Received: July 31, 2021. Revised: October 6, 2021. 616

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Modified Flower Pollination Algorithm for Disease Identification in Swine

Yulianto Triwahyuadi Polly[1,2] Sri Hartati[3]* Suprapto[3] Bambang Sumiarto[4]

[1]*Doctoral Program Department of Computer Science and Electronics, Faculty of Mathematics and Natural Science, Universitas Gadjah Mada, Yogyakarta, Indonesia*
[2]*Department of Computer Science, Faculty of Science and Engineering, Universitas Nusa Cendana, Kupang, Indonesia*
[3]*Department of Computer Science and Electronics, Faculty of Mathematics and Natural Science, Universitas Gadjah Mada, Yogyakarta, Indonesia*
[4]*Department of Veterinary Medicine, Faculty of Veterinary Medicine, Universitas Gadjah Mada, Yogyakarta, Indonesia*
* Corresponding author's Email: shartati@ugm.ac.id

Abstract: Pigs have become an essential part of the cultural and economic life of the people in Nusa Tenggara Timur (NTT) Province. Diseases in pigs significantly affect the success of pig farming. Identification of disease in pigs is a classification problem. Metaheuristic algorithms are widely used in Neural Network (NN) optimization to solve classification problems. Flower Pollination Algorithm (FPA) is grouped into a metaheuristic algorithm that has been commonly used in optimization cases in the real world. To improve FPA performance, this study proposes replacing the FPA step vector parameter, namely Levy distribution, with Newton Polynomial Quadratic Interpolation (NPQI), known as Quadratic Interpolation Flower Pollination (QIFP). Quadratic Interpolation Flower Pollination Neural Network (QIFPNN), Flower Pollination Neural Network (FPNN), Bat Neural Network (BANN), and Particle Swarm Optimization Neural Network (PSONN) algorithms were used to train NN in real cases of disease identification in pigs, covering 11 diseases with 68 clinical symptoms. The results showed that the proposed algorithm, namely QIFPNN, outperformed FPNN, BANN, and PSONN in classification accuracy. QIFPNN is also able to improve classification accuracy and speed of convergence when compared to FPNN. QIFPNN and FPNN, respectively, provide 82.6159 % and 67.4766 % accuracy, and the training time is 6056.240 seconds and 6555.179 seconds. QIFPNN accuracy increased by 22.40%, and training time was 7.61 % faster. It concluded that QIFPNN could be used as a complementary model in disease identification in pigs.

Keywords: Disease identification in pigs, Flower pollination algorithm, Quadratic interpolation, Neural network.

List of symbols

X	Population
x_i^t	The pollen position vector or the i-th solution vector in iteration t
x_{ic}^t	The c-th element of the i-th solution vector in iteration t
g^*	The best solution vector found among all solutions in the current iteration
L	The step vector of the Levy distribution
γ	The scaling factor that controls the step size
ϵ	Random walk
Q	Quadratic interpolation step vector
P_m	The polynomial of degree m
(r_m, f_m)	Data point (abscissa, ordinate)
r^*	Best abscissa
$LbReal$	The lower limit of the original search space
$UbReal$	The upper limit of the original search space
Lb	The lower limit of the search space
Ub	Upper limit of search space
xx_e	The e-th neuron in the input layer
vv_{eo}	Connection weight between the input layer and hidden layer

vv_{0o}	The bias towards the hidden layer
zz_o	The o-th neuron in the hidden layer
ww_{ow}	Connection weight between the hidden layer and output layer
ww_{0w}	The bias towards the output layer
yy_w	The w-th neuron in the output layer

Subscripts

n	Population size or number of solutions
t	Iteration
d	Dimensions or number of solution variables
i, j, k	Solution (1, 2, …, n)
c	Dimension index (1, 2, …, d)
m	Interpolation degree
h	Number of neurons in the input layer
e	The index of the neurons in the input layer (1, 2, …, h)
v	Number of neurons in the hidden layer
o	The index of the neurons in the hidden layer (1, 2, …, v)
z	Number of neurons in the output layer
w	The index of the neurons in the output layer (1, 2, …, z)

1. Introduction

The agricultural sector in the province of nusa tenggara timur (NTT) Indonesia is still the economic base of the people in rural areas, controls the livelihoods of most of the population and absorbs a lot of labor. The economic structure of NTT is still dominated by the agricultural sector, where its contribution to the formation of gross regional domestic product (GDP) during 2016-2019 is between 28.00 percent to 29.03 percent. From this agricultural sector, the livestock sub-sector contributes between 9.44 percent to 9.46 percent. Other sub-sectors contribute less than the livestock sub-sector, so it can be said that the livestock sub-sector is the primary buffer for the agricultural sector [1].

Traditionally, pigs have played an essential role in various cultural activities, where these events are embedded in the social order of the people of NTT while also being a source of protein for domestic consumption. The health of pigs has been compromised. These disorders are caused by diseases, such as worm infections, dystocia, endometritis, myiasis, mastitis, pneumonia, eye inflammation, retensio secundinarum, scabies, streptococcosis, and septichaemia epizootica (Source: iSIKHNAS).

Identification of disease in pigs is a classification problem. Several researchers have researched classification using NN in the management of pigs. J. Shao [2] investigated the behavior of pigs adapting to temperature using images of pig behavior in cages, then classified using neural network (NN). Y. Wang, W. Yang, P. Winter, and L. Walker [3] created a machine vision to determine pig liveweight based on pig image using NN. A. Apichottanakul, S. Pathumnakul, and K. Piewthongngam [4] applied NN to measure the average weight of pigs. S. K. Biswas, B. Baruah, B. Purkayastha, and M. Chakraborty [5] used NN for swine flu diagnosis. NNs often use gradient descent to update weights in the training process. However, gradient descent can result in the training process being trapped in the local optima [6]. This limitation of gradient descent encourages researchers to use metaheuristic algorithms for updating weights in the NN training process [7].

Flower pollination algorithm (FPA) is grouped into a metaheuristic algorithm that works stochastically inspired by nature. FPA imitates flower pollination behavior to get the best flowers. There are two pollination models: cross-pollination occurs from pollen from flowers of other plants, and self-pollination occurs from pollen from the same or different flowers on the same plant [8]. In this study, a new variant of FPA, namely quadratic interpolation flower pollination (QIFP), is proposed to improve cross-pollination ability in searching for solutions. Quadratic interpolation (QI) uses a parabolic curve to fit the objective function near the optima [9]. The use of QI in metaheuristic algorithms is auspicious in finding global optimum solutions [9–11] and very fast convergence to optimal solutions [9, 11].

Both gradient and linear interpolation techniques are based on two-point information. Inspired by the use of the gradient technique in gradient descent and its drawbacks, it is proposed to use QI in exploring promising areas based on information on three fitnesses, namely the prominent pollen and the other two pollens. QIFP views Levy's step vector in FPA as a step length. QI against three fitness produces a quadratic polynomial. This polynomial is derived to get the step length that produces the best fitness of the three existing fitness. Next, a random step vector is generated around the step length, hoping it does not move away from the global optima point. In FPA, the random step vector is generated by a levy distribution whose movement is not controlled so that it is possible to move away from the global optima point.

