

All



ADVANCED SEARCH

Conferences > TENCON 2015 - 2015 IEEE Region 10 Conference

Implementation of an artificial neural network in recognizing in-flight quadrotor images

Publisher: IEEE

Cite This



Reiichiro Christian S. Nakano; Argel Bandala; Gerard Ely Faelden; Jose Martin Maiiiiigo; Elmer P. Dadios

All Authors

4

Cites in
Papers

261

Full
Text Views

Need Full-Text

access to IEEE Xplore
for your organization?

CONTACT IEEE TO SUBSCRIBE >

Abstract

Document
Sections

- I. Introduction
- II. Artificial Neural Network
- III. Architecture of the System
- IV. Experiment Results
- V. Discussion and Analysis of Results

Show Full Outline ▾

Authors

Figures

References

Citations

Keywords

Metrics

Abstract:

This paper shows an implementation of a feedforward artificial neural network capable of recognizing images of the CrazyFlie 2.0 quadrotor during flight. The network is to be used in a real-time quadrotor swarming application and has to be able to successfully differentiate pictures that show a quadrotor in flight versus pictures that do not. The network was trained using a standard backpropagation algorithm and images taken from a video of the said quadrotor in flight. These images were divided into three groups: a training set and validation set for the training stage, and a testing set for verification of the trained neural network. The results showed that the neural network was able to correctly identify the images in the testing phase 100 percent of the time while achieving a 94 percent accuracy for the images in the testing set.

Published in: TENCON 2015 - 2015 IEEE Region 10 Conference

Date of Conference: 01-04 November 2015

DOI: 10.1109/TENCON.2015.7372944

Publisher: IEEE

Date Added to IEEE Xplore: 07 January 2016

Conference Location: Macao, China

▼ ISBN Information:

▼ ISSN Information:

I. Introduction

Swarm robotics deals with swarming behaviors, such as that of ants gathering food or bees constructing beehives, and its applications to robotics [1] [2]. One of the defining characteristics of swarm behavior is that each member of the swarm can only sense the actions of its neighbors. No member of the swarm has access to information such as the total size of the swarm, location of every member in the swarm, or the location of its neighbors. The members follow a simple set of rules governing its behavior. From these local interactions between members, there is an emergent effect on the entire swarm and it is able to accomplish a complex task that no single

Sign in to Continue Reading

More Like This

The Influence of Gray Scale on
Image Recognition Under
Simulated Prosthetic Vision

2023 International Conference on
Communications, Computing,
Cybersecurity, and Informatics (CCCI)
Published: 2023

Locomotion Control and Gait
Planning of a Novel Hexapod
Robot Using Biomimetic Neurons

IEEE Transactions on Control Systems
Technology
Published: 2018

Show More

Authors	▼
Figures	▼
References	▼
Citations	▼
Keywords	▼
Metrics	▼

IEEE Personal Account

CHANGE USERNAME/PASSWORD

Purchase Details

PAYMENT OPTIONS

VIEW PURCHASED DOCUMENTS

Profile Information

COMMUNICATIONS PREFERENCES

PROFESSION AND EDUCATION

TECHNICAL INTERESTS

Need Help?

US & CANADA: +1 800 678 4333

WORLDWIDE: +1 732 981 0060

CONTACT & SUPPORT

Follow

f

@

in

▶

×

About IEEE *Xplore* | Contact Us | Help | Accessibility | Terms of Use | Nondiscrimination Policy | IEEE Ethics Reporting  | Sitemap | IEEE Privacy Policy

A public charity, IEEE is the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.



© Copyright 2026 IEEE - All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies.

