ISSN 2278-3091

Volume 9, No.1, January – February 2020

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse32912020.pdf

https://doi.org/10.30534/ijatcse/2020/32912020

Training of a deep learning algorithm for quadcopter gesture recognition

Calvin Ng[1], **Alvin Chua**[2]

[1, 2] Mechanical Engineering Department, De La Salle University - Manila, Philippines,

[1]calvin_alexander_ng@dlsu.edu.ph

[2]alvin.chua@dlsu.edu.ph

ABSTRACT

Traditional methods to control Unmanned Aerial Vehicles are unintuitive and susceptible to radio interference. Recent research has shown that hand gestures are the most intuitive method for quadcopter control. Also, deep learning in the form of a convolutional neural network is a more compatible approach to gesture recognition than other methods. This paper presents the design, and training of a deep learning convolutional neural network for gesture recognition and tracking of a quadrotor Unmanned Aerial Vehicle. The neural network was coded in Python using the Keras library and was trained on a laptop computer. Inference was performed on a Raspberry Pi 4 computer that is intended for use as a companion computer aboard a quadcopter.

Key words: companion computer, convolutional neural network, deep learning, unmanned aerial vehicle

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs), commonly called 'drones' is a very broad term. Basically, it describes any kind of aircraft without a human pilot onboard [1]. This is why there are many different kinds of UAVs, such as powered gliders, powered parachutes, helicopters, fixed wing aircraft, and quadcopters [2]. Quadcopters the commonly used design for small UAVs due to their simplicity [3]. The fact that there can be many possible kinds of UAVs means that they can be adapted to many different uses, such as mapmaking, surveillance, reconnaissance, search and rescue, and more [1]. Multiple kinds of autopilot systems have also been developed and evaluated for their performance and effectiveness in various situations [4].

Previous researches have uncovered shortcomings in the traditional method of Quadcopter control. The most common method to control quadcopters is radio control (RC), where control commands are given by the use of keypresses on a controller. This has three main shortcomings. According to [5], there are significant levels of time delay lag, according to [6], the conventional scheme for quadcopter control is unnatural and unintuitive, lastly, according to [7], RC is highly susceptible to electromagnetic interference.

Based on [8], communication systems based on radio waves also suffer from latency, disruptions, or unreliability when the radio device is in motion, such as in a train. Thus, a UAV may also suffer radio problems due to the fact that it is in motion.

Past research has also established that increasing the intuitiveness of UAV control by increasing the interaction between the human and the drone results in a significant improvement in the learning process of senior high school students [9]. Towards a similar goal, [10] implemented a brain-computer interface to improve the effectiveness in controlling portable robots.

The currently available methods of quadcopter control have significant shortcomings in terms of effectiveness and intuitiveness. Recent research, however has uncovered the desirability of hand gestures as an efficient and effective solution. Some research [11] has also made a method to analyze the facial expressions of humans. Thus, there is currently an existing need for a study that will implement hand gestures to control a quadcopter.

The existing methods of quadcopter control typically use RC, which is limited by the range of radio signals and more importantly, electromagnetic interference. Much work has been done in the past to control a quadcopter using computer vision methods, however the common impediment is the connection to a ground-based desktop computer. This approach is clearly still limited by the same electromagnetic interference that limits RC.

From this, it follows that a control method that is not dependent on a constant radio connection to a ground-based computer will alleviate the limitations inherent to radio control and substantially improve the effectiveness of quadcopter UAVs. Additionally, the focus on hand gestures as a method of quadcopter control will improve the accessibility of quadcopters to the common person because it is more intuitive and natural for humans to use hand gestures in expressing their intentions [6].

Acknowledgment: This research was funded by the Department of Science and Technology, Engineering Research and Development for Technology (DOST ERDT) of the Philippines.

Consequently, the main objective of this paper is to develop a deep learning algorithm for hand gesture recognition which will run on a Raspberry Pi 4 computer. The algorithm will be tested through the use of an actual image to verify the training performance of the algorithm.