To improve the accuracy and speed of FPNN convergence in disease identification in pigs, QIFPNN is proposed. QIFPNN performance was

also tested against other metaheuristic algorithms, namely BANN and PSONN. The BA and PSO algorithms used are based on [12]. As a result, QIFPNN outperformed FPNN, BANN, and PSONN in terms of accuracy. The convergence speed of QIFPNN exceeds FPNN, so the use of QI in QIFPNN can improve the performance of FPNN.

2. Theory

2.1 Population

A population-based metaheuristic algorithm means that each individual (solution) in the population acts as a solution candidate. In each generation (iteration), the algorithm will improve the population. The population represents the set of solution vectors in the search space. The population and solution vectors are expressed as Eq. (1) and Eq. (2).

$$X = \{x_1^t, x_2^t, \dots, x_n^t\} \tag{1}$$

$$x_i^t = (x_{i1}^t, x_{i2}^t, \dots, x_{id}^t) \tag{2}$$

where n is the population size or number of solutions, t and d respectively represent the iterations and the dimensions or number of solution variables. Each element of the solution vector can be expressed in terms of x_{ic}^t, where $i = 1,2,\dots,n$ and $c = 1,2,\dots,d$.

2.2 Flower pollination algorithm

FPA is inspired by the pollination process in flowering plants with the following rules [8]:

1. Biotic pollinators such as bees, insects, birds, and bats carry pollen and fly long distances resulting in cross-pollination.
2. Abiotic pollinators such as wind, water, and rain can only pollinate at close range so that self-pollination occurs.
3. The probability of reproduction or flower toughness is proportional to the similarity between two flowers.
4. Another parameter of the probability $p \epsilon rand[0,1]$ switch controls pollination in rules 1 and 2.

Assuming that one flower produces only one pollen, then the potential solution to a problem is the same as either flower or pollen. This algorithm is then formulated by utilizing the concept of cross-pollination and self-pollination. In cross-pollination, it is guaranteed that the best reproduction occurs in pollinators carrying pollen over long distances [13], so rules 1 and 3 can be presented as Eq. (3).

$$x_i^{t+1} = x_i^t + \gamma L\left(g^* - x_i^t\right) \tag{3}$$

where x_i^t is the pollen position vector or the i-th solution vector in iteration t, g^* is the best solution vector found among all the solutions in the current iteration, γ is the scaling factor that controls the step size, and L is the step vector or pollination power. Levy flight describes the step distance from the pollinator, where L is derived from the Levy's distribution using Eq. (4).

$$L \sim \frac{\lambda\Gamma(\lambda)sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0) \tag{4}$$

$\Gamma(\lambda)$ is the standard gamma function and $\lambda = 1.5$, the Levy distribution at large steps applies $s > 0$, and s is determined from the gaussian distribution U and V as in Eq. (5).

$$s = \frac{U}{|V|^{1/\lambda}}, \ U \sim N(0,\sigma^2), \ V \sim N(0,1) \tag{5}$$

where $U \sim N(0,\sigma^2)$ shows that a normal gaussian distribution with variance σ^2 and a mean of zero, and $V \sim N(0,1)$ indicates a standard gaussian normal distribution. σ^2 can be calculated by Eq. (6).

$$\sigma^2 = \left\{\frac{\Gamma(1+\lambda)}{\lambda\Gamma\left(\frac{1+\lambda}{2}\right)}\frac{\sin\left(\frac{\pi\lambda}{2}\right)}{2^{(\lambda-1)/2}}\right\}^{1/\lambda} \tag{6}$$

Self-pollination involves neighbouring pollen, with rules 2 and 3 being presented in Eq. (7).

$$x_i^{t+1} = x_i^t + \epsilon\left(x_j^t - x_k^t\right) \tag{7}$$

where random walk ϵ is a uniform distribution vector [0,1], x_j^t and x_k^t are pollen positions j and k.

Rule 4 aims to arrange pollination activity randomly to switch. A preliminary parametric study shows that $p = 0.8$ can work better for most applications [8, 14].

2.3 Newton polynomial quadratic interpolation

It has several $m+1$ data point information in the range of $[a,b]$, namely (r_0,f_0), (r_1,f_1), ..., (r_m,f_m) so to determine the $P(r)$ function that replaces the original function, interpolation can be used. $P(r)$ is a polynomial function as in Eq. (8).

$$P_m(r) = a_0 + a_1(r - r_0) + a_2(r - r_0)(r - r_1) + \cdots + a_m(r - r_0)(r - r_1) \ldots (r - r_{m-1}) \quad (8)$$

$a_m = [f0, \ldots, fm]$, is the dividend value recursively defined as Eq. (9).

$$[f_0] = f_0$$
$$[f_0, f_1] = \frac{f_1 - f_0}{r_1 - r_0} \quad (9)$$
$$[f_0, f_1, \ldots, f_m] = \frac{[f_1, \ldots, f_m] - [f_0, \ldots, f_{m-1}]}{r_m - r_0}$$

Quadratic interpolation is known as second-degree interpolation ($m = 2$) where three data points are required, the polynomial form can be written as Eq. (10).

$$P_2(r) = f_0 + [f_0, f_1](r - r_0) + [f_0, f_1, f_2](r - r_0)(r - r_1) \quad (10)$$

2.4 Quadratic interpolation flower pollination

In QIFP, modification of FPA is carried out in cross-pollination, where γL in Eq. (3) is replaced by the quadratic interpolation step vector, namely Q, so that the equation looks like Eq. (11).

$$x_i^{t+1} = x_i^t + Q\left(g^* - x_i^t\right) \quad (11)$$

The data points in the NPQI represent r as Q, and f as the fitness of an objective function. The determination of the three data points is carried out as follows:

1. First data point (r_0, f_0)
   Assuming $Q = 0$ in Eq. (11), the value $f_0 = f\_obj\left(x_i^t\right)$, and $r_0 = 0$.
2. Second data point (r_1, f_1)
   Assuming $Q = 1$ in Eq. (11), then the values for $f_1 = f\_obj(g^*)$, and $r_1 = 1$.
3. Third data point (r_2, f_2)
   Assuming $Q = 1.25$ in Eq. (11), then the value of $f_2 = f\_obj\left(x_i^{t+1}\right)$, and the value of $r_2 = 1.25$. The value of x_i^{t+1} is calculated using Eq. (12).

$$x_i^{t+1} = x_i^t + 1.25\left(g^* - x_i^t\right) \quad (12)$$

These three data points are then used to obtain the quadratic polynomial function $P_2(r)$. Since the optimization problem is a minimum or maximum problem, the value of r^* which gives the minimum or maximum fitness of the quadratic polynomial function is found using Eq. (13).

$$\frac{dP_2}{dr} = 0$$
$$r^* = \frac{r_1}{2} + \frac{r_0}{2} + \frac{\frac{f(r_1) - f(r_0)}{r_1 - r_0}}{2\left(\frac{\left(\frac{f(r_2) - f(r_1)}{r_2 - r_1}\right) - \left(\frac{f(r_1) - f(r_0)}{r_1 - r_0}\right)}{r_2 - r_0}\right)} \quad (13)$$

An illustration of the three data point representations for the minimum objective function can be seen in Fig. 1. To get the step vector Q at point r^*, this is done:

1. $Q \sim N(\mu, \sigma)$ is a normal distribution with mean $\mu = r^*$ and deviation standard $\sigma = f_1$.
2. So that the solution vector elements do not always change, some solution vector elements need to be maintained using the 20% step vector element Q is made zero using Eq. (14).

$$Q_c = \begin{cases} 0, & c = l \\ Q_c, & otherwise \end{cases} \quad (14)$$

where l is an integer random number $[1, d]$ generated as much as 20 % of d.