Implementation of an Artificial Neural Network in Recognizing In-flight Quadrotor Images

Reiichiro Christian S. Nakano, Argel Bandala, Gerard Ely Faelden, Jose Martin Maningo, Elmer P. Dadios
Electronics and Communications Engineering Department, Manufacturing Engineering and Management Engineering Department
De La Salle University, Manila, Philippines
reii_nakano@yahoo.com, elmer.dadios@dlsu.edu.ph

Abstract— This paper shows an implementation of a feedforward artificial neural network capable of recognizing images of the CrazyFlie 2.0 quadrotor during flight. The network is to be used in a real-time quadrotor swarming application and has to be able to successfully differentiate pictures that show a quadrotor in flight versus pictures that do not. The network was trained using a standard backpropagation algorithm and images taken from a video of the said quadrotor in flight. These images were divided into three groups: a training set and validation set for the training stage, and a testing set for verification of the trained neural network. The results showed that the neural network was able to correctly identify the images in the testing phase 100 percent of the time while achieving a 94 percent accuracy for the images in the testing set.

Index Terms—neural network, quadrotor, CrazyFlie, swarm robotics

I. INTRODUCTION

Swarm robotics deals with swarming behaviors, such as that of ants gathering food or bees constructing beehives, and its applications to robotics [1] [2]. One of the defining characteristics of swarm behavior is that each member of the swarm can only sense the actions of its neighbors. No member of the swarm has access to global information about the swarm e.g. total size of the swarm, location of every member in the swarm [3]. Using this data from its neighbors, the members follow a simple set of rules governing its behavior. From these local interactions between members, there is an emergent effect on the entire swarm and it is able to accomplish a complex task that no single member can do on its own [3].

Quadrotors are a type of unmanned aerial vehicle (UAV) that fall under the planar vertical takeoff and landing type (PVTOL). This means that unlike a fixed wing aircraft, it can take off and land in a limited space, hover, and move in all directions [4]. In order to apply swarm behavior to a group of quadrotors, there needs to exist a method for each quadrotor to identify its neighbors i.e. to gather local data for determining its own behavior.

In this paper, an artificial neural network is proposed to serve as this method for local sensing. Artificial neural networks are well-suited for pattern recognition in noisy environments. In fact, neural networks are widely used for image and facial recognition. Thus, a neural network may be used to recognize whether or not a particular object is a fellow member of the swarm i.e. another quadrotor. The neural network is designed to recognize images of the CrazyFlie 2.0 quadrotor during flight.

The complete discussion of the system is arranged as follows: the first part will give a brief overview of artificial neural networks and their use in image recognition. The second part will discuss and present the architecture of the network proposed. The third part will discuss in detail each component

of the neural network: from preprocessing of images to the training and testing of the network. The fourth part will show the results of the experiment. Finally, the last part will provide a conclusion and discussion of the results.

II. ARTIFICIAL NEURAL NETWORK

Artificial neural networks are a form of soft computing that is loosely based on how the human brain works. Inspired by biological learning, a neural network “learns” based on adjustment of the weights on the connections between nodes. This is analogous to the connections between synapses and nodes in the human brain [5] [6].

The user of the neural network need not know the underlying mathematics behind the various weights and biases of the neural network. All one has to do is feed it with a known set of corresponding inputs and outputs and the network will adjust itself to “learn” and recognize patterns in the data. Eventually, the network will be able to apply this pattern to data not previously seen [7]. Thus, artificial neural networks are commonly used in image recognition and processing because of their inherent robustness and adaptability [6] [7] [8] [9]. They are able to solve classification problems that do not have a straightforward algorithmic solution [5] [7] [8] [9] [10].

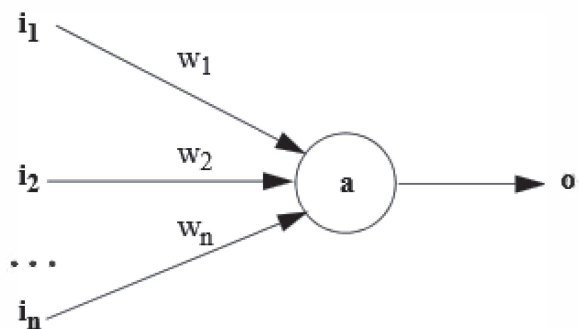


Figure 1. Artificial Neuron [7]

Figure 1 shows a single artificial neuron. A neuron may have any number of inputs and a single output. Each input of a neuron is associated with a certain weight. In order to get the output value of the neuron, each input is multiplied with its corresponding weight and the products added together. After getting this sum S , the actual output value of the neuron is dependent on the activation function a , which is an additional process that the sum S goes through to get the value of the output O . For some designs, the sum is compared with a certain threshold value. If the sum is higher than this threshold value,

the output value of the neuron is set to 1. Otherwise, the output is zero [7].