1.1 Hand Gestures for Quadcopter Control

Gestures are intuitive and natural methods that enhance the experience of humans in controlling robots. However, it is important to study the preferred gestures used by humans and to include them in the design process in order to maximize the intuitiveness of the control system.

The research done by Obaid et al. in 2016 collected and analyzed 300 samples of gesture data obtained from 25 participants in Sweden. The study aimed to investigate the gesture movements that humans naturally invent to correspond to normal drone operations. Specifically, they tested gestures for twelve actions as follows: move left, move right, move forward, move backward, go up, go down, turn left, turn right, take off, and land.
The results of their study found that 72% of gestures were "deictic" type gestures. In other words, they were movements that represent the idea of motion, such as pointing fingers, or waving of arms. This is in contrast to "iconic" type gestures, which made up 9% of the gestures and involved actually performing the task. One example of an iconic gesture in the study was for a person to take a step to the left when the participant wanted the drone to move to the left. Furthermore, the study found that approximately equal amounts of the gestures involved one hand and two hands, with 45% and 42% respectively. Thus, the study found that the hands constitute a vast majority of gestures with a combined percentage of 87%.

1.2 Using Deep Learning for Gesture Recognition

Work by Strezoski et al. in 2017 has established Deep Convolutional Neural Networks (DCNN) as a feasible method to perform vision-based hand gesture recognition. According to their research, the main methods for obtaining hand gesture data can be divided into Data-Glove and Vision methods. However, the main drawback for Data-Glove method is the fact that there is a large potential for hardware issues including connector sensitivity and hardware sensitivity [12].

Based on the work done by [7] and [8], it is preferable to use vision-based methods based on their simplicity. Consequently, however, a need arises to efficiently process and interpret the large amount of data gathered using cameras in real time. Additionally, the recognition system must be able to perform despite lighting changes, background variations, subject variations, and camera variations [14].

According to [12], recent advancements have made deep learning approaches to be the superior choice for computer vision problems, particularly Convolution Neural Networks because of their similarity to the human brain.

2. DEEP CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks are machine learning approaches that are based on biology. Essentially it emulates a collection of cells that are sensitive to certain stimuli and pass on some activating stimulus to subsequent cells [15]. Figure 1 is a simple example of a set of artificial neurons.

In other words, the inputs of layer *m* come from a subset of neurons in later *m*-1, which act similarly to the retina of the eye. In a deep neural network, there may be many layers between the input and the output, depending on the designer of the network.

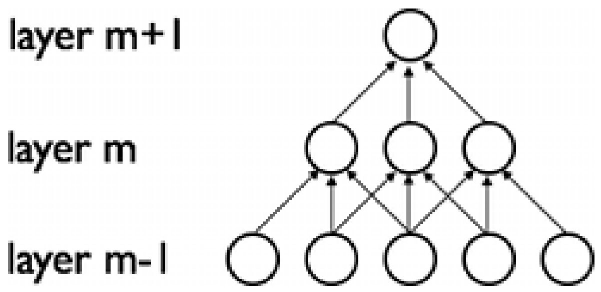

Figure 1: Neuron Interconnection[15]

2.1 Backpropagation

Backpropagation is a well-established method to facilitate learning in a Neural Network [16]. Essentially, it calculates the error between the output of the Neural Network and the desired output, and propagates an adjustment through the layers. The outputs of a neural network with respect to its inputs can be described by (1) – (4) below.

$$x_i = X_i; \quad 1 \leq i \leq m \ \#(1)$$

$$net_i = \sum_{j=1}^{i-1} W_{ij} x_i; \quad m < i \leq N + n \ \#(2)$$

$$x_i = s(net_i); \quad m < i \leq N + n \ \#(3)$$

$$Y_i = x_{i+N}; \quad 1 \leq i \leq n \ \#(4)$$

Where:

- X is the input
- Y is the output
- net_i is the total excitement level of a neuron
- $N+n$ is the overall number of neurons
- N is an arbitrary constant greater than or equal to m
- W is the weight that modifies the input x

The function *s* is the function that determines the activation of a neuron, commonly a sigmoidal function.

