Troop camouflage detection based on deep action learning

Muslikhin[1], Aris Nasuha[2], Fatchul Arifin[3], Suprapto[2], Anggun Winursito[2]
[1]Department of Electronics Engineering Education, Universitas Negeri Yogyakarta, Yogyakarta, Indonesia
[2]Department of Electronics Engineering, Universitas Negeri Yogyakarta, Yogyakarta, Indonesia
[3]Department of Electronics and Informatic Engineering Education, Universitas Negeri Yogyakarta, Yogyakarta, Indonesia

Article Info

Article history:

Received Dec 8, 2021
Revised May 22, 2022
Accepted Jun 2, 2022

Keywords:

Action learning
Deep learning
Deep action learning
SquezeeNet
Troop camouflage
YOLOv3

ABSTRACT

Detecting troop camouflage on the battlefield is crucial to beat or decide in critical situations to survive. This paper proposed a hybrid model based on deep action learning for camouflage recognition and detection. To involve deep action learning in this proposed system, deep learning based on you only look once (YOLOv3) with SqueezeNet and the fourth steps on action learning were engaged. Following the successful formulation of the learning cycle, an instrument examines the environment and performance in action learning with qualitative weightings; specific target detection experiments with view angle, target localization, and the firing point procedure were performed. For each deep action learning cycle, the complete process is divided into planning, acting, observing, and reflecting. If the results do not meet the minimal passing grade after the first cycle, the cycle will be repeated until the system succeeds in the firing point. Furthermore, this study found that deep action learning could enhance intelligence over earlier camouflage detection methods, while maintaining acceptable error rates. As a result, deep action learning could be used in armament systems if the environment is properly identified.

Corresponding Author:

Muslikhin
Department of Electronics Engineering Education, Universitas Negeri Yogyakarta
1st Colombo Street, Karangmalang Campus, Yogyakarta 55281, Indonesia
Email: muslikhin@uny.ac.id

1. INTRODUCTION

Troop camouflage in military operations is indispensable to trick and move as close as possible to the opponent, while from the opponent's side constantly trying to extract field conditions from possible enemy camouflage. Separately, the development of artificial intelligence has provided many benefits for recognition and detection purposes [1]. However, the problem of camouflage detection is considered to be difficult to overcome because distinguishing between objects and the same background requires a different strategy [2]. The camouflage subdomains recognition includes segmentation, distance measurement, and troop recognition. The use of several approaches is required and even relatively new methods.

Previous work by Shen *et al.* [3] introduced rapid camouflage detection using polarization and deep education. Unfortunately, the testing of this work uses artificial targets. It is similar with using deep learning; you only look once (YOLOv3) with an average accuracy of 91.55% [1]. Furthermore, it turns out that Xiao *et al.* [1] used a camouflage dataset that was not considered as a vague, for example, a fighter plane with a sky background or a frigate with an ocean background. However, the interesting point lies within how to camouflage the detected object according to the background, even though it is not necessary for the attacking troops in battle. A recent study introduced deep learning using camouflaged object detection with cascade and feedback fusion (CODCEF) [4] which can detect within 37 ms using an NVIDIA Jetson Nano device. Another study [5] used data augmentation to perform camouflage detection. This method is considered adequate with 99% accuracy, superior in a lightweight but does not specify the computer specifications used.

Although research [6], [7] had used many datasets, the detection results were not exceptionally good. Experiments conducted by [8] claimed that as the rapid camouflage detection using deep learning, this work still needs 0.82 seconds for a single detection process. A quite unique method is tried by Yan *et al.* [9]. According to [9], which offers the MirrorNet, it is effective for camouflage patterns detection with an accuracy up to 87%. Basically, deep learning that has been implemented in general still has some flaws. Examples of merging deep learning with several scenarios are becoming more common and have been widely applied, such as Shi *et al.* [10] and Tsai *et al.* [11], who used reinforcement learning (RL) and deep convolutional neural network (DCNN) for mobile robots, respectively [12]. Chen *et al.* [13] combined recurrent neural network (RNN), deep reinforcement learning (DRL), and long short-term memory (LSTM) in a comparable way; however, their capability was only around 47%. Visible weaknesses, the level of accuracy is determined during previous training. Where the number of datasets, learning rate, and epoch greatly affect performance. Another weakness is due to the nature of camouflage, which tends to have the same texture or pattern between objects against the background. Therefore, it is presumed that the current use of deep learning is no longer able to overcome the problem of camouflage detection. For this reason, other deep learning methods are needed, either as supervised or unsupervised learning [10], [14]–[17].

Current works by [11], [18]–[20] showed the use of deep learning followed by generative adversarial networks (GAN). Although GAN has gained popularity after being combined with other techniques in the camouflage concern, there is no improvement in its intelligence because the environment is examined from each perspective separately, and the system is trained with static data, which is insufficient for upgrading the knowledge itself [21], [22]. We are trying to decipher the weakness of GAN or deep learning, which is commonly used in camouflage problems. The key idea offered in this study is the ability to upgrade the intelligence of a deep learning for satisfactory detection. The concept was adopted from the educational world where action learning has long been applied, but for the fields of artificial intelligence, image processing or robotics, it has not been widely reported in scientific publications [14], [23]–[25].

The principle of action learning imitates human learning, where the learner will try to achieve the passing grade, and if he failed to achieve it and try to repeat [23], [25], [26]. For every effort to achieve passing grade, an evaluation is carried out by the instructor, which resulted to a need of an assessment instrument in action learning [27], [28]. In this light, action learning will have a repeated cycle and updates the evaluation of vision at different angles until it meets a certain level of passing grade. Besides the deep learning primary intelligence, it also learns to improve its capabilities by introducing the action learning. In practice, we will apply to troop camouflage for recognition and detection. Therefore, we expect that our system driven by deep action learning will be more accurate in detecting. Specifically, we propose to develop a rapid detection system using action learning that is robust in camouflage, which are frequently confronted while identifying troops for battlefield. This paper contains the following information:

- A deep action learning is employed to estimate troop camouflage, which is a pretty homogonous pattern, vary in battlefield environment, also various military uniform.
- We strive to be accurate in recognizing and detecting troop camouflage with deep action learning as basic detection.
- We occupied an action learning to update system knowledge independently. Self-correction for troop camouflage detection in the battlefield was given and assessed; it might contribute as alternative routes to military devices.

In section 2 of this rest article, we introduced our proposed system, and we examined the research method in section 3. Results and discussion on deep action learning applied to troop camouflage detection will be presented in section 4. Finally, in section 5, we conclude up the work and provide suggestions for future projects.

2. PROPOSED SYSTEM

Figure 1 depicts our entire system; the dashed line box represents the deep learning process in which YOLOv3 is combined with SqueezeNet. The image outside the dashed line is a chart of action learning. The combination of both is called a deep action learning, while the system is without preprocessing step on targets detection. Since we do not use preprocessing, we are worried about the detection precision. For this reason, action learning in camouflage detection optimizes the self-correction method; if the detection precision does not match the specified passing grade, the system will see the input image from a different alternative point of view. The red-green-blue (RGB) images with a resolution of 275×183 pixels ~ 640×480 pixels are used as inputs. The detection process in action learning occurs in the acting phase, where previously, the output of YOLOv3 was used as input in the planning phase. After the detection process is known, it will continue with the observation process; the detected target is observed again and its perspective is checked from a certain angle as a reflection discussion.