In this paper, however, the sigmoidal output function is used as the activation function. Using this function, the output of the neuron is allowed to take values between 0 and 1. This allows us to see how close the neural network is to the ideal answer of 0 or 1 during training and adjust the weights of the network accordingly. In order to get the sigmoidal output O , equation 1 is used on the sum S [7].

$$O = \frac{1}{1+e^{-S}} \quad (1)$$

It is clear that with this equation, even as the sum S gets arbitrarily high, the output will never reach 1, and even if S gets arbitrarily low, the output will never reach 0.

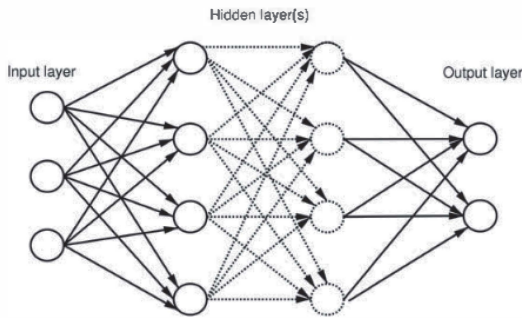


Figure 2. Feedforward Neural Network [7]

Figure 2 shows a group of neurons interconnected into a complete feedforward neural network. Feedforward neural networks are defined as having three or more layers of neurons with no feedback path anywhere. The output from one layer of neurons is directly propagated to the next. Every feedforward neural network is composed of three parts: the input layer, the hidden layers, and the output layer. The input layer is directly connected to the external inputs of the network while the output layer is the last layer in the network giving out the results. Any layers in between these two are in the hidden layer, for they cannot be directly seen from the outside [7]. The particular network architecture used in this paper is discussed in the next chapter.

III. ARCHITECTURE OF THE SYSTEM

Figure 3 shows a block diagram of the entire system in the paper. The input images, i.e. the frames to be searched for quadrotors in flight, are input into the pre-processing system so that it will be suited for processing by the main neural network. After this, the artificial neural network, trained through backpropagation and a database of training pictures, will determine whether the image contains a quadrotor or not, outputting either a value close to 1 or 0.

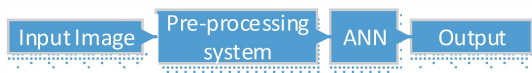


Figure 3. Block Diagram of System Architecture

A. Pre-processing

In order to input images into the system, the image has to be converted into a format suited for the neural network. Therefore, before entering it into the network, all images were

converted into a grayscale image with a 100x40 pixel resolution in RAW format. The RAW format for a 100x40 grayscale image is an array of 4000 values, where each value corresponds to the color of a single pixel of the image. Each value is an integer ranged from 0 to 255, where 0 means that the particular pixel is purely white, while 255 means that the particular pixel is purely black. After the conversion, the 4000 values are entered into the input layer of the neural network.