Backpropagation is essentially manipulating the values for W to minimize the errors as shown by (5).

$$E = \sum_{t=1}^{T}\sum_{i=1}^{n}\left(\frac{1}{2}\right)\left(\widehat{Y}_i(t) - Y_i(t)\right)^2 \#(5)$$

Where:

- E is the error
- T is the total number of output values
- n is the total number of outputs

This process is essentially similar to the method of least squares, but the difference is in its implementation into neurons. Initially, arbitrary values for W are chosen based on prior knowledge. Next, the outputs Y(t) and error E(t) are calculated, along with the change of E with respect to the weights. If increasing a particular weight would lead to increased error, it is adjusted downwards. Likewise, if decreasing a particular wright would lead to increased error, it is adjusted upwards.

2.2 Convolution

Due to the fact that there can be many possible weights for a complex neural network, some approaches combine multiple neurons together with a common weight. This forms an object called feature map.

The term *convolution* means to repeatedly apply a mathematical function to an image. Specifically, the input is subjected to a linear filter, a bias, and a nonlinear function in order to obtain the final feature map. As seen in (6) below.

$$h_{ij}^{k} = \tanh\left(\left(W^k * x\right)_{ij} + b_k\right) \#(6)$$

Where:

- h is the feature map
- W^k is the weight filter
- b_k is the bias

Applying feature maps in this way means that the neural network is able to detect features regardless of their position in the image. Also, it increases efficiency by reducing the number of independent variables to be learned.

3. METHODOLOGY

As discussed above, computer vision algorithms tend to require large amounts of computing power, and is beyond the capabilities of the flight controller. Thus, recent publications have either utilized desktop computers over Wi-fi, or added lightweight secondary processors.

3.1. Companion Computer

There have been many advancements in miniaturization of computers in recent years, with several different models available to choose from. Three of the models that were considered are summarized below.

Table 1: Companion Computers

	Nvidia TX2	**Latte Panda**	**Raspberry Pi 4**
Cost	$321.23	$103.51	$35.00
Size	50 mm x 87 mm	88 mm x 70 mm	88 mm x 58 mm
Mass	85g	55g	46g
CPU	ARM Cortex A57 (2GHz)	Intel Atom Z8350 (1.92 GHz)	BCM2711B0 (1.5GHz)

Based on Table 1, it is evident that the Raspberry Pi is desirable in three out of the four criteria (cost, size, and mass). Thus, it was chosen as the companion computer for this research.

Table 2: Comparison of Cameras

	OpenMV	**Raspberry Pi Camera**
Cost	$65	$29.95
Size	35.6mm x 44.4mm	25mm x 23mm
Mass	16g	3g
Resolution	480p Standard Definition	4k HD, 1080p Video

3.2. Choice of Camera

The camera to be chosen must be compatible with the Raspberry Pi in order to be used in this study. Additionally, the main design considerations include cost, weight, and size. Two cameras were evaluated as viable options for this research. Based on the information from table 2, the choice with the clear advantage is the Raspberry Pi camera. It is lighter, smaller, and cheaper than the OpenMV Camera. Thus, the research proceeded with the Raspberry Pi camera.

3.3. List of Gestures

The specific had gestures to be used to test the system will be based on the research of [6] in order to maximize their intuitiveness. Specifically, the they are listed in table 3.