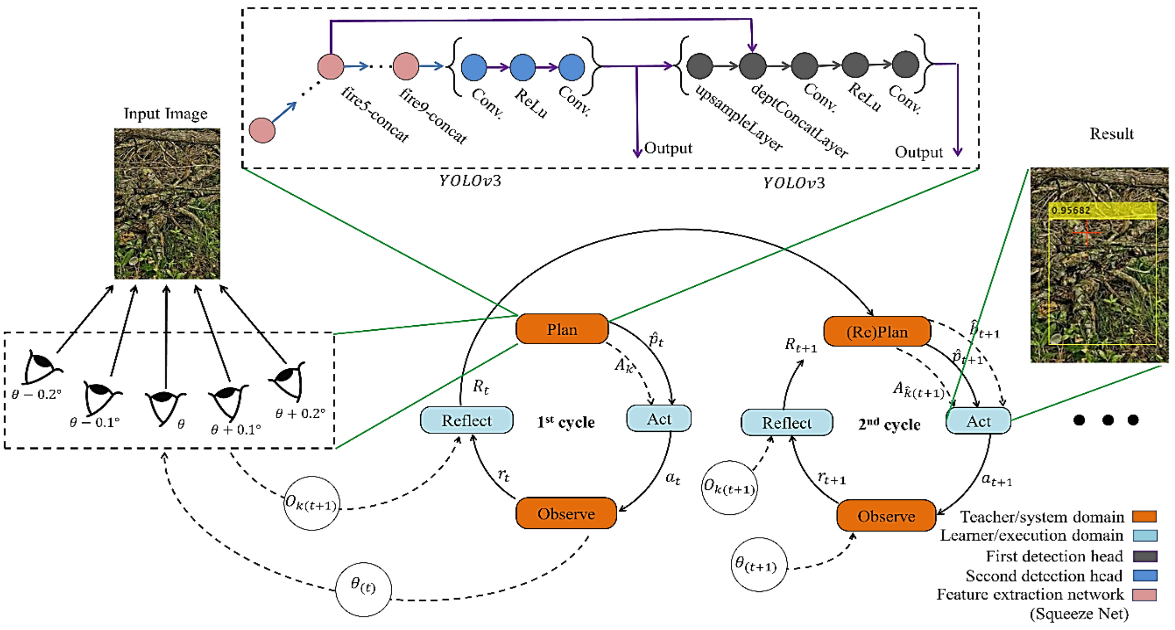


Figure 1. Overall architecture diagram

In action learning theory, there is no limit to when the cycle will end, and it depends on the performance of the ability to reach the passing grade. Therefore, in our proposed deep action learning may depend on the hardware specification that our proposed system is running on. The system built for deep action learning is ultimately run on a central processing unit (CPU) with a Core i5-6200 CPU@ 2.4 GHz (4 CPUs) processor speed and random-access memory (RAM) of 16 GB capacity with onboard high definition (HD) Graphics 520, 4 GB of memory. Meanwhile, the deep action learning algorithm was developed using MATLAB.

The deep action learning offered in this study was purely intended for detecting troops camouflage on the battlefield without the help of preprocessing or digital image processing intervention. In other words, the output of this system is the actual result of detection with a confidence value, then compared with various optimizers. There are three optimizers used in this study, stochastic gradient descent with momentum (SGDM), root mean square propagation (RMSProp), and adaptive moment optimization (ADAM). The findings were presented as a confidence level with a bounding box. The target will be determined using the square shape of the bounding box as a reference. Hence, detection utilizing deep action learning is a hybrid method, so it can be easy to describe separately in section 3.

3. METHOD

3.1. Proposed approach in deep action learning

Deep action learning was developed based on a combination of deep learning and action learning. YOLOv3 was selected as the method for deep learning using SqueezeNet as a feature extractor. The rectified linear unit (ReLU) function used in fire modules, kept the original SqueezeNet activation settings [19], [29]. The fully connected (FC) layers will follow the leaky ReLU function. Leaky ReLU is a modified version of ReLU with a bit of slope in the function output for negative data. As a result, the derivative is never zero; it can limit the appearance of silent neurons, resolving the issue of ReLU failing to learn when negative intervals are encountered. As (1), the term of Leaky ReLU is explained in (1).

$$\phi(x) = f(x) = \begin{cases} x, & x > 0 \\ 0.1x, & x < 0 \end{cases} \quad (1)$$

The categorical cross-entropy loss function in (2) will be used to tune our design during training. While training, we would utilize the categorical cross-entropy loss function in (2) to improve our model:

$$\text{loss} = -\sum_{i=1}^{n} \hat{y}_{i1} \log y_{i1} + \hat{y}_{i2} \log y_{i2} + \cdots + \hat{y}_{im} \log y_{im} \quad (2)$$

where the numbers n and m denote the number of samples and categories, respectively. The real value is represented by y, whereas the estimated value is represented by ŷ.

In practice, it is crucial to pay more attention to the categories with small samples when having the loss function, since it aims to resolve the problem of sample imbalance. We add loss components to the loss function, as shown in (3), to let the model training proceed smoothly and avoid overfitting.

$$\mathrm{loss} = -\sum_{i=1}^{n} \lambda_1 \hat{y}_{i1} \log y_{i1} + \lambda_2 \hat{y}_{i2} \log y_{i2} + \cdots + \lambda_m \hat{y}_{im} \log y_{im} \quad (3)$$

For each target category, the values of loss factor λ have been computed as presented in (4).

$$\lambda_i = \frac{C_n}{nN_i} \quad (4)$$

where the total number of samples is represented by C_n. The number of target categories is n, and N_i denotes the sample amount of class I.

Although action learning was presented, RL was a source of inspiration. RL used the Bellman equation to calculate a discounted value from the goal point; the multiple paths were trained to achieve supreme value. The value of each state in RL was defined in advance. On the other hand, with action learning, the value was eliminated and substituted with a real-time assessment based on the instrument, also known as a passing grade for students (system). At the same time, the passing grade is expressed by β_p. The assessment of the environmental value is derived from the eight evaluation indicators in Table 1.

Table 1. Assessment of the environment on deep action learning

Aspect (Indicators)	Scale/Probability (*n*) 0	1	2	3	ω
Planning and action					
Measure the YOLOv3 detection on an image δ (%)	>50	≤80	81~95	≥96	12
Assess the current passing grade β	-	≤75	76~90	≥91	3
Adjust view an input image θ	-	-	-	-	9
Suspect the appearance of bounding box ε	No	-	-	Yes	3
Observing and reflecting					
Confirm on previous δ camouflage detection	No	-	-	Yes	8
Ensure firing point is visible	No	-	-	Yes	5
Compare the β_p versus β	-	$\leq\beta_P$	$\beta = \beta_P$	$\geq\beta_P$	5
Compare the result of β versus β_0 (%)	-	$\leq\beta_0$	$\beta = \beta_0$	$\geq\beta_0$	5

To verify self-correction algorithm using deep action learning, the targeted images were performed with the following details. Where the output of YOLOv3 detection is δ and ε is the result of bounding box, and $\delta, \varepsilon \in \mathbb{R}$. In addition to using scales or probabilities, we used weighting to ascertain the influence composition in deep action learning. The weights were intuitive in their settings, with a maximum total score of 50. The assessment indicators result in Table 1 can be represented as (5).