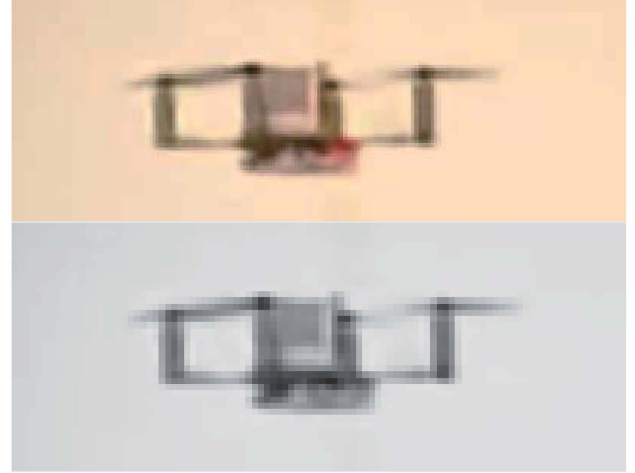


Figure 4. Image pre-processing

Figure 4 shows an example of the preprocessing done to an image before it is used in the neural network. The first picture shows the image prior to pre-processing. The second picture shows the image after pre-processing. Notice the conversion of the original image into grayscale. Also, the pixelation of the pre-processed image shows the conversion of the original image into the lower 100x40 resolution.

B. Architecture of the Neural Network

The network used in this paper is a simple feedforward neural network with only a single hidden layer. The input layer is composed of 4000 neurons while the hidden layer is composed of 100 neurons. The output layer consists of only one layer. In this network, each of the 4000 neurons in the input layer corresponds to one pixel of the image input into the system after pre-processing. This will be discussed in detail in the next section. The single output layer determines whether the image input into the network contains a CrazyFlie 2.0 in flight or not. An output close to 1 signifies that the image is a quadrotor. On the other hand, an output close to 0 signifies that the image is not a quadrotor.

C. Training

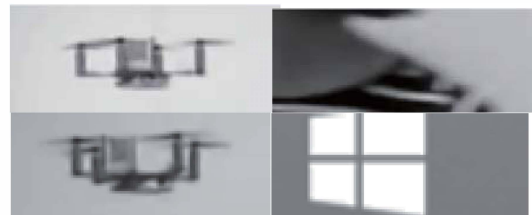


Figure 5. Pre-processed training images showing a picture with a quadrotor and a picture without a quadrotor

In order to train the neural network, a set of 448 images were used. Half of these images showed a CrazyFlie 2.0 in flight while the other half did not. An example of these images are shown in Figure 5.

These 448 images were further subdivided into two: a training set of 398 images and a validation set of 50 images. The validation set was used in order to prevent overtraining of the neural network. Overtraining occurs when the neural network fits itself too much to the training data, leading to a loss of generality and consequently, a loss of the neural network's ability to recognize images not part of the training data [7].

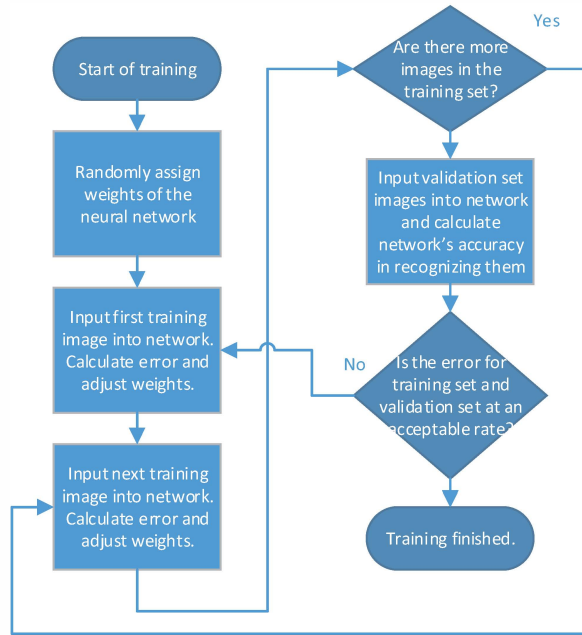


Figure 6. Flowchart of training process

Figure 6 shows the flowchart of the entire training process used for the neural network presented in this paper. The initialization of the network is done by randomizing the weights of the network. After that, the first training image is input into the network. The error from this allows network to adjust its own weights using the backpropagation algorithm, discussed in the next section. The next training image is then input into the network, its error calculated, and the weights of the network further adjusted. It must be noted that the training images are input alternately, that is, if the first training image contains a quadrotor, then the second does not while the third contains another quadrotor. This process is repeated until all 398 images in the training set have been input into the network. After this, the validation set is then used to test the accuracy of the neural network for data it has not seen before. The validation set is not used to adjust the weights of the network. It is used simply to see how well the network recognizes images it has not seen before and prevent overtraining. Once the validation set recognition accuracy has stopped increasing, signifying the start of overtraining, the training process is finished.