Table 3: List of gestures

Action	**Description**	**Gesture Type**	**Body Parts**
Move left	Swipe left	Deictic	Hand
Move right	Swipe right	Deictic	Hand
Move forward	Push front	Deictic	Hand
Move backward	Pull back	Deictic	Hand
Ascend	Move up	Deictic	Hand
Descend	Move down	Deictic	Hand

4. DEVELOPMENT OF THE ALGORITHM

Considering that the Raspberry Pi is limited in terms of processing power, it is imperative to choose a model that requires the least processing power to classify images. Thus, the algorithm will be based on the Custom Model as described by [12] because it requires the least amount of time to classify and performs with an acceptable level of accuracy. Specifically, the model they describe contained a total of 13 layers, with 5 convolutional layers and 5 pooling layers with a field of view of 194x194 pixels. The detailed breakdown of the network is shown in Table 4.

Table 4: Original Custom Model

Layer	Type	Units	Kernel
0	Input	194x194	N/A
1	Convolutional	192x192	4x4
2	Max pooling	96x96	2x2
3	Convolutional	92x92	4x4
4	Max pooling	46x46	2x2
5	Convolutional	42x42	5x5
6	Max pooling	21x21	2x2
7	Convolutional	18x18	4x4
8	Max pooling	9x9	2x2
9	Convolutional	6x6	5x5
10	Max pooling	3x3	2x2
11	Fully connected	600	1
12	SoftMax	6	1

4.1. Neural Network Architecture

However, the neural network had to be simplified in order to fit within the memory constraints of the Raspberry Pi. This was due to the fact that when the original architecture was tested by running on the Raspberry Pi, it was able to process less than one frame per second. The finalized architecture is as shown in Table 4.

Table 4: Modified Custom Model

Layer	Type	Units	Kernel
0	Input	100x100	N/A
1	Convolutional	32x32	4x4
2	Max pooling	12x12	4x4
3	Convolutional	8x8	4x4
4	Max pooling	6x6	2x2
5	Convolutional	4x4	4x4
6	Max pooling	2x2	2x2
7	Fully connected	16	1
8	SoftMax	7	1

Specifically, the input layer was reduced to 100 neurons by 100 neurons in line with the dimensions of the input images. The number of layers was reduced from 13 to 9, which was necessary to reduce the overall number of neurons in order to reduce the memory requirement. Also, the final softmax layer was increased from 6 to 7, in accordance to the instruction of [12] to follow the number of output classes.

4.2. Training Data

The training data was composed of 5432 images taken in multiple different locations including a hallway with yellow light, a hallway with dim light, a hallway with white light, and a bedroom with white light. The training data was captured in video form using a Samsung Galaxy S7 and was then split into individual frames using VLC media player. This resulted in images of the subject in various arm positions including intermediate positions.

The figure 2 provides some samples of the training images. It was important to consider the composition of the training images so that there would be a wide variety of gesture information. Specifically, it was important to include images with bright light, dim light, yellow light, and uneven light. Many variations of shirts were also included such as red, blue, orange, and green stripes.

Besides the author, images of other people were also included in the training data but they did not consent to their photos being shown publicly.

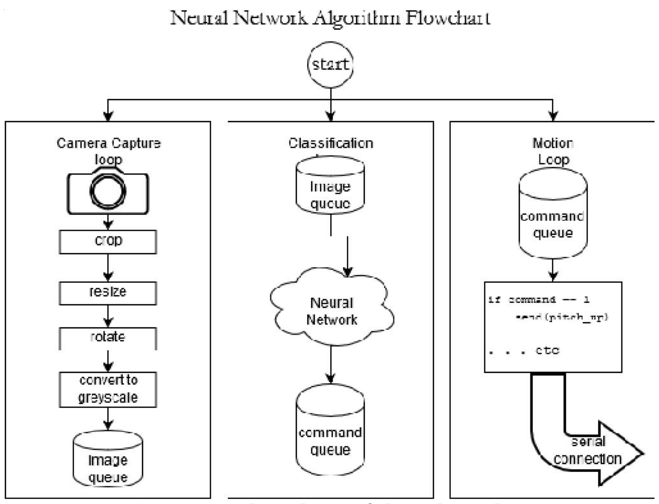

Figure 3: Flowchart of the Algorithm

5. RESULTS

The neural network was programmed using Python 3.7, with the Keras deep learning library and a Tensorflow backend. While the original Custom Model would have taken a significantly long time to train, the simplified model was trained in one hour. The results of the training are shown in table 5.

