maximized.

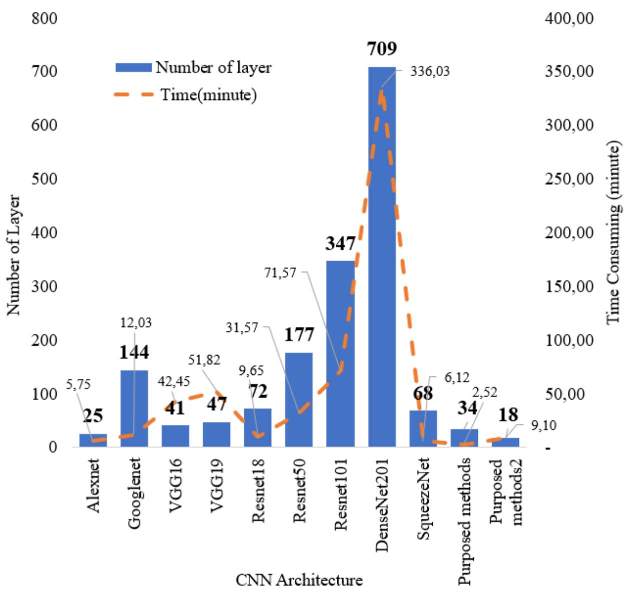


Figure. 17 Number of layer vs. time-consuming

Table 3. CNN Comparison

Models	No.of layer	Class	Training Accuracy	Time (minute)	Precision	Recall	F-1 Score	MSE	RMSE	MAE
Alexnet	25	4	99,75%	05:26	99,75%	99,75%	99,75%	0,00630	0,07910	0,00370
Googlenet	144	4	99,75%	12:02	99,75%	99,75%	99,75%	0,00630	0,07910	0,00370
VGG16	41	4	99,63%	42:27	99,62%	99,62%	99,61%	0,00740	0,08410	0,00510
VGG19	47	4	100,00%	51:49	100,00%	100,00%	100,00%	-	-	-
Resnet18	72	4	99,75%	09:39	99,75%	99,75%	99,75%	0,00250	0,05000	0,00250
Resnet50	177	4	99,88%	31:34	99,88%	99,88%	99,88%	0,00500	0,07070	0,00250
Resnet101	347	4	100,00%	71:34	100,00%	100,00%	100,00%	-	-	-
DenseNet201	709	4	100,00%	336:02	100,00%	100,00%	100,00%	-	-	-
SqueezeNet	68	4	99,50%	06:07	99,50%	99,51%	99,50%	0,00880	0,09350	0,00630
Purposed Methods-	34	4	100,00%	02:31	100,00%	100,00%	100,00%	-	-	-
Purposed Methods-	18	4	99,75%	01:54	99,75%	99,75%	99,75%	0,00250	0,05000	0,00250

4.8.2. Comparison of accuracy

In the accuracy stage, the comparison is made by looking at the values of Precision, Recall, and F-1 measure, as shown in Fig. 18.

Comparisons are made to obtain differences in the accuracy of the existing architecture with the proposed layers. The accuracy comparison shows that the classification results using 18-layers and 34-layer structures can already meet the desired criteria. It is related to the computational time needed in the training process also does not take a long time. In the 18- purposed layer, it takes 9 minutes 6 seconds, and in the 34-purposed layer, it takes only 2 minutes 30 seconds. It is supported by the results of the image prediction and confusion matrix, which shows a range between 99.5% to 98.8%. But keep in mind that when the training process runs, the more convolution layers, the tighter and the smaller the ripple factor will be. When it gets closer, the prediction process is more accurate but runs slowly. It means several layers needed, but not less depending on the number of convolution layer and dropout needed.

4.8.3. Comparison of classification errors

This research has three indicators used to identify the errors, namely MSE, RMSE, and MAE. Error comparison results show that the proposed method has a minimum error between zero to 0.05. It can be compared with the 68-layer squeezed. The purpose has only 18 and 34 layers. Other comparisons can be made by looking at the number of layers and