D. Backpropagation Algorithm

To adjust the weights and biases of the neural network, a standard backpropagation algorithm is used [11] [12]. The first step is to apply the input image into the network and calculate its output. This will give you an error that will be used for backpropagation. To calculate the error, the following equation is used.

$$E = o(1 - o)(Target - o) \quad (2)$$

Where E is the error, o is the output of the network, and target is the desired output of the network i.e. 1 if the image contains a quadrotor, 0 otherwise.

After getting the error, the next step is to correct the weights of the output neuron. To do this, we use equation 3.

$$W_{new} = W_{old} + LR * (E * O_{hidden}) \quad (3)$$

Where O_{hidden} is the output of the hidden layer neuron which the weight being calculated is associated with. LR is called the learning rate, a parameter between 0 and 1 that is used to adjust the rate at which the neural network is trained. A higher learning rate means that the neural network will learn faster, however, a learning rate too high may lead to instability of the network.

After changing the weights of the output layer, the next step is to change the weights associated with the hidden layer. First, we get the error of the hidden layer. However, this is not as straightforward as getting the error for the output layer because we have no target for the hidden layer i.e. we do not know what its correct value must be. Therefore, we use the equation

$$Error = O_{hidden}(1 - O_{hidden})(E * W_{new}) \quad (4)$$

Where E is the error of the output neuron calculated in equation 2 and W_{new} is the output layer weight associated with that hidden neuron. After getting the errors for the hidden layer neurons, use equation 3 to get the new weights.

IV. EXPERIMENT RESULTS

A. Training Results

The first experiment was done to be able to determine the best time to stop the training for the neural network. As discussed in the previous chapters, training too little may lead to the network not performing as well as it could, while training too much may lead to over-fitting of the network to the training data. For each cycle of training, the recognition rate over the training data and the recognition rate over the validation data were both recorded.

Figure 7 graphically shows the improvement of the neural network in recognizing images in the training data and the validation data respectively as the number of training cycles increases.

The graph shows two lines, one for the network's recognition over the training data, and one for the network's recognition over the validation data. It can be seen that as the number of cycles increase, the recognition rate for the training data continuously increases until it reaches a perfect recognition rate and stays there. However, the validation data set recognition rate peaks at somewhere near cycle 25 and falls

off after, indicating that overtraining has been reached. Thus, the network is considered trained after cycle 25.

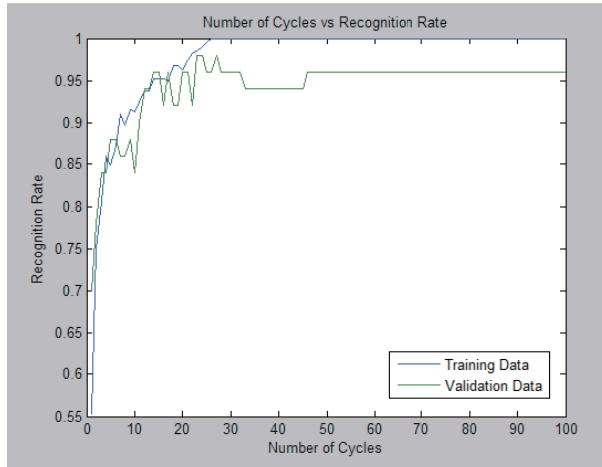


Figure 7. Number of cycles vs. Recognition Rate

B. Testing Results

In order to test the trained neural network, a set of 100 new images were used as test images. If an image input to the network resulted in an output of greater than 0.6, the image was considered a quadrotor. If the image input resulted in an output of less than 0.4, the image was considered to not be a quadrotor. Between 0.4 and 0.6, the image was considered unrecognized. 50 images contained images of the quadrotor in flight while the other 50 did not contain quadrotors. Out of the 100 images, 94 were correctly recognized. The other 7 were either falsely recognized or unrecognized i.e. generated an output between 0.4 and 0.6.