Table 5: Training results

	precision	recall	F1-score	support
Back	0.90	0.92	0.91	193
Down	0.91	0.94	0.92	170
Forward	0.96	0.95	0.95	192
Idle	0.96	0.96	0.96	201
Left	0.94	0.91	0.93	190
Right	0.91	0.93	0.92	196
Up	0.95	0.93	0.94	216
Accuracy			0.93	1358
Macro Average	0.93	0.93	0.93	1358
Weighted Average	0.93	0.93	0.93	1358

Notably, the neural network was able to score an F1-score greater than or equal to 0.91 for all cases with a precision of at least 0.90 and a recall of at least 0.91. This means that the neural network will be able to classify an image with at least 90% true positives with 10% or fewer false positives and false negatives.

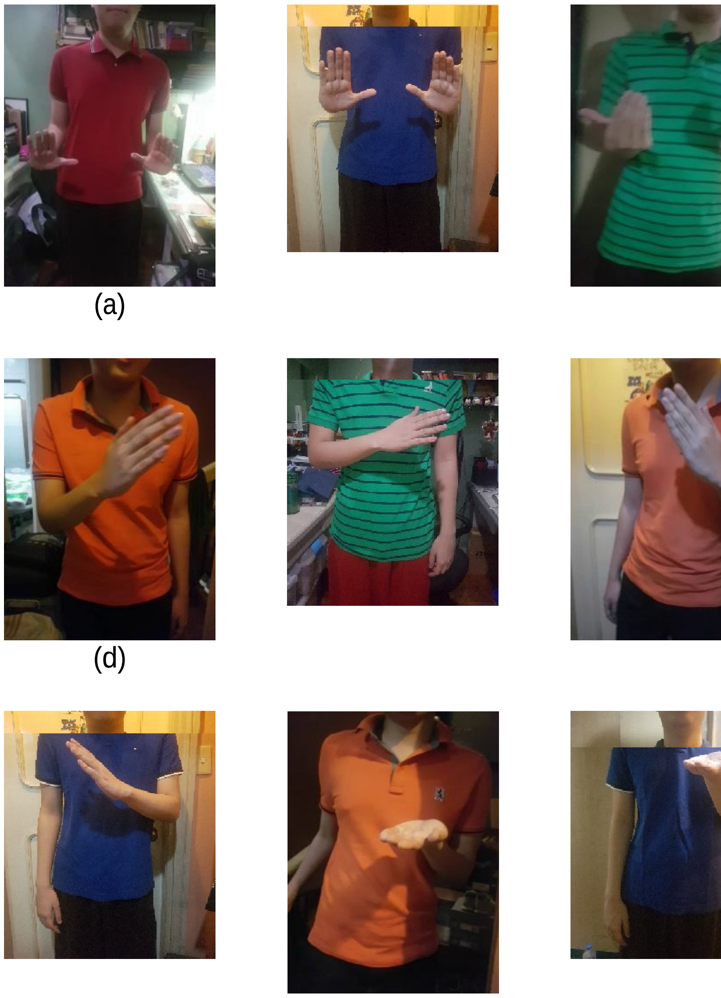

Figure 2: Sample Training Data

6. CONCLUSION

This paper has presented the implementation and training of a neural network algorithm for hand gesture recognition to be used for a quadcopter companion computer. The gestures used for control were based on established research which maximized intuitiveness and user friendliness. Also, a customized neural network architecture was developed to fit within the limitations of the hardware.

The authors recommend future research that may include an actual flight test of a quadcopter in order to verify its effectiveness in real world scenarios. In addition, further research can be done on various kinds of neural network architectures which may further improve the speed and accuracy of the system.