$$\beta = \sum_{i=1}^{8} n_i\omega_i + n_{i+1}\omega_{i+1} + \cdots + n_{i+6}\omega_{i+6} \quad (5)$$

As a function, the first cycle's plan can be stated as (6), where the planning is denoted by p_t, acting by a_t, observing by o_t, and reflecting by r_t.

$$p_t = \beta \wedge \delta \wedge \varepsilon \Longrightarrow a_t \quad (6)$$

If the p_t has satisfied the states by $\beta, \delta, \varepsilon$, it will proceed to the a_t procedure, which will include conditions such as (7).

$$a_t \neq 0 \Longrightarrow o_t \quad (7)$$

We can write (7) from (8), and the reflection value result is determined by o_t with binary properties,

$$\begin{cases} o_t = \beta \wedge \delta \Longrightarrow r_t; \quad \beta, \delta \neq 0 \\ r_t = o_t, \quad o_t \in \{0,1\} \end{cases} \quad (8)$$

If $r_t = 1$, the cycle will come to a halt; if $r_t = 0$, it will scroll to the next cycle to evaluate $\beta_{(t+1)}$ and return its value. When observation o_t receives inputs from value $\beta \wedge \delta$ in (8), deep action learning works a second time to assess if the camouflaged target has been detected or not, and the cycle continues.

3.2. The system limitation

The detailed deep action learning for troop camouflage is limited to detecting for the trained classes of 1249, and the cycle number episodes in deep action learning cannot be predicted. In this paper, the number of cycles is natural (unlimited) and depends on deep action learning performance. We fulfilled this because it does not involve hardware such as rifles, cannons, or other battle equipment, but there will be a limit on the cycles number if later apply hardware. Additional explanations are discussed in section 3.

4. RESULTS AND DISCUSSION

Testing the detection results of troops camouflaged in the forest battlefield using a deep action learning approach was conducted separately. This separate test can be understood comprehensively considering that the process combines detection using deep learning YOLOv3 while self-correction used action learning. Three results will be presented: first, evaluating the detection results with a comparison of the optimizer, then evaluating the performance of action learning, and evaluating deep action learning.

4.1. Results

4.1.1. Camouflage targets detection

The testing was conducted separately, and Figure 2 shows the results of camouflage detection with the SGDM optimizer in various viewing angles. Figure 2(a) rotating target -0.2° can detect three targets with confidence values of 0.804, 0.785, and 0.807, respectively. The next confidence value when the target is rotated -0.1° experiences a significant increase as presented in Figure 2(b); for example, from 0.804 as shown in Figure 2(c), it increases to 0.899. However, as shown in Figure 2(d), there is a downward trend and increases again when rotated +0.2°, as can be seen in Figure 2(e). Through a series of experiments, the optimal rotation limit was -1°<θ <1°, over this limit, the recall was not quite perfect, as shown in Figure 2(f).

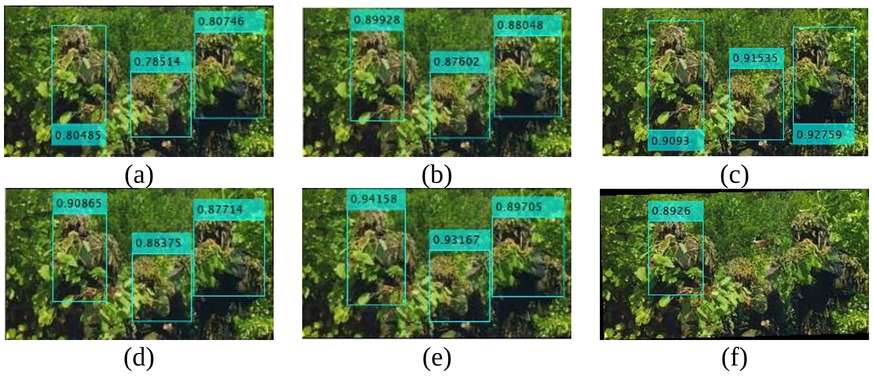


Figure 2. Camouflage detection results (a) rotated image $\theta - 0.2^\circ$, (b) rotated image $\theta - 0.1^\circ$, (c) original image $\theta - 0^\circ$, (d) rotated image $\theta + 0.1^\circ$, (e) rotated image $\theta + 0.2^\circ$, and (f) rotated image $\theta + 1.5^\circ$

As seen in the value of $> \theta \pm 1^\circ$, the detection results did not show a significant improvement. However, it does not mean that the value of $(\theta + t) \approx 0^\circ$ was the best. This assumption needed to be confirmed with various approaches, as manifested in Figure 2(c), the second bounding box toward Figure 2(d). Let us observe the differences in inconsistent perspectives; for instance, for a certain angle, the value of $\theta > (\theta + t)$, and the result is not always better. It means that through this stage, a decision-maker needs another method to adjust image rotation to get an optimal result, and action learning will work for this purpose. Another argument is that the optimizer of a detector has a significant role as well; and it needs to be disclosed. Figure 3 shows a comparison of the detection results of the three optimizers; SGDM, RMSProp, and ADAM are shown in red, green, and blue, respectively. Prior to that phase, the three optimizers were trained to generate each detector with SqueezeNet. Parameter details during training using initial learning rate 0.0001, mini-batch size of 16, maximum epochs of 200, and verbose frequency of 30. After being tested on one of the same image inputs, the results are presented in Figure 3. Figure 3(a) shows an original image with $\theta - 0^\circ$ detected by RMSProp optimizer, while Figure 3(b) shows rotated image $\theta - 0.1^\circ$ detected by the same optimizer. Figures 3(c) and 3(d) are detected by ADAM optimizer rotated image $\theta - 0^\circ$, $\theta - 0.1^\circ$ with confidence values of 0.980 and 0.971, respectively. If compared between Figure 3(e) original image $\theta - 0^\circ$ detected by SGDM optimizer and Figure 3(f) rotated $\theta - 0.1^\circ$, Figure 3(f) is significantly better with a confidence value of 0.960.

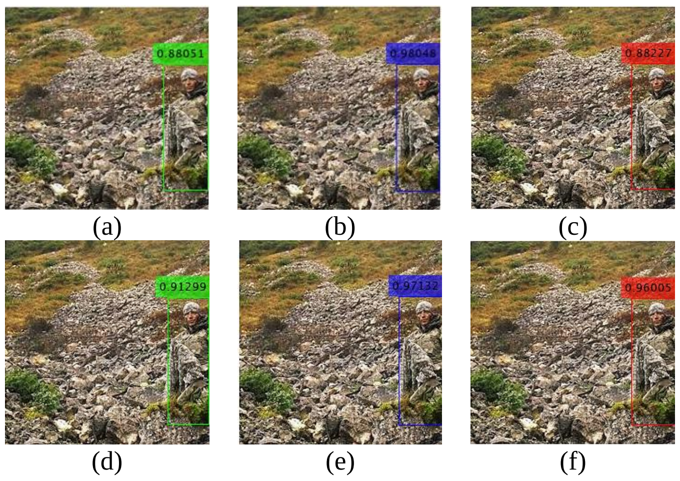


Figure 3. Troops camouflage detection results with different optimizers without rotating image. (a) original image $\theta - 0°$ detected by RMSProp optimizer, (b) rotated image $\theta - 0.1°$ detected by RMSProp, (c) original image $\theta - 0°$ detected by ADAM optimizer, (d) rotated image $\theta - 0.1°$ detected by ADAM optimizer, (e) original image $\theta - 0°$ detected by SGDM optimizer, and (f) rotated image $\theta - 0.1°$ detected by SGDM optimizer

Figure 3 shows that the ADAM optimizer has a reasonably good detection result for images that fused with the background. On the other hand, the SGDM optimizer has the lowest. The detection results presented in Figure 3 are pure detection results from a YOLOv3 detector on an image repeated three times without any interference from other methods such as rotating, zooming in, and zooming out the image. Several tests, including those in Figures 3(c) and 3(d), have a saturation detection where the test value will stagnate at a particular value after several repetitions of the test as can be seen in Table 2. The comparison between optimizer performance is significantly different for each optimizer as we have italicized. It means that the probability of testing remains to be valid in this case, although sometimes it is not significant.