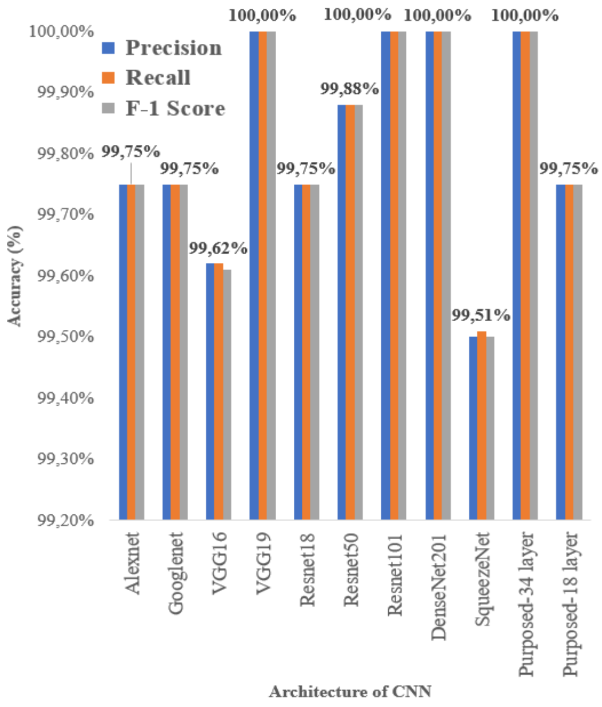


Figure. 18 Comparison of accuracy

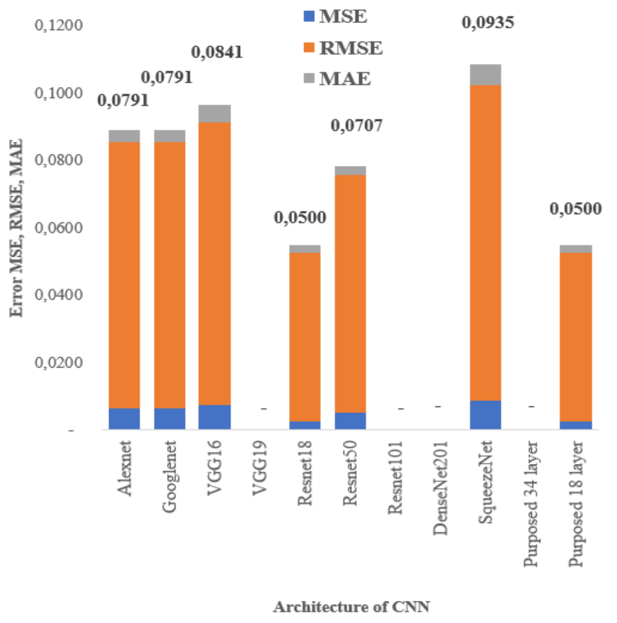


Figure. 19 Comparison of MSE, RMSE, and MAE

computational time. Densenet 201 also has excellent accuracy results, but the computational time required is very high at around 5 hours. The comparison of the training process shown in Fig. 19.

Densenet error results are too shallow. It given the accuracy of the resulting densenet is almost equal to the purposed method. But the purposed methods need a training time of only about 2 minutes. Then this research can be used as a reference for further analysis.

4.8.4. Comparison of the training optimization

The fitting process of this research uses Bayesian optimization to get the ideal value of momentum and average initial learning. Optimizable The desired variable to be a target is in the range of 0.8 to 0.95, and the initial learning rate is in the range $1x10^{-4}$ to $1x10^{-2}$. Furthermore, the Bayes option used has a Maxobj value of 10, which means a maximum of 10 re-training and a maximum of time 60 x 60, which means no more than 3600 seconds. From the comparison graph, it can be concluded that the number of layers influences the convergence of accuracy, which can be seen from the number of re-training processes carried out. During the re-training process, the objective function and minimum fair values will be updated until the maximum actual agreement is reached, the 10th training. When the maximum amount is reached, the process will stop to display the desired value of momentum and initial learning.

Table 4. Total time to find the best point of optimization

Models	No.of layer	No.of Re-training	Momen tum	learning Rate	time (minute)
Alexnet	25	9	0,80389	0,000740	66,51
Googlenet	144	8	0,94422	0,000123	66,13
VGG16	41	5	0,90843	0,000505	71,66
VGG19	47	5	0,93668	0,000197	70,34
Resnet18	72	8	0,90001	0,000220	62,93
Resnet50	177	8	0,89737	0,000887	62,96
Resnet101	347	6	0,82034	0,000113	68,088
DenseNet201	709	6	0,94086	0,000144	62,35
SqueezeNet	68	8	0,80725	0,000204	60,87
purposed methods	34	7	0,92302	0,000829	61,04
purposed methods2	18	8	0,91461	0,000888	64,58