ACKNOWLEDGEMENT

The authors express their gratitude to the members of the De La Salle University Unmanned Aerial Vehicle Team for providing training on basic UAV construction and programming.

REFERENCES

[1] M. Wagner, **Unmanned Aerial Vehicles**, *Max Planck Encyclopedia of Public International Law*. Oxford University Press, 2015.

[2] C. Zhang and J. M. Kovacs, **The application of small unmanned aerial systems for precision agriculture: A review**, *Precis. Agric.*, vol. 13, no. 6, pp. 693–712, 2012. https://doi.org/10.1007/s11119-012-9274-5

[3] T. Luukkonen, **Modelling and Control of Quadcopter**, Espoo, Finland, 2011.

[4] C. Dim, F. Nabor, G. Santos, M. Schoeler, and A. Chua, **Novel Experiment Design for Unmanned Aerial Vehicle Controller Performance Testing**, *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 533, no. 1, p. 012026, May 2019. https://doi.org/10.1088/1757-899X/533/1/012026

[5] R. Vivek Krishna, B. S. Sathish, P. Ganesan, P. Jawahar Babu, and R. Abilash, **Design of voice and hand gesture controlled Quadcopter**, in *IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication Systems ICIIECS'15*, 2015.

[6] M. Obaid, F. Kistler, G. Kasparavičiūtė, A. E. Yantaç, and M. Fjeld, **How would you gesture navigate a drone?: a user-centered approach to control a drone**, *Proc. 20th Int. Acad. Mindtrek Conf. - Acad. '16*, pp. 113–121, 2016. https://doi.org/10.1145/2994310.2994348

[7] K. Natarajan, T.-H. D. Nguyen, and M. Mete, **Hand Gesture Controlled Drones: An Open Source Library**, 2018. https://doi.org/10.1109/ICDIS.2018.00035

[8] T. M. Nyandika, G. Okeyo, and M. Kimwele, **Enhancing service availability during handover in wireless communication-based train control systems**, *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 7, no. 3, pp. 41–51, 2018.

[9] C. B. Toribio, J. D. Espinola, J. E. Ignacio, A. Chua, and J. P. Lacaden, **Virtual Simulations for Drone Education of Senior High School Students**, *Int. J. Eng. Adv. Technol.*, vol. 8, no. 6S3, pp. 220–226, Nov. 2019.

[10] S. Rao, M. Kaivalya, C. Janaki Devi, and M. V. N. Raju, **International Journal of Advanced Trends in Computer Science and Engineering**, *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 7, no. 6, pp. 159–162, 2018.

[11] S. Rao, G. Srujana, B. Priyanka, and R. Nanda, **Effective Analysis of Human Facial Appearance using JAFFE Images**, vol. 7, no. 6, pp. 149–151, 2018. https://doi.org/10.30534/ijatcse/2018/17762018

[12] G. Strezoski, D. Stojanovski, I. Dimitrovski, and G. Madjarov, **Hand gesture recognition using deep convolutional neural networks**, *Adv. Intell. Syst. Comput.*, vol. 665, no. November, pp. 49–58, 2017.

[13] Y. Choi, I. Hwang, and S. Oh, **Wearable gesture control of agile micro quadrotors**, *IEEE Int. Conf. Multisens. Fusion Integr. Intell. Syst.*, vol. 2017-Novem, pp. 266–271, 2017. https://doi.org/10.1109/MFI.2017.8170439

[14] E. Ohn-Bar and M. M. Trivedi, **Hand Gesture Recognition in Real Time for Automotive Interfaces: A Multimodal Vision-Based Approach and Evaluations**, *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2368–2377, Dec. 2014.

[15] LISA Lab., **Convolutional Neural Networks (LeNet) — DeepLearning 0.1 documentation**, 2018. [Online]. Available: http://deeplearning.net/tutorial/lenet.html. [Accessed: 07-Dec-2018].

[16] P. J. Werbos, **Backpropagation through time: what it does and how to do it**, *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990. https://doi.org/10.1109/5.58337