Table 2. Bechmarking of optimizer's performance

Optimizers	Parameters						
	Confidence	Accuracy	Precission	Recall	F1	mAP	Time (s)
RMSProp	0.80	*0.97*	0.99	*0.99*	*0.99*	*0.99*	0.40
SGDM	*0.86*	0.92	*1*	0.92	0.96	0.88	*0.39*
ADAM	0.82	0.94	*1*	0.94	0.97	0.93	0.41

Now, we focus on the test results; Table 2 shows 25% of the tested samples from the dataset or about 288 of 1,153 images. The SGDM optimizer in terms of confidence value is 0.86 or superior to other optimizers. However, in terms of accuracy, precision, recall, performance, and mean average precision (mAP), RMSProp is superior while the SGDM detection time is 0.01 seconds or 0.39 seconds faster. On the other hand, the ADAM and SGDM optimizers have perfect precision of 1.

4.1.2. Deep action learning for self-correction detecting

It is critical to understand the principles of the existing general learning approaches to adopt deep action learning in artificial intelligence, and it must be stressed that action learning is different from other learning approaches. While there are several syntactical similarities between RL, active learning, experimental learning, and metacognitive learning, the process for instant planning, acting, assessing, reflecting, evaluating, or reviewing are different. Dick *et al.* and Altricther *et al.* were the first to introduce action learning [26], [27], [30] in general, while Aldridge, Bell, Norton, Mc Niff, Stringer *et al.* Whitehead, and others modified it and called it classroom action research [28], [31]–[33]. It resulted in no scholarly articles in engineering about the development of deep action learning in machine learning, and its application remains restricted to the educational field [28], [31]–[33].

Deep action learning consists of deep learning and action learning; let us go into detail about action learning separately. The proposed action learning is similar with that offered by Dick *et al.* and Altricther *et al.* consisted of four cycles. The first cycle is the planning step, where the input image is observed from the perception of =0° to be detected using YOLOv3. In this status, the output of YOLOv3 is provided as input for planning. Next on the acting step, the possible value of the highest confidence level will be observed by using image playback, selecting the type of optimizer, and providing alternative shooting points from the detection results. Followed by the third step, observing, in which this section compares the interim detection results to the passing grade. The results of the comparison at the observing step are delivered to the reflecting section to make decisions. If the detection value or confidence level being compared has not reached the passing grade, the cycle is repeated until it meets the passing grade value. The overview ofthe entire series of deep action learning procedures is presented in Figure 4. We try to evaluate the same image in Figure 4(a) with an original input $\theta + 0°$. After deep action learning was run for the first cycle with the detection results using the ADAM optimizer as can be seen in Figure 4(b), the result was unable to detect. Deep action learning rolled to the second cycle with the SGDM optimizer $\theta - 1°$ as shown in Figure 4(c) with a confidence value of 0.6441; Figure 4(d) showed a slight increase when using the RMSProp optimizer, and the input was rotated $\theta - 0.2°$. When we tried to detect in the fourth cycle of Figure 4(e) with the ADAM optimizer and the angle at $\theta - 0.1°$, the result dropped to 0.6051, whereas when the SGDM optimizer was used again with the angle at $\theta + 0.2°$, as can be seen in Figure 4(f), the confidence level increased and continued to do so as presented in Figure 4(g) using RMSProp optimizer at $\theta - 0.1°$. Finally, Figure 4(h) with RMSProp optimizer and angle $\theta + 0.1°$ was saturated at 0.9148 was considered as the best at that time.

As shown in Figure 4, the results of deep action learning showed a variable increase that cannot be predicted on the way to reaching the specified passing grade = 0.9. From 0.64 to 0.91, it took seven cycles, and detection speed took 1.14 seconds. Optimizer settings and rotation angle selection were purely made by the system that had been built, which will be discussed in the discussion subsection.

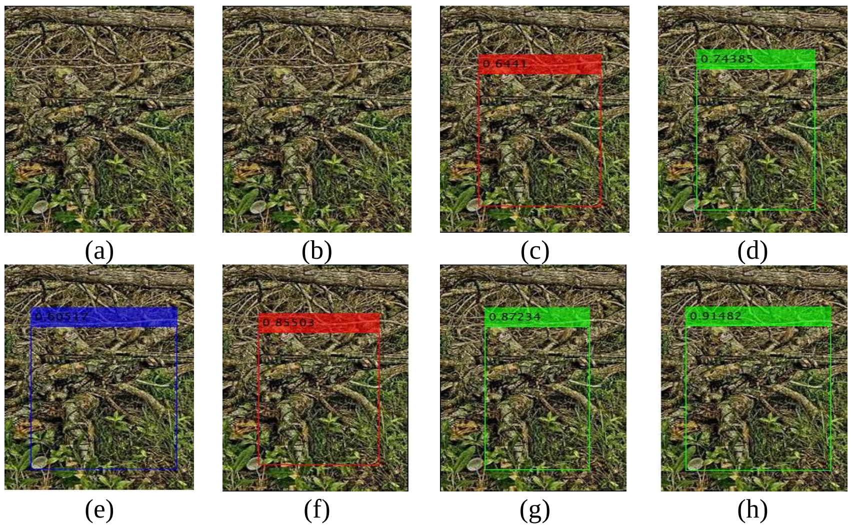


Figure 4. Sequence of camouflage detection using deep action learning (a) an original input $\theta + 0°$, result of detecting with, (b) ADAM optimizer, (c) SGDM optimizer $\theta - 0.1°$, (d) RMSProp optimizer $\theta - 0.2°$, (e) ADAM optimizer $\theta - 0.1°$, (f) SGDM optimizer $\theta + 0.2°$, (g) RMSProp optimizer $\theta - 0.1°$, (h) RMSProp optimizer $\theta + 0.1°$

4.2. Discussion

4.2.1. Detection targets in camouflaged object (CAMO) dataset

The detection results for the experiments to detect objects was similar to the background from CAMO dataset with some extended data. After being observed, significant differences were found in the detection results, for example, bright and dark images. There is a tendency for dark images to have low confidence values. Figure 5 depicts the confidence value of the histogram. Therefore, the bounding box alone is not enough, and it is necessary to add the centroid of the target for target aiming purposes. If only the results of the confidence value are pure, the detection results with deep learning are less valuable.