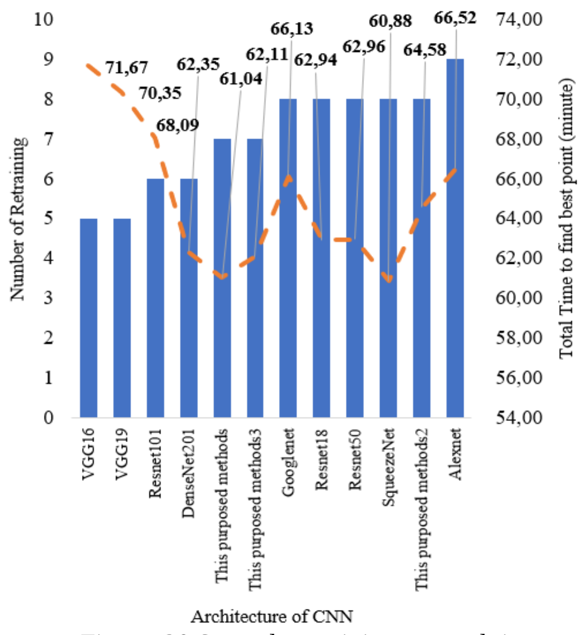


Figure. 20 Several re-training vs. total time

To get the ideal value of momentum and initial learning rate to achieve stability, shown in Table 4. Vgg19 and resnet101 have the smallest re-training value, which means they have reasonably good confidence by only doing 5-6 training to get the benefit of optimization. However, the total time to look for this value is longer than the purposed methods ranging from 8-10 minutes. Intended plans 34 layers show the number of Re-training is less than google net and densenet with computational time to find the ideal value of momentum and learning rate better, ranging from 1-5 minutes.

This method gets 60 minutes in a total time comparable to the squeezenet with seven re-training processes.

4.8.5. Comparison with another dataset

This research also tries to use the secondary data DIBAS (Digital Image of Bacterial Species) dataset [22]. There are 20 images divide into 250 images per class. Klebsiella pneumonia replaced with Neisseria gonorrhoeae because it was not available in this current dataset. The author also compares the proposed method using the Bird, Fruit, Animal, and Fruit dataset from the Kaggle.com image dataset. Each Class folder contains 400 images [23]. The comparison has shown in Table 5.

Conclusion

This research has a GAP with previous research in terms of saving computing time while maintaining accuracy results. The novelty offered is the CNN Custom layer, which is equipped with auto contrast, transfer learning, data augmentation, and optimization. The training data used amounted to 1000 data for four classes of Gram-negative bacteria. Efficiency can be said to be good, using 18-34 layers. The results of the accuracy and validation of the training process are in the range of 95% to 99.8%, with training process time starting from 2 minutes 30 seconds. Optimization with Bayesian Optimization

Table 5. Comparison with other dataset

No	Data base	No.of layer	Size of image	Class	Training Accuracy	Time (Minute)	Precision	Recall	F-1 Score	MSE	RMSE	MAE
1	DIBAS Bacteria	34	[224 224 3]	4	99,88%	01:49	0,9988	0,9988	0,9988	0,0013	0,0354	0,0013
		26	[224 224 3]	4	99,63%	01:50	0,9963	0,9963	0,9963	0,0075	0,0866	0,0050
2	Bird	34	[224 224 3]	4	99,61%	00:32	0,9961	0,9962	0,9961	0,0352	0,1875	0,0117
		26	[224 224 3]	4	92,31%	00:30	0,9531	0,9578	0,9548	0,3398	0,5830	0,1211
3	Fruit	34	[224 224 3]	4	100,00%	05:56	1,0000	1,0000	1,0000	-	-	-
		26	[224 224 3]	4	100,00%	03:10	1,0000	1,0000	1,0000	-	-	-
4	Animal	34	[224 224 3]	4	99,70%	03:13	0,9977	0,9977	0,9977	0,0047	0,0685	0,0031
		26	[224 224 3]	4	100,00%	03:13	1,0000	1,0000	1,0000	-	-	-
5	Flower	34	[224 224 3]	4	98,67%	03:22	0,9867	0,9869	0,9868	0,0469	0,2165	0,0234
		26	[224 224 3]	4	99,92%	03:34	0,9992	0,9992	0,9992	0.0070	0.0836	0.0023