Figure 8 shows a graphical representation of the error of the neural network over the 100 test images. A value for an image above the green threshold line, i.e. 0.40, means that the image was incorrectly identified by the neural network. It is clear from viewing the graph that there were 6 images incorrectly identified by the network. Otherwise, most of the images kept below a 0.2 absolute error, showing the effectiveness of the neural network.

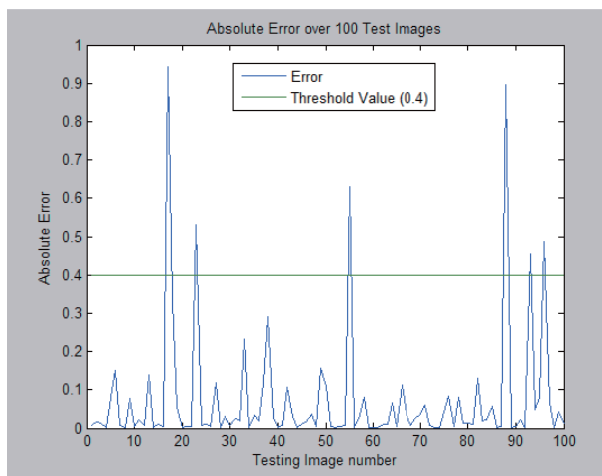


Figure 8. Absolute Error over 100 Test Images

Figure 9 shows a few of the correctly recognized images in the training set. It should be noted that even with the different backgrounds, quadrotor orientations, and noise of the various quadrotor images, the neural network was still able to identify them as quadrotors in flight. This shows the robustness of the trained network and its tolerance to noise.

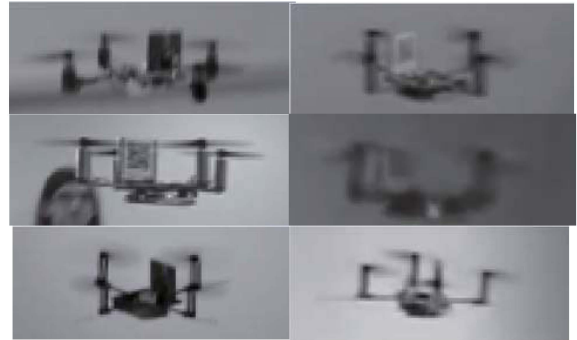


Figure 9. Correctly Recognized Images

Figure 10 shows the six images that were incorrectly recognized by the algorithm. It can be noticed that the quadrotors in the images are positioned at the bottom of the image. This may be a reason for the neural network being unable to identify the images. A solution to this may be to add more training examples such that the algorithm may learn to correctly generalize this case. In any case, the overall recognition rate of 94 percent is still acceptable for the application.

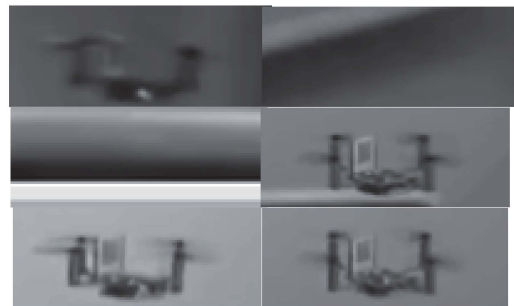


Figure 10. Incorrectly Recognized Images

C. Runtime

Because the network is expected to be used in a real-time application i.e. quadrotor swarm neighbour identification, the time it takes for a neural network to generate an output for an input has to be at a minimum. During the testing phase, it was found that the neural network was able to generate an output from an input in 0.017 seconds. This is deemed fast enough for the real-time application it is designed for.

V. DISCUSSION AND ANALYSIS OF RESULTS

The neural network proposed in this paper has shown that it is robust and accurate enough to use in its intended application of neighbour recognition in quadrotor swarms. The testing result of 94 percent accuracy shows that it is appropriate for use in a real-time system wherein hundreds of images per second from a camera will be processed and searched for a neighbouring quadrotor. A filtering algorithm may be used to

offset the error of the network. Additionally, the accuracy of the network may be further improved by increasing the number of images used for training.