Calculating the centroids from the bounding box, as presented in (9), is the simplest method.

$$B_{box} = \begin{bmatrix} n_{11} & n_{12} & n_{13} & n_{14} \\ n_{21} & n_{22} & n_{23} & n_{24} \\ n_{31} & n_{32} & n_{33} & n_{34} \\ \vdots & \vdots & \vdots & \vdots \\ n_{n1} & n_{n2} & n_{n3} & n_{n4} \end{bmatrix} \tag{9}$$

The first line in the (9) matrix shows the first bounding box and the second line is the second bounding box. If $n_{11,}$ $n_{12,}$ are denoted for X_c, Y_c while $a = n_{13}$ and $b = n_{14}$. The bounding box matrix B_{box} comprises of four columns a$_{[1\ldots,4]}$ and the number of rows depends on the number of detected targets $n_{[n,4]}$ on each coordinate. So, we could find the centroids (X_{cen}, Y_{cen}) from (9) as follows.

$$\begin{cases} X_{cen} = X_c + \frac{a}{2} \\ Y_{cen} = Y_c + \frac{b}{2} \end{cases} \tag{10}$$

The centroid can be determined using (10) and serves as the target's firing reference point. The centroid point in this state was still in the 2D, as for the detection results obtained using (9) and (10) can be seen in Figure 5. Figures 5(a) and 5(b) show troop camouflage with more than one target, while Figures 5(c) and 5(d) show one target camouflage only. Observed from the histogram side, the top two images represented in Figures 5(e) and 5(f) are images with dark dominance, while the second bottom two images shown in Figures 5(g) and 5(h) had a relatively normal histogram distribution. The two groups of images could show empirical evidence that the tendency for the normal curve to dominate was lower in detection accuracy even with the same optimizer.

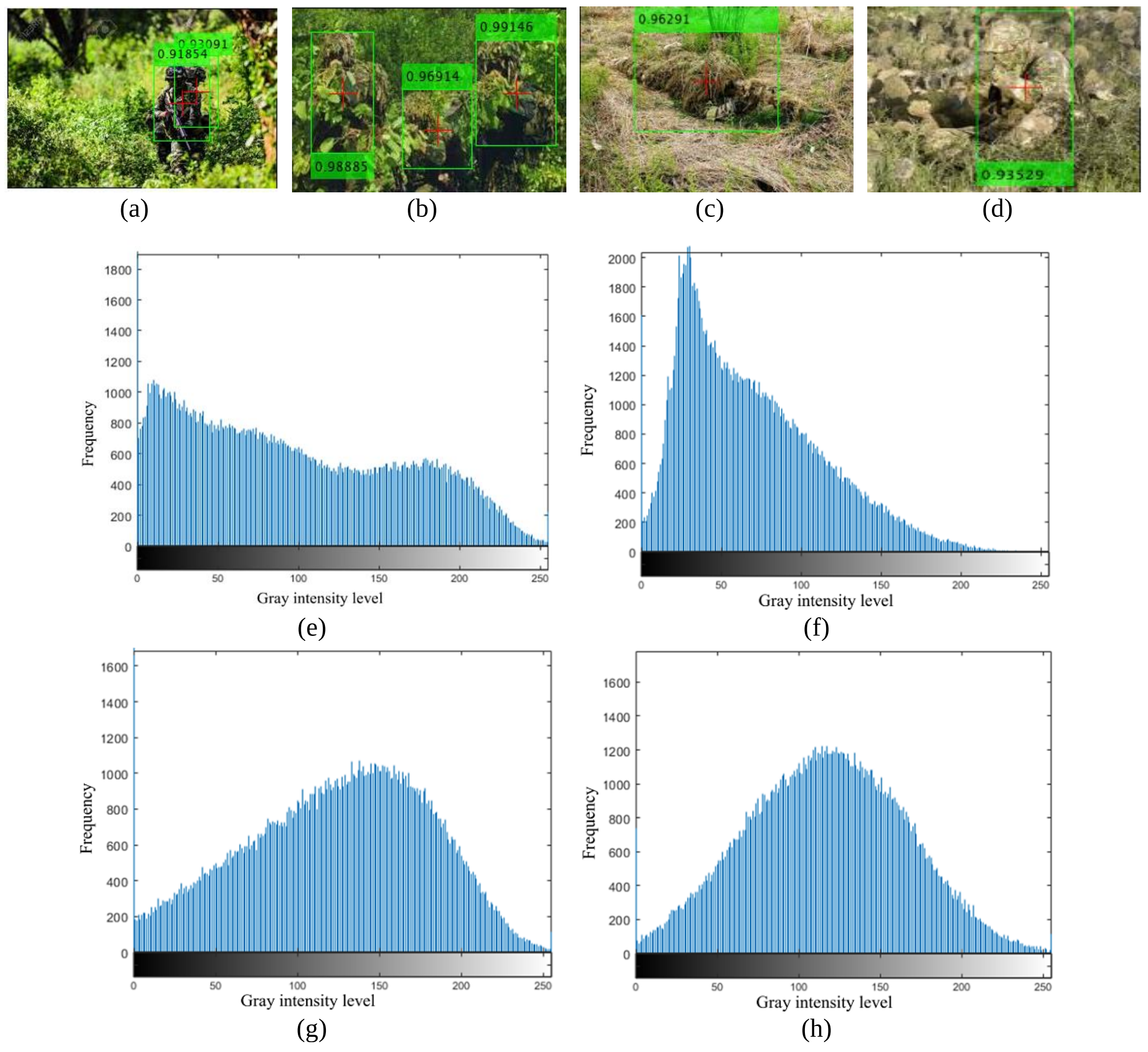


Figure 5. Shooting targets based on ADAM optimizer; (a)-(b) a dark input images, (c)-(d) a balance input image with each image hitogram distribution. The dark image (e)-(f) tends to have a left skewed histogram while the balanced image (g)-(h) has a normal curve and the detection rate results tend to be stable

4.2.2. Deep action learning evaluation

Before starting with the target detection technique and localization for firing target, the parameter settings for YOLOv3 as a deep learning approach should be understood. The results of deep learning were undoubtedly influenced by the differences in parameters. The starting mini-batch size, learning rate, and the maximum epoch used will all hold a major impact on detection accuracy and training duration. The training results on the three optimizers (SGDM, ADAM, and RMSProp) are presented in Figure 6. If the learning rate is too low, for example, training may take a longer process. If the learning rate is too high, however, the training may provide a suboptimal or diverge output. The centroid can be determined using (10) and serves as the target's firing reference point.

A detector constructed during training can be seen in general performance based on the training loss over iteration numbers. The training loss of the YOLOv3 detector is depicted in Figure 6 using the SGDM, ADAM, and RMSProp optimizers. In contrast to the other two, RMSProp was found to be the best, as seen in Figure 6. The value of training loss was virtually nil in the 500th iteration and tended to stagnate until the 600th iteration, while the ADAM and SGDM optimizers were close to each other in the 800th iteration. This detector's precision was critical for overall system testing verification. At all recall levels, the precision should ideally be one.