The search space defines the area the algorithm uses to find all possible solutions. The search space $[LbReal, UbReal]$ says $LbReal$ is the lower limit and $UbReal$ is the upper limit. In the optimization problem with a single objective function, the metaheuristic algorithm may not get satisfactory results in the original search space. The original search space is divided into smaller search subspaces [15, 16]. The search technique in this

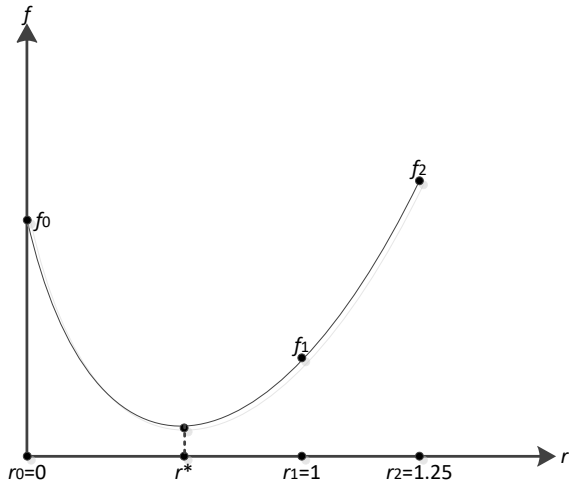

Figure. 1 An illustration of the representation of three data points for the minimum objective function.

study was that the search starts from a small search space and then slowly expands to the area of the original search space. Expanding the search space follows the steps as follows:

1. Identification of the initial search space:

$$[Lb, Ub]^d = [-1,1]^d \tag{15}$$

2. The next search space is expanded by:

$$[Lb, Ub]^d = \begin{cases} [Lb-1, Ub+1]^d & t \text{ MOD } 50 = 0 \\ [Lb, Ub]^d & \text{otherwise} \end{cases} \tag{16}$$

3. Step 2 is repeated until $Lb = LbReal$ and $Ub = UbReal$

Where $LbReal$ and $UbReal$ are integers, and the sum of $LbReal$ and $UbReal$ is zero. In each identified search space, a metaheuristic algorithm is used to find possible solutions.

2.5 Feed-forward neural network

Feed-forward neural network (FNN) is a classifier that carries out supervised forward learning in a pattern layer to obtain a classification model. FNN contains several neurons arranged in layers, neurons in each layer can be connected to neurons in the next layer, and each connection has a weight. FNN has at least one hidden layer that is between the input and output layers. Fig. 2 is an FNN with one hidden layer. The explanation of the three types of neurons in each layer is from Fig. 2, namely:

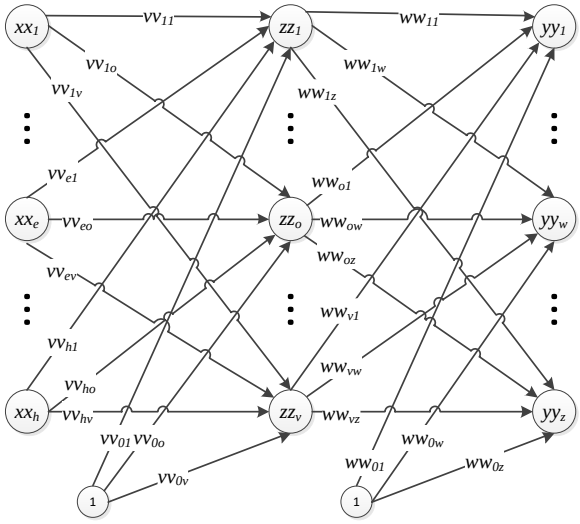

Figure. 2 FNN architecture.

- Input neurons xx_e: neurons receive input from the outside world. No computation is done, only passing information to hidden neurons. Where $e = 1,2,\dots h$.
- Hidden neuron zz_o : information from input neuron xx_e, connection weight vv_{eo} and bias vv_{0o} are computed to produce output on hidden neuron zz_o which is then forwarded to the output neuron, where $o = 1,2,\dots v$. The calculation for hidden neurons can be seen in Eq. (17).

$$zz_o = \frac{1}{\left(1+\exp\left(-\left(vv_{0o}+\sum_e^h xx_e vv_{eo}\right)\right)\right)} \tag{17}$$

- Output neuron yy_w : output from the hidden neuron zz_o, connection weights ww_{ow} and bias ww_{0w} are computed to produce output on the output neuron, where $w = 1,2,\dots z$. The calculation on the output neuron can be seen in Eq. (18).

$$yy_w = \frac{1}{\left(1+\exp\left(-\left(ww_{0w}+\sum_w^z zz_o ww_{ow}\right)\right)\right)} \tag{18}$$

Learning error is determined by Eq. (19).

$$MSE = \frac{\sum_{y=1}^{Ndt}\left(\frac{\sum_{w=1}^{z}\left(dt_{wy}-yy_{wy}\right)^2}{z}\right)}{Ndt} \tag{19}$$

where MSE (mean square error) is the average squared error, Ndt is the amount of training data, z is the number of neurons in the output layer, dt_{wy} is the target of the y-th actual data on the w-element, and yy_{wy} is the output of the w -th neuron corresponding to the actual data-target y.

The calculation of classification accuracy using Eq. (20) is done by:

- Use weights and biases for instances.
- Perform forward propagation to get the output, compare the output with the actual class, and calculate the correct class.

$$classification\ accuracy = \frac{total\ number\ of\ correct\ classification}{total\ number\ of\ instances} \quad (20)$$

2.6 *k*-Fold cross-validation

k-fold cross-validation is used to evaluate and select a good classification model. *k*-fold-cross-validation means that the dataset is divided into k parts, and cross-validation is carried out for k rounds so that each part has the opportunity to be tested [17]. Of the k parts, $k-1$ parts are used as training data, and one part as validation data. In k cycles it will produce k different exact models.

2.7 Quadratic interpolation flower pollination neural network with k-fold cross-validation

The weights and bias of the FNN are represented in pollen, as in Fig. 3.