The threshold value of 0.4 for determining whether the absolute error is too high for a particular image may also be discussed. It can be seen that decreasing the threshold to 0.3 will actually leave the result of the testing phase unchanged and the accuracy over the test images remaining at 94 percent.

VI. CONCLUSIONS AND RECOMMENDATIONS

In this paper, a feedforward neural network was designed and trained to recognize images of a CrazyFlie 2.0 quadrotor in flight for the purposes of using it in a neighbor sensing system for a swarm of quadrotors. The idea was to be able to give the quadrotor the ability to identify where its neighbors are in the swarm and use this information to determine its own behavior. After training the network using backpropagation and a database of 448 training images and 50 validation images, the testing results showed that the network was able to correctly identify whether the picture contained a quadrotor in flight at least 94 percent of the time even under different lightings, backgrounds, or quadrotor orientations. In practice, the neural network is expected to try to identify its neighbors at least a few hundreds of times per second. Due to this high refresh rate, a 94 percent accuracy is acceptable for the task.

For future works, even more training data should be gathered. 448 samples for training the neural network is a small number when compared with the number of angles, lighting schemes, and backgrounds that quadrotors in flight may find themselves in. Increasing the number of training data will allow the network to generalize even better.

As in all real-time applications, speed is essential. Therefore, techniques to decrease the runtime of the neural network may be explored, such as reducing the number of input and hidden layer neurons or even using specialized neural network hardware.

Research may also be done to find architectures that work better than the discussed feedforward neural network. Additional hidden layers and new training techniques such as dropout or regularization [7] may be used. Also, additional output neurons may be added to look for specific angles and orientations of the quadrotor. This will lead to the gathering of more data that may be used to distinguish various things, such as the quadrotor's movements.

REFERENCES

- [1] G. Beni, "From Swarm Intelligence to Swarm Robotics," in *Swarm Robotics*, Springer, 2004, pp. 1-9.
- [2] A. A. Bandala, R. R. P. Vicerra and E. P. Dadios, "Swarming Algorithm for Unmanned Aerial Vehicle Quadrotors," *Journal of Advanced Computational Intelligence and Informatics*, vol. 8, no. 5, 2014.
- [3] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*, Springer, 2004, pp. 10-19.
- [4] T. S. Kim, K. Stol and V. Kecman, "Control of 3 DOF Quadrotor Model," in *Robot Motion and Control*, London, Springer London, 2007, pp. 29-38.
- [5] Encyclopedia of Machine Learning, Springer, 2010.
- [6] I. Quraishi, J. P. Choudhury and M. De, "Image recognition and processing using Artificial Neural Network," *Recent Advances in Information Technology (RAIT)*, pp. 95-100, 2012.
- [7] T. Braunl, *Embedded Robotics*, Berlin: Springer Berlin Heidelberg, 2006.
- [8] K. Aihara, M. Okada, M. Adachi and M. Watanabe, "Neural Networks," in *Encyclopedia of Neuroscience*, Springer Berlin Heidelberg, 2009, pp. 2588-2592.
- [9] T. Cheema, I. Qureshi, A. Jalil and A. Naveed, "Blur and image restoration of nonlinearly degraded images using neural networks based on modified ARMA model," in *Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International*, 2004.
- [10] N. Khoa, K. Sakakibara and I. Nishikawa, "Stock price forecasting using back propagation neural networks with time and profit based adjusted weight factors," in *SICE-ICASE, 2006. International Joint Conference*, Busan, 2006.
- [11] P. Munro, "Backpropagation," in *Encyclopedia of Machine Learning*, Springer US, 2010, p. 73.
- [12] A. Andrew, "Backpropagation," in *A Missing Link in Cybernetics*, New York, Springer-Verlag New York, 2009, pp. 85-104.