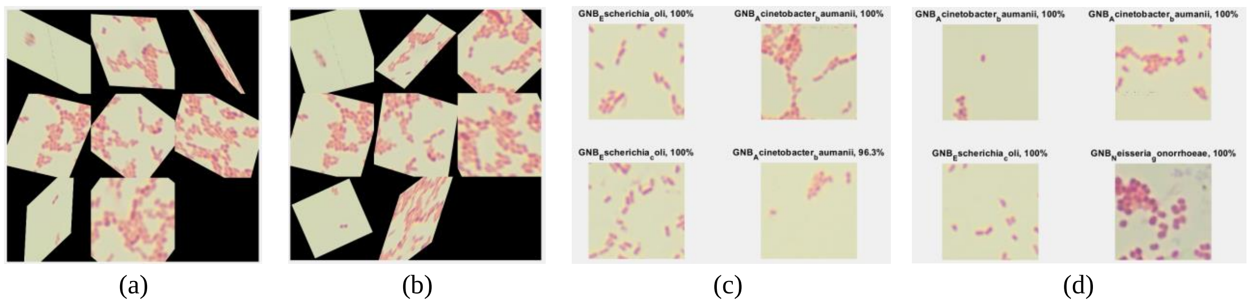

GNB$_E$scherichia$_c$oli, 100%
GNB$_A$cinetobacter$_b$aumanii, 100%
GNB$_E$scherichia$_c$oli, 100%
GNB$_A$cinetobacter$_b$aumanii, 96.3%
GNB$_A$cinetobacter$_b$aumanii, 100%
GNB$_A$cinetobacter$_b$aumanii, 100%
GNB$_E$scherichia$_c$oli, 100%
GNB$_N$eisseria$_g$onorrhoeae, 100%
(a)
(b)
(c)
(d)

Figure. 21 : (a) DIBAS data augmentation 1, (b) DIBAS data augmentation 2, (c) prediction 1, and (d) prediction 2

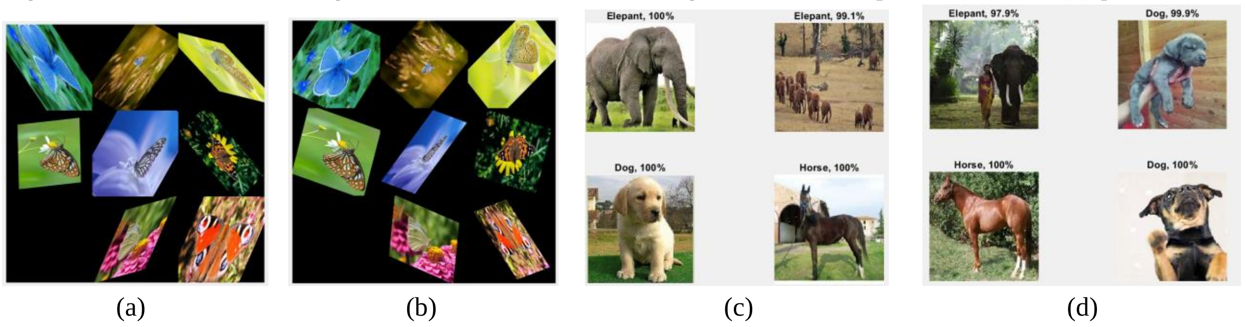


Figure. 22 : (a) animal data augmentation 1, (b) animal data augmentation 2, (c) prediction 1, and (d) prediction 2

gets momentum value at 0.92813, and the initial learning level is 0.00022397. The best accuracy errors were obtained at MSE 0.0025, RMSE 0.05, and MAE 0.0025. The scientific contribution to this research is saving training time when compared to the existing architecture. Purposed 18-34 layer is comparable to a 25-layer of the alexnet. However, the computational time needed for the training process is 40-50% more efficient. In the data training process without auto contrast, the ideal number of layers to maintain accuracy stable is at 26-34 layers. The work feature that can be done is comparing the parameters used by layers in each CNN architecture.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, Budids, and Imam; methodology, Budids, and Riries; software, Budids; validation, Riries, and Eko; formal analysis, Eko; investigation, Riries, and Imam; resources, Budids and Eko; data curation, Budids, and Eko; writing—original draft preparation, Budids, and Imam; writing—review and editing, Imam and Riries; visualization, Budids and Riries; supervision, Imam and Eko; project administration, Imam; funding acquisition, Imam and Riries.

Acknowledgments

The author would like to thank Dr. Soetomo Surabaya hospital's microbiology laboratory for providing the primary data of pneumonia patients. This work supported by the Ministry of Research, Technology and Higher Education, Grant 2020, with contract number: 812 / UN3.14 / PT / 2020.

References

[1] S. Vieira, W. H. L. Pinaya, and A. Mechelli, "Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications", *Neuroscience and Biobehavioral Reviews*, Vol. 74, pp. 58-75, 2017.

[2] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Impact of deep learning-based dropout on shallow neural networks applied to stream temperature modelling", *Earth-Science Reviews*, Vol. 201, pp. 103076, 2020.

[3] Smith, Kenneth, P. Kang, and Anthony, "Automated Interpretation of Blood Culture Gram Stains by Use of a Deep Convolutional Neural Network", *Journal of Clinical Microbiology*, Vol. 56, No. 3, 2018.