The objective function is the FNN function, as in Eq. (21).

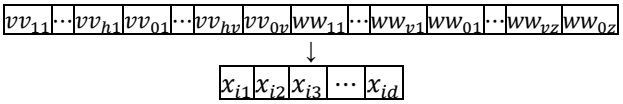

Figure. 3 Arrangement of the weight and bias elements in the solution vector.

$$f\_obj(x) = MSE(x) \quad (21)$$

Each pollen in the population produces its fitness. The set of fitness in the t-iteration is expressed as Eq. (22).

QIFPNN algorithm:

Objective minimize $f\_obj(x), x = (x_1, x_2, \dots, x_d)$
Initialize QIFPNN control parameters
*[number of flowers/pollen gametes (**n**), max iteration (**maxIteration**), switch probability (**p**), problem dimensions (**d**), lower bound (**Lb**), upper bound (**Ub**), target error (**targetError**), NN structure, iteration threshold of convergence accuracy (**iterThOfCA**), lower bound real (**LbReal**), upper bound real (**UbReal**)]*
Load the diseases of pig training data
Load the diseases of pig validation data
$\boldsymbol{t} = 1$
*Initialize a population of **n** flowers/pollen gametes with random solutions* $[0,1]$
Find the candidate best solution $\boldsymbol{g}*$ *in the initial population based on the fitness of training data*
Using $\boldsymbol{g}^*$ *to compute* $\boldsymbol{globalBestAccuracy}$ *of validation data*
$\boldsymbol{countIterOfConvAccuracy} = 1$
While (t $\leq$ ***maxIteration) and** (fitness of* $\boldsymbol{g}^* \geq$ ***targetError) and** (**countIterOfConvAccuracy** $\leq$ ***iterThOfCA***)
    For i** = 1 : **n
        If $\left|fitness\ of\ \boldsymbol{x_i^t} - fitness\ of\ \boldsymbol{g}^*\right| < 10^{-2}$ then $\boldsymbol{temp}(i) = "1"$
        If $sum\left(count(\boldsymbol{temp}, "1")\right) \geq 90\%$ *of population*
            Generate the population via $x^t = Lb + rand[0,1](Ub - Lb)$ *except the solution vector containing* $\boldsymbol{g}^*$
            Empty $\boldsymbol{temp}$
        End if
        If** rand < **p
            ***If** i ≠ CandidateBestSolutionIndex*
                Draw a (d-dimensional) step vector Q based on quadratic interpolation mechanism
                Global pollination via Eq. (11)
            End if
        Else
            ***If** i ≠ CandidateBestSolutionIndex*
                Draw $\boldsymbol{\epsilon}$ *from a uniform distribution in* $[0,1]$
                Do local pollination via Eq. (7)

End if
End if
Check the bounds, $x_{ic}^{t+1} = \begin{cases} Lb & x_{ic}^{t+1} < Lb \\ Ub & x_{ic}^{t+1} > Ub \end{cases}$
Using new solution $\boldsymbol{x_i^{t+1}}$ *to compute fitness of training data*
If the fitness of new solution < fitness of old solution then $old\ solution = new\ solution$
If the fitness of new solution < fitness of $\boldsymbol{g}^*$ *then update* $\boldsymbol{g}^*$
End for
Compute classification accuracy of validation data ($\boldsymbol{accuracyOfValData}$) *using* $\boldsymbol{g}^*$ *via Eq. (20)*
If ($\boldsymbol{accuracyOfValData \geq globalBestAccuracy}$)
Update current best solution $\boldsymbol{bs = g}^*$
globalBestAccuracy = ***accuracyOfValData***
countIterOfConvAccuracy = *0*
End if
Expand of ***Lb*** *and* ***Ub*** *coverage using Eq. (16)*
countIterOfConvAccuracy = ***countIterOfConvAccuracy*** + *1*
$\boldsymbol{t = t + 1}$
End while
Output the best solution $\boldsymbol{bs}$ *found*

$$fitness^t = \{MSE(x_1^t), MSE(x_2^t), \dots, MSE(x_n^t)\} \quad (22)$$

The optimization problem in FNN is to minimize learning errors.

Training and validation data are used in the QIFPNN training process with the aim that the solution obtained is able to provide the best classification accuracy. This accuracy is obtained in two stages. First, find the solution that provides the best accuracy on the training data in each iteration indicated by the fitness value or minimum MSE of the solution. This solution is called the best candidate solution (g^*). Second, check the accuracy of the validation data ($accuracyOfValData$) of g^*. If $accuracyOfValData$ is greater than or equal to the best accuracy ($globalBestAccuracy$), then update the best solution (bs) and $globalBestAccuracy$, and reset the accuracy convergence counter($countIterOfConvAccuracy$). During iteration, the value of $countIterOfConvAccuracy$ is incremented. If $countIterOfConvAccuracy$ has reached the convergence accuracy ($iterThOfCA$) limit value, the training is stopped.

2.8 Pig diseases data

The disease dataset in pigs was obtained from the national animal health information system (iSIKHNAS) [18]. The data collected consisted of disease data and accompanying clinical symptoms. There were 11 diseases and 68 clinical symptoms, according to data availability. The eleven diseases consist of worm infections, dystocia, endometritis, myiasis, mastitis, pneumonia, eye inflammation, retensio secundinarum, scabies, streptococcosis, and septichaemia epizootica. sixty-eight clinical symptoms consist of: anorexia, limp, swollen joints, anemia, dull hair, thinness, weakness, lack of response, postpartum, diarrhea, fever, skin inflammation, parasitic feces, lack of body fluids, standing abnormal, injuries feet, paralyzed, difficulty in standing, cough, runny nose, abnormalities/difficulty breathing, pale mucosa, vaginal inflammation, itching, loss of hair, collapsing, abscess, bleeding sores, skin disorders, vulva abnormalities, inflammation of vulva, vomiting, abnormal walk, unsteadiness, constipation, nose with mucus, vulvar pus discharge, inflammation of the eyes, ear scabs, ulcers, temperature disorders, snoring, (jaw) / (lower jaw) swelling, respiratory system disorders, green feces, abdominal abnormalities, purulent eyes, difficulty birth , bad smell vulva, miscarriage, vaginal abnormalities, stillbirth, excessive saliva, tremor, swollen udder, inflammation of nipples, blisters on udder, Scrotum Hernia, Swollen Testicles, Nose problems, blindness, light sensitive eyes, placental retention, cloudy eye corneas, swollen head, milk disorders, difficulty chewing/swallowing, and hypothermia.

The disease dataset in pigs was categorical, so it needs to be converted into numeric. The value of clinical symptoms that appear in specific diseases is set to 1 while others are 0. Sixty-eight clinical symptoms become xx_e input neurons and 11 diseases were converted in binary to produce four

output neurons yy_w. The dataset of diseases in pigs can be categorized into medical datasets. It had 68 input attributes and 11 class attributes. The input attribute had binary characteristics. Class characteristics can be seen in Table 1.