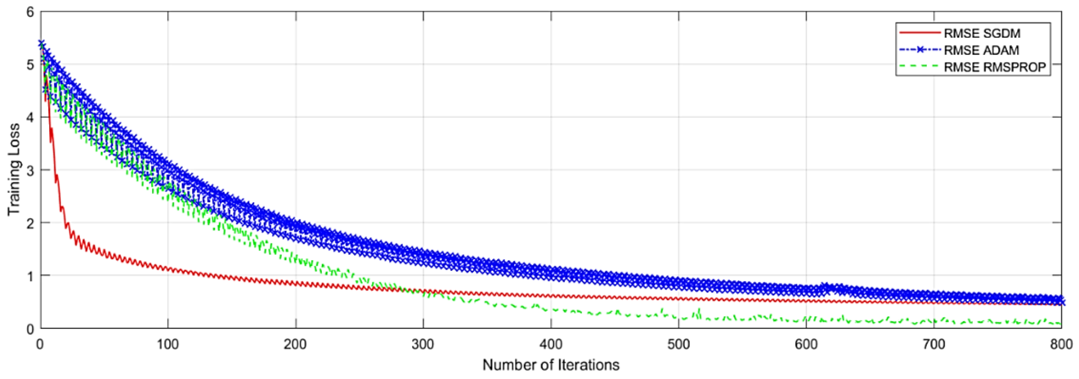


Figure 6. RMSEs of training loss during iteration in a YOLOv3 detector; training loss for SGDM optimizer (solid line), ADAM optimizer (dashed-cross line), and RMSProp optimizer (dashed line)

Comparisons were also made to compare deep action learning with other methods. As presented in Table 3, the system can perform self-correction to achieve a minimum passing grade β_0. The duration of passing grade achievement varies according to the cycles, and of course, the more cycles, the longer the time required. We set $\beta_0 = 0.88$ with the developed algorithm and were able to detect as seen in the second row of Table 3 quickly; the detection result was 0.97, meant that it did not require self-correction using deep action learning. Meanwhile, self-correction dynamics occurred in the first, third, fourth, and fifth rows with each detection result. Focusing on the first row, shows in Table 3 with $\theta = 0.2°$ was unable to detect the target in the first cycle, and in the second cycle, the target was detected at 0.79 and 0.92. Deep action learning tried to find another optimizer alternative using RMSProp, and the result was able to exceed β_0 with a result of 0.94. The fourth line was almost similar with the first line, while the last line had two cycles with the same optimizer in detecting improvements with 0.58 seconds for both cycles.

In this paper, RMSProp was found as the best optimizer, as shown in Figure 6, but we only use it as a single option. The selection of detectors was completely determined by deep action learning through the assessment mechanism in Table 1. Due to this reason, we need to compare the results of camouflage detection using deep action learning with other methods such as Unet++, CPD, SINet, and MirrorNet, which at least use CAMO dataset.

Obviously, the deep action learning annotation procedure for determining the firing target will be more accurate than the existing methods, as presented in Table 4. We compare SqueezeNet's performance to state-of-the-arts stated in [34] to establish a fair comparison. Table 4 compares the E-measure (Eφ) [35], S-measure (Sα) [36], weighted F-measure (F_β^ω) [37], and MAE performance of several approaches [38]. As can be seen, the strategies introduced recently tend to produce superior results. Deep action learning with SqueezeNet, our suggested method, achieved the greatest results in terms of Eφ, Sα, F_β^ω, and MAE. In every metric, deep action learning with the SqueezeNet backbone outperformed state-of-the-art approaches by a significant margin.

Table 3. Self-correction process sequence on camouflage detection using deep action learning

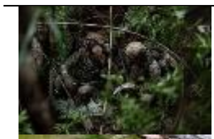
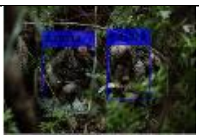
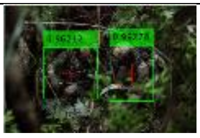
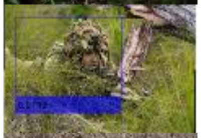
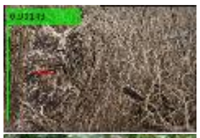
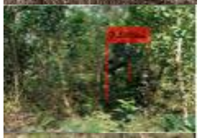
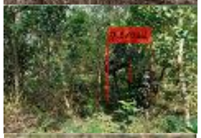
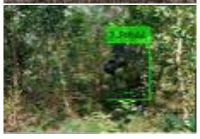
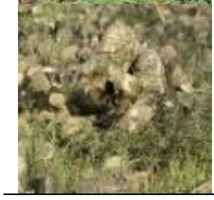
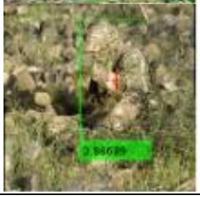
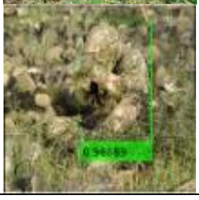
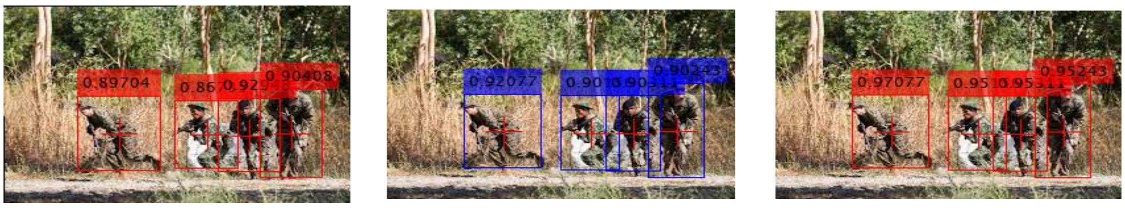
Input ($\theta = 0°$)	$\boldsymbol{C_{n+1}}$	$\boldsymbol{C_{n+2}}$	$\boldsymbol{C_{n+3}}$	$\boldsymbol{\theta}$	$\boldsymbol{\beta_0 \mid \beta_n}$	**T (s)**
				$-0.2°$ $-0.1°$ $0.1°$	n/a 0.79 | 0.92 0.96 | 0.94	1.03
		na	na	$0.1°$ - -	na 0.97 -	0.35
				$0°$ $0.1°$ $0.2°$	na na 0.76	1.12
				$-0.1°$ $0.0°$ $0.1°$	0.87 0.67 0.88	1.04
			na	$-0.2°$ $0.1°$ -	0.88 0.98 -	0.58

Table 4. Comparison of methods performance on CAMO dataset

Method	Year	Training Setting	Evaluation Metrics			
			$S_\alpha \Uparrow$	$E_\phi \Uparrow$	$F_\beta^\omega \Uparrow$	$MAE \Downarrow$
Unet++ [39]	2018	CAMO [38] + COD [34]	0.599	0.653	0.392	0.149
CPD [34]	2019	CAMO [38] + COD [34]	0.726	0.729	0.550	0.115
SINet [34]	2020	CAMO [38]	0.708	0.706	0.476	0.131
MirrorNet [9]	2020	CAMO [38]	0.741	0.804	0.652	0.100
SquezeeNet	2021	CAMO [38] + extended	*0.782*	*0.856*	*0.657*	*0.083*

4.2.3. Experiments of self-correction on deep action learning

In this subsection, we will discuss the self-correction process using deep action learning. As presented in the preceeding subsection, in Figure 5, the detection results held values that were upgraded by the system. Self-correction uses deep action learning, while the passing grade was set at β_0=0.95 in Figure 7(a) at the beginning of the system detected the target using SGDM optimizer which resulted in 0.89, 0.86, 0.92, and 0.90 at $\theta + 0.2°$, because it failed to reach β_0=0.95, the system wastried to be updated this time using the ADAM optimizer at $\theta + 0°$ as shown in Figure 7(b), and the detection results were 0.92, 0.90, 0.90, and 0.90. The cycle was added from cycle two to cycle three, where this time the input image was set at $\theta - 0.1°$ and back to SGDM optimizer and now successfully passed β_0 as shown in Figure 7(c), which was set with detection results of 0.97, 0.95, 0.95, and 0.95 for each target.