[4] C. Bai, L. Huang, X. Pan, and J. Zheng, "Optimization of deep convolutional neural network for large scale image retrieval", *Neurocomputing*, Vol. 303, pp. 60-67, 2018.

[5] A. Darwish, D. Ezzat, and A. E. Hassanien, “An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis”, *Swarm and Evolutionary Computation*, Vol. 52, p. 100616, 2020.

[6] E. Jahani Heravi, H. Habibi Aghdam, and D. Puig, “An optimized convolutional neural network with bottleneck and spatial pyramid pooling layers for classification of foods”, *Pattern Recognition Letters*, Vol. 105, No, pp. 50-58, 2018.

[7] J. E. Sykes, “Chapter 36 - Gram-negative Bacterial Infections”, *In: Canine and Feline Infectious Diseases*, Saint Louis. W. B. Saunders, pp. 355-363, 2014.

[8] T. M. Pham, M. Kretzschmar, X. Bertrand, and M. Bootsma, “Tracking Pseudomonas aeruginosa transmissions due to environmental contamination after discharge in ICUs using mathematical models”, *PLOS Computational Biology*, Vol. 15, No. 8, p. e1006697, 2019.

[9] X. S. Yang, “8 - Neural networks and deep learning”, *In: Introduction to Algorithms for Data Mining and Machine Learning, Academic Press*, pp. 139-161, 2019.

[10] A. J. E. Kell, D. L. K. Yamins, E. N. Shook, and S. V. Norman-Haignere, “A Task-Optimized Neural Network Replicates Human Auditory Behavior, Predicts Brain Responses, and Reveals a Cortical Processing Hierarchy”, *Neuron*, Vol. 98, No. 3, pp. 630-644. e16, 2018.

[11] A. B. Risum and R. Bro, “Using deep learning to evaluate peaks in chromatographic data”, *Talanta*, Vol. 204, No, pp. 255-260, 2019.

[12] M. Svanera, M. Savardi, S. Benini, and A. Signoroni, “Transfer learning of deep neural network representations for fMRI decoding”, *Journal of Neuroscience Methods*, Vol. 328, No, p. 108319, 2019.

[13] S. Ma, T. Huang, S. Li, and J. Huang, “MCSM-Wri: A Small-Scale Motion RecognitionMethod Using WiFi Based on Multi-Scale Convolutional Neural Network”, *Sensors (Basel, Switzerland)*, Vol. 19, No. 19, p. 4162, 2019.

[14] V. Suárez-Paniagua and I. Segura-Bedmar, “Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction”, *BMC bioinformatics*, 19, 209 DOI: 10.1186/s12859-018-2195-1, 2018.

[15] L. Ying and L. Boqin, “Application of Transfer Learning in Task Recommendation System”, *Procedia Engineering*, Vol. 174, pp. 518-523, 2017.

[16] H. Tercan, A. Guajardo, J. Heinisch, T. Thiele, “Transfer-Learning: Bridging the Gap between Real and Simulation Data for Machine Learning in Injection Molding”, *Procedia CIRP*, Vol. 72, pp. 185-190, 2018.

[17] H. H. Zhuo and Q. Yang, “Action-model acquisition for planning via transfer learning”, *Artificial Intelligence,* Vol. 212, pp. 80-103, 2014.

[18] S. M. Salaken, A. Khosravi, T. Nguyen, and S. Nahavandi, “Extreme learning machine based transfer learning algorithms: A survey”, *Neurocomputing*, Vol. 267, pp. 516-524, 2017.

[19] D. Han, Q. Liu, and W. Fan, “A new image classification method using CNN transfer learning and web data augmentation”, *Expert Systems with Applications*, Vol. 95, pp. 43-56, 2018.

[20] V. Kotu and B. Deshpande. “Chapter 8 - Model Evaluation”, *In: Data Science (Second Edition),* Morgan Kaufmann, pp. 263-279, 2019.

[21] H. Chen, Q. Dou, L. Yu, and J. Qin. “Chapter 6 - Deep Cascaded Networks for Sparsely Distributed Object Detection from Medical Images”, *In: Deep Learning for Medical Image Analysis*, Academic Press, pp. 133-154, 2017.

[22] B. Zieliński, A. Plichta, K. Misztal, and P. Spurek, “Deep learning approach to bacterial colony classification”, *PLOS ONE*, Vol. 12, No. 9, pp. 1-14, 2017.

[23] D. Thi Phuong Chung and D. Van Tai, “A fruits recognition system based on a modern deep learning technique”, In: *Proc. of Journal of Physics: Conf. Series*, Vol. 1327, p. 012050, 2019.