Table 1. Class characteristics

Class	Information
Worm infections	17 data (11%)
Dystocia	3 data (2%)
Endometritis	10 data (6%)
Myiasis	10 data (6%)
Mastitis	6 data (4%)
Pneumonia	21 data (13%)
Eye Inflammation	4 data (3%)
Retensio Secundinarum	3 data (2%)
Scabies	22 data (14%)
Streptococcosis	44 data (28%)
Septichaemia Epizootica	18 data (11%)
Total	**158 data (100%)**

3. Experiment setup

3.1 System configuration

The QIFPNN algorithm proposed in this study was used to identify diseases in pigs. QIFPNN was coded and implemented using MATLAB R2018b on a computer with an Intel Core i7-9750H configuration, CPU @ 2.6GHz - Up to 4.6GHz, 16GB RAM, 64-bit operating system. All experiments were carried out on the specified machine. The training and testing process was based on a parallel computing model, which took advantage of the six physical cores available on the Intel Core i7-9750H.

3.2 Parameter settings

The optimal solution obtained by a metaheuristic algorithm was to control the parameters in the iteration cycle. However, the exact way to create parameter diversity was unknown [19]. QIFPNN, FPNN, BANN, and PSONN were vulnerable to the parameters used, similar to other metaheuristic algorithms, which were also susceptible to their parameters. In this experiment, the population parameter $n = 30$ [20] and the original search space for all dimensions $LbReal = -80$ and $UbReal = 80$ were determined. For FPNN, BANN and PSONN, the search space is directly performed against the original search space so that $Lb = LbReal = -80$ and $Ub = UbReal = 80$. Especially for QIFPNN and FPNN, the switch probability parameter is $p = 0.8$. In BANN, parameter $loudness = 0.25$ and parameter $pulseRate = 0{,}5$, while in PSONN, learning parameters $\alpha \approx \beta \approx 2$ [12].

In FNN, a hidden layer was used, given the fact that one hidden layer is relatively close to function with arbitrary accuracy [21]. The determination of the number of hidden neurons, according to [22], uses Eq. (23).

$$v = \sqrt{(h + z)} + aa \tag{23}$$

where v is the number of neurons in the hidden layer, h is the number of neurons in the input layer, z is the number of neurons in the output layer, and aa is an integer. By choosing $aa = 1$, we get $v = 9$ so that the FNN architecture used is 68-9-4.

The specified stopping parameters for the QIFPNN, FPNN, BANN and PSONN criteria were the maximum iteration ($maxIteration = 4000$), the target error ($targetError = 0.001$) and the iteration allowance limit for the convergence condition of classification accuracy ($iterThOfCA = 700$).

3.3 Comparison algorithms and running the proposed QIFPNN

The disease dataset in pigs is divided into three subsets, i.e. 90 % of the data for the training and validation subsets are based on 10-fold cross-validation, and 10 % of the data for the test subset. The test records the minimum (Min), maximum (Max), and average (Ave) values of the classification accuracy values for QIFPNN, FPNN, BANN, and PSONN.

4. Result and discussion

Finding optimal solutions resistant to uncertainty in real-world problems is very difficult, so an algorithm must be treated on various data groups. *k*-fold cross-validation is very helpful in measuring the performance of algorithms in different data groups. The algorithm performance measurement used is the minimum, maximum, and average classification accuracy per fold.

The performance comparison results of QIFPNN, FPNN, BANN, and PSONN are presented in Table 2-6 and Fig. 4-11. Table 2 and Fig. 4-5 represent information on a mean of 20 runs of classification accuracy per fold and an average of 10 folds of the training subset. Table 3 and Fig. 6-7 represent information on a mean of 20 runs of classification accuracy per fold and an average of 10 folds of the validation subset.

Table 4 and Fig. 8-9 represent information on a mean of 20 runs of classification accuracy per fold and a mean of 10 folds of the test subset. Table 5 describes the information recapitulation of the average classification accuracy of the training, validation, and test subsets for ten folds. In comparison, Table 6 and Fig. 10-11 represent the average information of 20 runs of training time per fold and an average of 10 folds of training time.

Based on the classification accuracy measurement results on the training, validation, and test subsets between QIFPNN, FPNN, BANN, and PSONN, it can be seen that QIFPNN provides promising results in experiments than FPNN, BANN, and PSONN. The average value of classification accuracy in training, validation, and test subsets for each fold obtained by QIFPNN is higher than FPNN, BANN, and PSONN. Likewise,

Table 2. Average 20 runs of classification accuracy per fold in the training subset of disease in pigs

Fold	Method											
	QIFPNN (%)			FPNN (%)			BANN (%)			PSONN (%)		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
1	89.0625	99.2188	**96.4453**	40.625	92.9688	82.8906	8.5938	100	68.6328	10.9375	32.8125	21.2891
2	90.5512	99.2126	**96.5354**	22.8347	92.126	68.2284	7.0866	98.4252	72.2047	13.3858	40.1575	24.9606
3	87.4016	99.2126	**95.63**	52.7559	95.2756	79.4095	7.0866	99.2126	72.126	11.0236	31.4961	20.7874
4	75	99.2188	**91.4453**	56.25	92.1875	74.5313	8.5938	98.4375	71.7969	5.4688	39.8438	21.1328
5	85.9375	99.2188	**95.8984**	40.625	90.625	69.5313	14.0625	100	76.25	7.0313	38.2813	22.0703
6	75	99.2188	**94.6094**	46.875	89.8438	69.2578	13.2813	98.4375	75.3125	12.5	36.7188	23.0859
7	89.8438	99.2188	**95.625**	32.0313	92.1875	75.1953	9.375	99.2188	78.9844	4.6875	31.25	18.4375
8	85.1563	98.4375	**95.1172**	33.5938	90.625	65.7422	5.4688	99.2188	70.5469	8.5938	35.1563	19.9609
9	90.625	99.2188	**97.5**	50	93.75	76.1719	5.4688	100	71.1719	10.1563	36.7188	22.1094
10	76.5625	100	**93.6328**	59.375	94.5313	79.1797	10.9375	99.2188	79.1406	14.0625	38.2813	21.6016
Mean	84.514	99.2175	**95.2439**	43.4966	92.412	74.0138	8.9955	99.2169	73.6167	9.7847	36.0716	21.5436

Table 3. Average 20 runs of classification accuracy per fold in the validation subset of disease in pigs