(a) (b) (c)

Figure 7. A sequence of self-correction in deep action learning, firstly, detection using SGDM optimizer (a), changed to ADAM optimizer (b) and switched back to SGDM optimizer (c) with internal assessment

The exciting notion about self-correction using deep action learning is that the shift in the bounding box localization results is insignificant. The target firing point is approximately the same if compared to the optimizer and viewing angle variations. In fact, Figure 7 structurally has a histogram pattern like Figures 5(a) to (c) with a relatively even distribution of histograms. A quarter of the approximately 1249 images trained in deep action learning utilizing CAMO datasets was used for testing in this study. However, in addition to the testing of CAMO dataset, accuracy is beyond the scope of this paper.

5. CONCLUSION

This study has established an effective deep action learning for troop camouflage recognition detection in the CAMO dataset. A deep action learning designed with deep learning (YOLOv3) and action learning can detect and make firing points camouflaged in 2D image workspace. Inside, the YOLOv3 is equipped with SqueezeeNet and modified the view angle on the input image driven by deep action learning. The processes of detecting troops in deep action learning include planning, acting, observing, and reflecting on whole steps without preprocessing. However, the results showed a value at 0.97 and 0.99 for accuracy and recall, respectively. Within a passing grade of 0.88, this evaluation mechanism calculated with an indefinite cycle in deep action learning. In the future, we intend to investigate the problem of camouflaged instance segmentation. For the experiment, we will improve video-based detection using YOLOv4 or YOLOv5 with a preprocessing approach.

ACKNOWLEDGEMENTS

Authors thank Biomedic and Artificial Intelligence Laboratory of Engineering Faculty, Universitas Negeri Yogyakarta for financial support.

REFERENCES

[1] H. Xiao, Z. Qu, M. Lv, Y. Jiang, C. Wang, and R. Qin, "Fast self-adaptive digital camouflage design method based on deep learning," *Appl. Sci.*, vol. 10, no. 15, Jul. 2020, doi: 10.3390/app10155284.

[2] H. Lu, X. Wang, S. Liu, M. Shi, and A. Guo, "The possible mechanism underlying visual anti-camouflage: a model and its real-time simulation," *IEEE Trans. Syst. Man, Cybern. - Part A Syst. Humans*, vol. 29, no. 3, pp. 314–318, May 1999, doi: 10.1109/3468.759290.

[3] Y. Shen, J. Li, W. Lin, L. Chen, F. Huang, and S. Wang, "Camouflaged target detection based on snapshot multispectral imaging," *Remote Sens.*, vol. 13, no. 19, Oct. 2021, doi: 10.3390/rs13193949.

[4] K. Huang, C. Li, J. Zhang, and B. Wang, "Cascade and fusion: A deep learning approach for camouflaged object sensing," *Sensors*, vol. 21, no. 16, Aug. 2021, doi: 10.3390/s21165455.

[5] J. Yu, G. Zhou, S. Zhou, and J. Yin, "A lightweight fully convolutional neural network for SAR automatic target recognition," *Remote Sens.*, vol. 13, no. 15, Aug. 2021, doi: 10.3390/rs13153029.

[6] W. Yu, X. Wang, P. Calyam, D. Xuan, and W. Zhao, "Modeling and detection of camouflaging worm," *IEEE Trans. Dependable Secur. Comput.*, vol. 8, no. 3, pp. 377–390, May 2011, doi: 10.1109/TDSC.2010.13.

[7] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *ArXiv160302199*, Mar. 2016, [Online]. Available: http://arxiv.org/abs/1603.02199.

[8] Y. Shen *et al.*, "Rapid detection of camouflaged artificial target based on polarization imaging and deep learning," *IEEE Photonics J.*, vol. 13, no. 4, pp. 1–9, Aug. 2021, doi: 10.1109/JPHOT.2021.3103866.

[9] J. Yan, T.-N. Le, K.-D. Nguyen, M.-T. Tran, T.-T. Do, and T. V Nguyen, "MirrorNet: Bio-inspired camouflaged object segmentation," *IEEE Access*, vol. 9, pp. 43290–43300, 2021, doi: 10.1109/ACCESS.2021.3064443.

[10] H. Shi, L. Shi, M. Xu, and K.-S. Hwang, "End-to-end navigation strategy with deep reinforcement learning for mobile robots," *IEEE Trans. Ind. Informatics*, vol. 16, no. 4, pp. 2393–2402, Apr. 2020, doi: 10.1109/TII.2019.2936167.

[11] C.-Y. Tsai, Y.-S. Chou, C.-C. Wong, Y.-C. Lai, and C.-C. Huang, "Visually guided picking control of an omnidirectional mobile manipulator based on end-to-end multi-task imitation learning," *IEEE Access*, vol. 8, pp. 1882–1891, 2020, doi: 10.1109/ACCESS.2019.2962335.

[12] A. Caglayan and A. B. Can, "Volumetric object recognition using 3-D CNNs on depth data," *IEEE Access*, vol. 6, pp. 20058–20066, 2018, doi: 10.1109/ACCESS.2018.2820840.

[13] S. Chen, M. Wang, W. Song, Y. Yang, Y. Li, and M. Fu, "Stabilization approaches for reinforcement learning-based end-to-end autonomous driving," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 4740–4750, May 2020, doi: 10.1109/TVT.2020.2979493.

[14] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with AutoRL," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2007–2014, Apr. 2019, doi: 10.1109/LRA.2019.2899918.

[15] G. Yang, R. Zhu, Z. Fang, C.-Y. Chen, and C. Zhang, "Kinematic design of a 2R1T robotic end-effector with flexure joints," *IEEE Access*, vol. 8, pp. 57204–57213, 2020, doi: 10.1109/ACCESS.2020.2982185.

[16] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Trans. Emerg. Top. Comput.*, vol. 8, no. 2, pp. 517–532, Apr. 2020, doi: 10.1109/TETC.2017.2740364.

[17] Y. Zou and R. Lan, "An end-to-end calibration method for welding robot laser vision systems With deep reinforcement learning," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 7, pp. 4270–4280, Jul. 2020, doi: 10.1109/TIM.2019.2942533.

[18] V. Digani, L. Sabattini, and C. Secchi, "A probabilistic eulerian traffic model for the coordination of multiple AGVs in automatic warehouses," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 26–32, Jan. 2016, doi: 10.1109/LRA.2015.2505646.

[19] Y. Ibrahim *et al.*, "Soft errors in DNN accelerators: A comprehensive review," *Microelectron. Reliab.*, vol. 115, Dec. 2020, doi: 10.1016/j.microrel.2020.113969.

[20] Y. Kim, H. Kang, N. Suryanto, H. T. Larasati, A. Mukaroh, and H. Kim, "Extended spatially localized perturbation GAN (eSLP-GAN) for robust adversarial camouflage Patches," *Sensors*, vol. 21, no. 16, Aug. 2021, doi: 10.3390/s21165323.