Fold	Method											
	QIFPNN (%)			FPNN (%)			BANN (%)			PSONN (%)		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
1	64.2857	92.8571	**72.8571**	57.1429	85.7143	67.8571	7.1429	78.5714	46.4286	7.1429	42.8571	26.7857
2	80	93.3333	**84.3333**	40	86.6667	68.6667	13.3333	86.6667	56.3333	13.3333	40	30.3333
3	60	86.6667	**73**	60	80	69.3333	6.6667	73.3333	51.6667	20	46.6667	25.3333
4	57.1429	92.8571	**79.2857**	57.1429	85.7143	72.1429	14.2857	92.8571	58.5714	7.1429	50	26.7857
5	64.2857	85.7143	**76.0714**	50	78.5714	66.4286	21.4286	85.7143	58.5714	7.1429	42.8571	26.4286
6	71.4286	92.8571	**79.2857**	50	92.8571	69.6429	0	85.7143	57.1429	14.2857	35.7143	26.0714
7	71.4286	92.8571	**83.9286**	57.1429	85.7143	74.6429	7.1429	85.7143	63.9286	14.2857	42.8571	27.8571
8	71.4286	92.8571	**81.0714**	50	78.5714	63.5714	7.1429	85.7143	55.3571	7.1429	42.8571	25
9	71.4286	85.7143	**79.2857**	64.2857	92.8571	72.1429	21.4286	85.7143	61.0714	14.2857	50	29.6429
10	71.4286	78.5714	**75.3571**	64.2857	78.5714	72.8571	7.1429	78.5714	57.1429	7.1429	35.7143	26.7857
Mean	68.2857	89.4286	**78.4476**	55	84.5238	69.7286	10.5714	83.8571	56.6214	11.1905	42.9524	27.1024

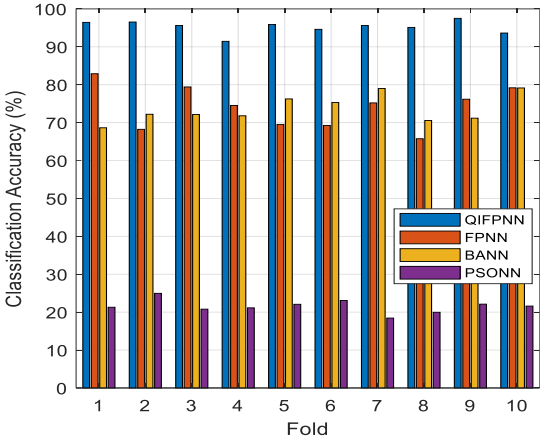

Figure. 4 Average 20 runs per fold of classification accuracy of the training subset of diseases in pigs

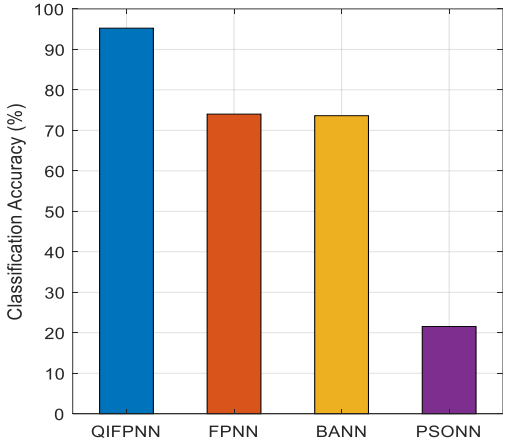

Figure. 5 Average 10 fold of classification accuracy of the training subset of diseases in pigs

Table 4. Average 20 runs of classification accuracy per fold in the test subset of disease in pigs

Fold	Method											
	QIFPNN (%)			FPNN (%)			BANN (%)			PSONN (%)		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
1	56.25	87.5	**75.3125**	37.5	81.25	63.75	0	87.5	41.875	0	43.75	20
2	62.5	87.5	**77.8125**	25	81.25	56.5625	6.25	75	51.875	12.5	50	27.5
3	62.5	93.75	**78.4375**	31.25	81.25	61.25	6.25	81.25	51.5625	6.25	37.5	22.1875
4	56.25	81.25	**68.125**	31.25	75	56.875	6.25	75	48.4375	0	50	21.875
5	68.75	87.5	**77.8125**	25	75	54.375	6.25	81.25	56.5625	6.25	37.5	21.875
6	62.5	87.5	**72.8125**	31.25	75	53.4375	6.25	87.5	54.375	6.25	37.5	22.5
7	62.5	87.5	**75.3125**	37.5	81.25	64.375	6.25	81.25	53.4375	0	37.5	19.375
8	50	87.5	**76.5625**	31.25	75	51.875	0	87.5	44.6875	6.25	43.75	22.1875
9	56.25	87.5	**73.75**	31.25	93.75	63.75	6.25	81.25	51.875	0	37.5	19.375
10	50	81.25	**65.625**	31.25	75	60.625	6.25	68.75	48.75	0	37.5	20.9375
Mean	58.75	86.875	**74.1563**	31.25	79.375	58.6875	5	80.625	50.3438	3.75	41.25	21.7813

Table 5. Recapitulation of mean classification accuracy of the training, validation, and test subsets for the ten folds of disease in pigs

Method	Average accuracy of training subset (%)	Average accuracy of validation subset (%)	Average accuracy of test subset (%)	Mean accuracy of training, validation, and test subsets (%)
QIFPNN	95.2439	78.4476	74.1563	**82.6159**
FPNN	74.0138	69.7286	58.6875	67.4766
BANN	73.6167	56.6214	50.3438	60.194
PSONN	21.5436	27.1024	21.7813	23.4758

Table 6. Average 20 runs of training time per fold

Fold	Method			
	QIFPNN (seconds)	FPNN (seconds)	BANN (seconds)	PSONN (seconds)
1	6122.22	7850.327	5304.251	5069.914
2	5568.101	5523.534	5917.551	3812.06
3	5808.79	7062.422	4680.401	4273.359
4	4839.996	6728.211	5473.157	3645.717
5	7166.274	6054.811	5709.082	4095.621
6	6800.633	5656.272	6231.315	4774.752
7	5516.892	7134.534	6109.82	4029.035
8	5872.615	5550.709	6364.938	3690.586
9	7512.588	7136.171	5075.679	3193.189
10	5354.292	6854.8	4879.294	3488.122
Mean	6056.240	6555.179	5574.549	**4007.235**

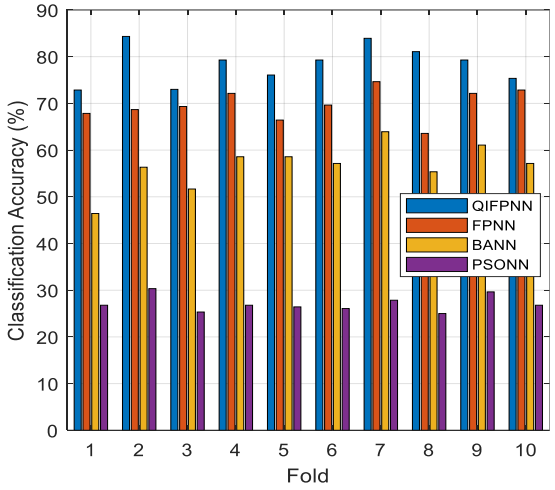

Figure. 6 Average 20 runs per fold of classification accuracy of the validation subset of diseases in pigs

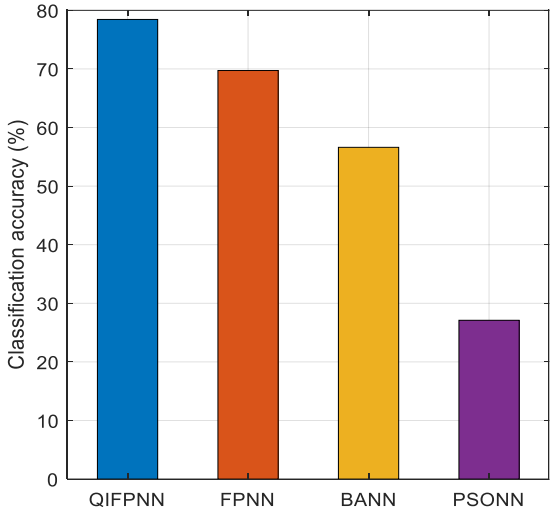

Figure. 7 Average 10 fold of classification accuracy of the validation subset of diseases in pigs

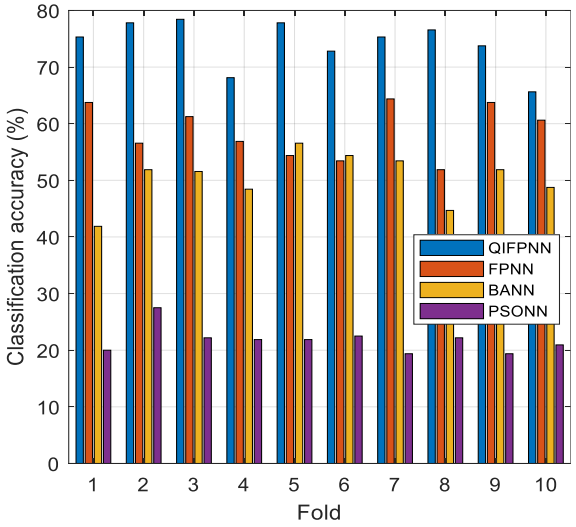

Figure. 8 Average 20 runs per fold of classification accuracy of the test subset of diseases in pigs

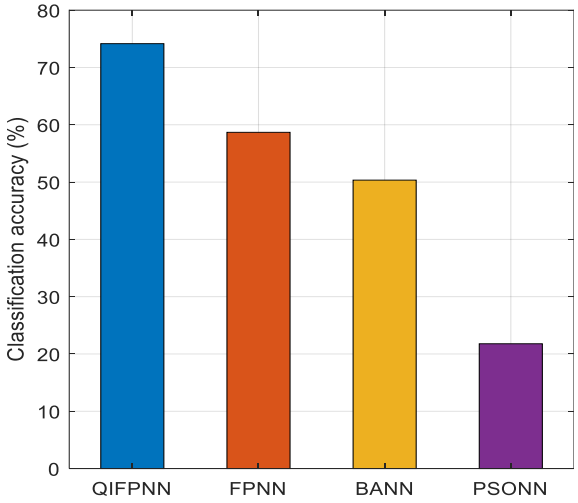

Figure. 9 Average 10 fold of classification accuracy of the test subset of diseases in pigs

the average value of classification accuracy for ten folds shows the QIFPNN 82.6159 % higher than FPNN 67.4766 %, BANN 60.194 %, and PSONN 23.4758 %. The measurement of training time per fold QIFPNN 6056.240 seconds is relatively faster than FPNN 6555.179 seconds.

QIFPNN accuracy increased by 22.40 %, and training time was 7.61 % faster against FPNN. However, the QIFPNN training time is slower than BANN and PSONN.

QIFPNN was applied to identify disease in pigs based on data on clinical symptoms of pigs that were tested to see the identification ability of QIFPNN. Disease identification in pigs by FPNN outperformed the accuracy of BANN and PSONN but with a slower convergence rate.

As expected, the proposed QIFPNN can improve the accuracy and speed of convergence of the FPNN. This shows that QIFPNN managed to avoid local

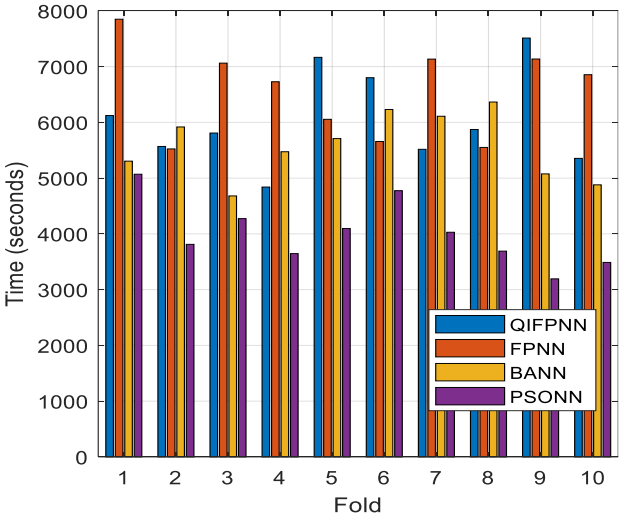

Figure. 10 Average of 20 runs per fold of training time

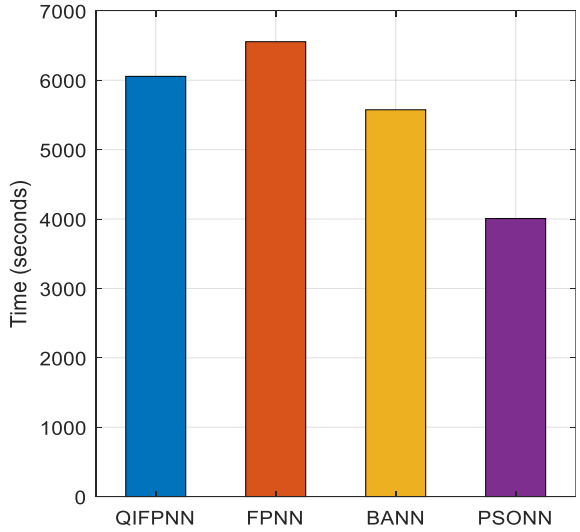

Figure. 11 Average of 10 folds of training time

minima and has the ability to explore the search space to find the best weights and biases that result in minimum MSE training, thereby increasing classification accuracy. The QI concept adopted by QIFPNN can accelerate convergence. The poor classification accuracy of FPNN, BANN, and PSONN is caused by poor exploration and exploitation capabilities.

Overall, QIFPNN is accurate and promises to solve real-life optimization problems. The proposed QIFPNN algorithm can be used as a complementary model for human expertise, provided that data on clinical symptoms of disease in pigs is well available.