[21] H. Li, A. Abdelhadi, R. Shi, J. Zhang, and Q. Liu, "Adversarial hardware with functional and topological camouflage," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 68, no. 5, pp. 1685–1689, May 2021, doi: 10.1109/TCSII.2021.3065292.

[22] P. V. Mohan, S. Dixit, A. Gyaneshwar, U. Chadha, K. Srinivasan, and J. T. Seo, "Leveraging computational intelligence techniques for defensive deception: A review, recent advances, open problems and future directions," *Sensors*, vol. 22, no. 6, Mar. 2022, doi: 10.3390/s22062194.

[23] C. Brook and M. Pedler, "Action learning in academic management education: A state of the field review," *Int. J. Manag. Educ.*, vol. 18, no. 3, Nov. 2020, doi: 10.1016/j.ijme.2020.100415.

[24] R. Van Gasse, K. Vanlommel, J. Vanhoof, and P. Van Petegem, "Teacher interactions in taking action upon pupil learning outcome data: A matter of attitude and self-efficacy?," *Teach. Teach. Educ.*, vol. 89, Mar. 2020, doi: 10.1016/j.tate.2019.102989.

[25] O. Serrat, "Action learning," in *Knowledge Solutions*, Singapore: Springer Singapore, 2017, pp. 589–594.

[26] B. Dick, E. Stringer, and C. Huxham, "Theory in action research," *Action Res.*, vol. 7, no. 1, pp. 5–12, Mar. 2009, doi: 10.1177/1476750308099594.

[27] H. Altrichter, S. Kemmis, R. McTaggart, and O. Zuber-Skerritt, "The concept of action research," *Learn. Organ.*, vol. 9, no. 3, pp. 125–131, Aug. 2002, doi: 10.1108/09696470210428840.

[28] L. M. Bell and J. M. Aldridge, *Student voice, teacher action research and classroom improvement*. Rotterdam: SensePublishers, 2014.

[29] W. Fang, L. Wang, and P. Ren, "Tinier-YOLO: A real-time object detection method for constrained environments," *IEEE Access*, vol. 8, pp. 1935–1944, 2020, doi: 10.1109/ACCESS.2019.2961959.

[30] B. Dick, "Action research literature 2006—2008," *Action Res.*, vol. 7, no. 4, pp. 423–441, Dec. 2009, doi: 10.1177/1476750309350701.

[31] J. Whitehead and J. McNiff, *Action research living theory*. London: Thousand Oaks: SAGE Publications, 2006.

[32] L. Norton, "Action research in teaching and learning: a practical guide to conducting pedagogical research in universities." 2019, Accessed: Jun. 11, 2021. [Online]. Available: http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1926353.

[33] E. T. Stringer, L. M. Christensen, and S. C. Baldwin, *Integrating teaching, learning, and action research: enhancing instruction in the K-12 classroom*. Thousand Oaks, Calif: Sage, 2010.

[34] D.-P. Fan, G.-P. Ji, G. Sun, M.-M. Cheng, J. Shen, and L. Shao, "Camouflaged object detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 2774–2784, doi: 10.1109/CVPR42600.2020.00285.

[35] D.-P. Fan, C. Gong, Y. Cao, B. Ren, M.-M. Cheng, and A. Borji, "Enhanced-alignment measure for binary foreground map evaluation," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Jul. 2018, pp. 698–704, doi: 10.24963/ijcai.2018/97.

[36] D.-P. Fan, M.-M. Cheng, Y. Liu, T. Li, and A. Borji, "Structure-measure: A New Way to Evaluate Foreground Maps," *ArXiv:170800786*, Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.00786.

[37] R. Margolin, L. Zelnik-Manor, and A. Tal, "How to evaluate foreground maps," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 248–255, doi: 10.1109/CVPR.2014.39.

[38] T.-N. Le, T. V Nguyen, Z. Nie, M.-T. Tran, and A. Sugimoto, "Anabranch network for camouflaged object segmentation," *Comput. Vis. Image Underst.*, vol. 184, pp. 45–56, Jul. 2019, doi: 10.1016/j.cviu.2019.04.006.

[39] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A nested U-net architecture for medical image segmentation," in *DLMIA 2018, ML-CDS 2018: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, 2018, pp. 3–11.

BIOGRAPHIES OF AUTHORS

Muslikhin is robotics and artificial intelligent researcher who expertreceived. His B.S. and M.S. degrees in electronics engineering education, and technology and vocational education from Universitas Negeri Yogyakarta, Yogyakarta, Indonesia in 2011 and 2013, respectively. In 2021, he received his Ph.D. in electrical engineering from Southern Taiwan University of Sciences and Technology in Tainan, Taiwan. Robotics, machine vision, and artificial intelligence are among his research interests. In addition, he took first place in the TIRT International Innovative Robotics Festival in Taoyuan City, Taiwan, in 2020. The team used an AIoT robotic to help quarantine COVID-19 patients in this competition. He can be contacted at email: muslikhin@uny.ac.id.

Aris Nasuha is a machine learning researcher with a focus on research and development. He has received a B.Sc. (physics) from Universitas Gadjah Mada, M.Eng. and Dr. (electrical engineering) from Institut Teknologi Sepuluh Nopember, Indonesia respectively. His research interest includes digital signal processing, digital image processing and machine learning. Now he is head of Electronics Engineering Department at Vocational School Program, Universitas Negeri Yogyakarta, Indonesia. He can be contacted at email: arisnasuha@uny.ac.id.

Fatchul Arifin is received a B.Sc. in electric engineering at Universitas Diponegoro in 1996. Afterward, He received master's degree from the department of electrical engineering, Institutut Teknologi Bandung, Indonesia in 2003 and doctoral degree in electric engineering from Institut Teknologi Sepuluh November, Surabaya, Indonesia in 2014. Currently he is the lecturer at both undergraduate and postgraduate electronics and informatic program at Universitas Negeri Yogyakarta, Indonesia. His research interests include but not limited to artificial intelligent systems, machine learning, fuzzy logic, and biomedical engineering system. He can be contacted at email: fatchul@uny.ac.id.

Suprapto received B.S. degree from the department of electrical engineering education, Universitas Negeri Yogyakarta in 2001. Afterward, he received M.S. degree from the department of electrical engineering, Gadjah Mada University in 2005. He received Ph.D. degree in control theory, electrical engineering, National University of Science and Technology, Taiwan in January 2018. Since 2005, he has been a lecturer at the Departement of Electronics Engineering, Yogyakarta State University, Yogyakarta, Indonesia. His current research interest is related to control theory, signal processing, and computational intelligence. He can be contacted at email: suprapto@uny.ac.id.

Anggun Winursito is a researcher with a focus on signal processing and instrumentation. He has received M.Eng. from Gadjah Mada University and B.Ed. from Universitas Negeri Yogyakarta in 2014 and 2018. His areas of expertise are speech recognition, image recognition, analog and digital signal processing, and instrumentation. He has participated in several local and national research projects including speech pattern recognition and radar telecommunications systems. He was awarded the best paper at International Conference on Smart Computing and Electronic Enterprise (ICSCEE) held in Malaysia for his research on the compression of speech features to improve the accuracy of speech recognition systems. He can be contacted at email: anggunwinursito@uny.ac.id.