5. Conclusion

In this study, the QI approach was used to improve accuracy and accelerate the convergence of QIFPNN in disease identification in pigs. It is a complementary model that can be used to identify diseases in pigs. Updating the weights in NN training using metaheuristic algorithms such as FPA, BA, and PSO compared to QIFP aims to obtain an appropriate disease identification model in pigs. It was found that the proposed QIFPNN performs better than FPNN, BANN, and PSONN in terms of accuracy, and the convergence speed of QIFPNN is faster than FPNN but slower than BANN and PSONN. This is evidenced by the average value of classification accuracy for ten fold, showing QIFPNN 82.6159 % higher than FPNN 67.4766 %, BANN 60.194 %, and PSONN 23.4758 %. The measurement of training time per fold QIFPNN 6056.24 seconds is relatively faster than FPNN 6555.179 seconds. QIFPNN accuracy increased by 22.4 %, and training time was 7.61 % faster against FPNN. In addition, QIFP can be used as an alternative method of NN training. In the future, we recommend the QIFPNN step vector using the Cauchy distribution and testing it on various datasets with various data types.

Conflicts of interest

The authors declare there are no conflict interest in this work.

Author contributions

The contributions of each author are described as follows: "Conceptualization and methodology, Hartati, Suprapto, and Polly; implementation of methodology, Polly; data validation, Sumiarto and Polly; formal analysis, Hartati, Suprapto, and Sumiarto; investigation, Hartati, Suprapto, Sumiarto, and Polly; writing—original draft preparation, Polly; writing—review and editing, Hartati, Suprapto and Sumiarto; supervision, Hartati, Suprapto and Sumiarto; funding acquisition, Polly".

Acknowledgments

The authors would like to thank the LPDP (lembaga pengelola dana pendidikan) for funding this research. The author also would thank The directorate general of livestock and animal health services, ministry of agriculture, Indonesia, for supporting this research in providing data via iSIKHNAS.

References

[1] BPS NTT, *Profil Sektor Pertanian Provinsi Nusa Tenggara Timur 2019*. Kupang: © Badan Pusat Statistik Provinsi Nusa Tenggara Timur, 2020.

[2] J. Shao, “Classification of swine thermal comfort behavior by image processing and neural network”, *Paper 12033* (*Iowa State University*), 1997.

[3] Y. Wang, W. Yang, P. Winter, and L. Walker, “Walk-through weighing of pigs using machine vision and an artificial neural network”, *Biosyst. Eng.*, Vol. 100, No. 1, pp. 117–125, 2008, doi: 10.1016/j.biosystemseng.2007.08.008.

[4] A. Apichottanakul, S. Pathumnakul, and K. Piewthongngam, “The role of pig size prediction in supply chain planning”, *Biosyst. Eng.*, Vol. 113, No. 3, pp. 298–307, 2012, doi: 10.1016/j.biosystemseng.2012.07.008.

[5] S. K. Biswas, B. Baruah, B. Purkayastha, and M. Chakraborty, “An ANN based Classification Algorithm for Swine Flu Diagnosis”, *Int. J. Knowl. Based Comput. Syst.*, Vol. 3, No. 1,

2015, doi: 10.21863/ijkbcs/2015.3.1.005.
[6] M. Mavrovouniotis and S. Yang, "Training neural networks with ant colony optimization algorithms for pattern classification", *Soft Comput.*, Vol. 19, No. 6, pp. 1511–1522, 2015, doi: 10.1007/s00500-014-1334-5.
[7] V. K. Ojha, A. Abraham, and V. Snášel, "Metaheuristic design of feedforward neural networks: A review of two decades of research", *Eng. Appl. Artif. Intell.*, Vol. 60, No. January, pp. 97–116, Apr. 2017, doi: 10.1016/j.engappai.2017.01.013.
[8] X. S. Yang, "Flower Pollination Algorithm for Global Optimization", *Unconventional Computation and Natural Computation*, Vol. 7445, 2012, pp. 240–249.
[9] Y. Yang, X. Zong, D. Yao, and S. Li, "Improved Alopex-based evolutionary algorithm (AEA) by quadratic interpolation and its application to kinetic parameter estimations", *Appl. Soft Comput.*, Vol. 51, pp. 23–38, Feb. 2017, doi: 10.1016/j.asoc.2016.11.037.
[10] K. Deep and J. C. Bansal, "Hybridization of particle swarm optimization with quadratic approximation", *OPSEARCH*, Vol. 46, No. 1, pp. 3–24, Mar. 2009, doi: 10.1007/s12597-009-0002-5.
[11] D. Singh and S. Agrawal, "Self organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems", *Appl. Soft Comput.*, Vol. 38, pp. 1040–1048, Jan. 2016, doi: 10.1016/j.asoc.2015.09.033.
[12] X. S. Yang, *Nature-Inspired Optimization Algorithms*, Elsevier Inc, 2014.
[13] E. Nabil, "A Modified Flower Pollination Algorithm for Global Optimization", *Expert Syst. Appl.*, Vol. 57, pp. 192–203, Sep. 2016, doi: 10.1016/j.eswa.2016.03.047.
[14] X. S. Yang, "Chapter 11 - Flower Pollination Algorithms", *Nature-Inspired Optimization Algorithms*, 2014, pp. 155–173.
[15] M. Rajora, P. Zou, Y. Guang, Z. W. Fan, H. Y. Chen, W. C. Wu, B. Li, and S. Y. Liang, "A split-optimization approach for obtaining multiple solutions in single-objective process parameter optimization", *Springerplus*, Vol. 5, No. 1, p. 1424, Dec. 2016, doi: 10.1186/s40064-016-3092-6.
[16] Q. Fan, X. Yan, Y. Zhang, and C. Zhu, "A Variable Search Space Strategy Based on Sequential Trust Region Determination Technique", *IEEE Trans. Cybern.*, pp. 1–13, 2019, doi: 10.1109/TCYB.2019.2914060.
[17] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning", *arXiv*, Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.12808.
[18] iSIKHNAS, "Informasi Penyakit pada Babi dari iSIKHNAS", 2019. https://www.isikhnas.com/.
[19] X. S. Yang, *Cuckoo Search and Firefly Algorithm*, Vol. 516. Cham: Springer International Publishing, 2014.
[20] D. Chakraborty, S. Saha, and S. Maity, "Training feedforward neural networks using hybrid flower pollination-gravitational search algorithm", In: *Proc. of 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, Feb. 2015, pp. 261–266, doi: 10.1109/ABLAZE.2015.7155008.
[21] H. Chiroma, A. Khan, A. I. Abubakar, Y. Saadi, M. F. Hamza, L. Shuib, A. Y. Gital, and T. Herawan, "A new approach for forecasting OPEC petroleum consumption based on neural network train by using flower pollination algorithm", *Appl. Soft Comput.*, Vol. 48, pp. 50–58, Nov. 2016, doi: 10.1016/j.asoc.2016.06.038.
[22] H. Lin, Z. Chen, L. Wu, P. Lin, and S. Cheng, "On-line Monitoring and Fault Diagnosis of PV Array Based on BP Neural Network Optimized by Genetic Algorithm", *Multi-disciplinary Trends in Artificial Intelligence*, Vol. 9426, B. A and Z. X, Eds. Switzerland: Springer, Cham, 2015, pp. 102–112.