Received August 7, 2018, accepted September 17, 2018, date of publication September 24, 2018, date of current version October 17, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872025

Autoencoder-Combined Generative Adversarial Networks for Synthetic Image Data Generation and Detection of Jellyfish Swarm

KYUKWANG KIM, (Student Member, IEEE), AND HYUN MYUNG, (Senior Member, IEEE)
Urban Robotics Laboratory, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea

Corresponding author: Hyun Myung (hmyung@kaist.ac.kr)

This work was supported in part by the ICT R&D Program of MSIP/IITP (Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion) under Grant 2016-0-00563 and in part by the National Research Foundation of Korea (NRF) Grant through the Korean Government (MSIP) under Grant NRF-2015R1A2A1A15056262.

ABSTRACT Image-based sensing of jellyfish is important as they can cause great damage to the fisheries and seaside facilities and need to be properly controlled. In this paper, we present a deep-learning-based technique to generate a synthetic image of the jellyfish easily with autoencoder-combined generative adversarial networks. The proposed system can easily generate simple images with a smaller number of data sets compared with other generative networks. The generated output showed high similarity with the real-image data set. The application using a fully convolutional network and regression network to estimate the size of the jellyfish swarm was also demonstrated, and showed high accuracy during the estimation test.

INDEX TERMS Autoencoder, generative adversarial networks, jellyfish swarm, fully convolutional network, regression.

I. INTRODUCTION

Jellyfish rapidly proliferate and form a giant swarm in a nutrient-rich environment [1]. Recent changes in the ocean environment, such as global warming and pollution, generated conditions conducive to the jellyfish swarm by decreasing the potential predators of the jellyfish [2]. The huge population of jellyfish has caused great damage to various industries. Fisheries suffer greatly as the jellyfish swarm damages fishing grounds, aqua farms, and fishing tools such as nets [3]. Seaside power plants and resorts also spend a significant amount of their budget to remove jellyfish swarms. In the Republic of Korea, more than 500 cases of human injury and financial damages estimated to be about 135–280 million US dollars occur annually [4]. Many engineering solutions were developed to detect and remove jellyfish swarms. A screening method to clear jellyfish from the cooling water intake channel of the seaside power plants was developed [3]. Some researchers developed a jellyfish-removing robotic system to detect jellyfish population with a quadcopter platform and removing them with co-operating robotic boats [5], [6].

Image recognition plays a key role in the robotic jellyfish-control system. Detecting the moving jellyfish swarm and coordinating the robots to the place where the jellyfish are found allows the efficient and appropriate operation of the robotic system [5]. In addition, the detection system could be used as an alerting system for seaside resorts. Deep architectures used in these image-recognition systems have shown great advances in recent years, accomplishing several improvements in various tasks [7]. However, deep learning requires a large amount of quality-controlled data and manually labeled ground-truth answers to be trained. The jellyfish tend to be sensitive to the temperature and salinity of the water; it is difficult to predict the appearance of the jellyfish and collect the image data for the deep-learning system. Labeling and classification of the collected image data is also a labor-intensive and challenging job. To solve this issue, a generative model will be exploited to generate the image data for training image sensors in this paper.

Generative Adversarial Networks (GAN), an implicit generative model proposed by Goodfellow *et al.* [8], has attracted the most attention recently. The key feature of the GAN architecture is the two sub-architectures it is composed of the generative network that generates the image from the seed vector and the discriminator network that classifies the generator-synthesized fake images from the real images.

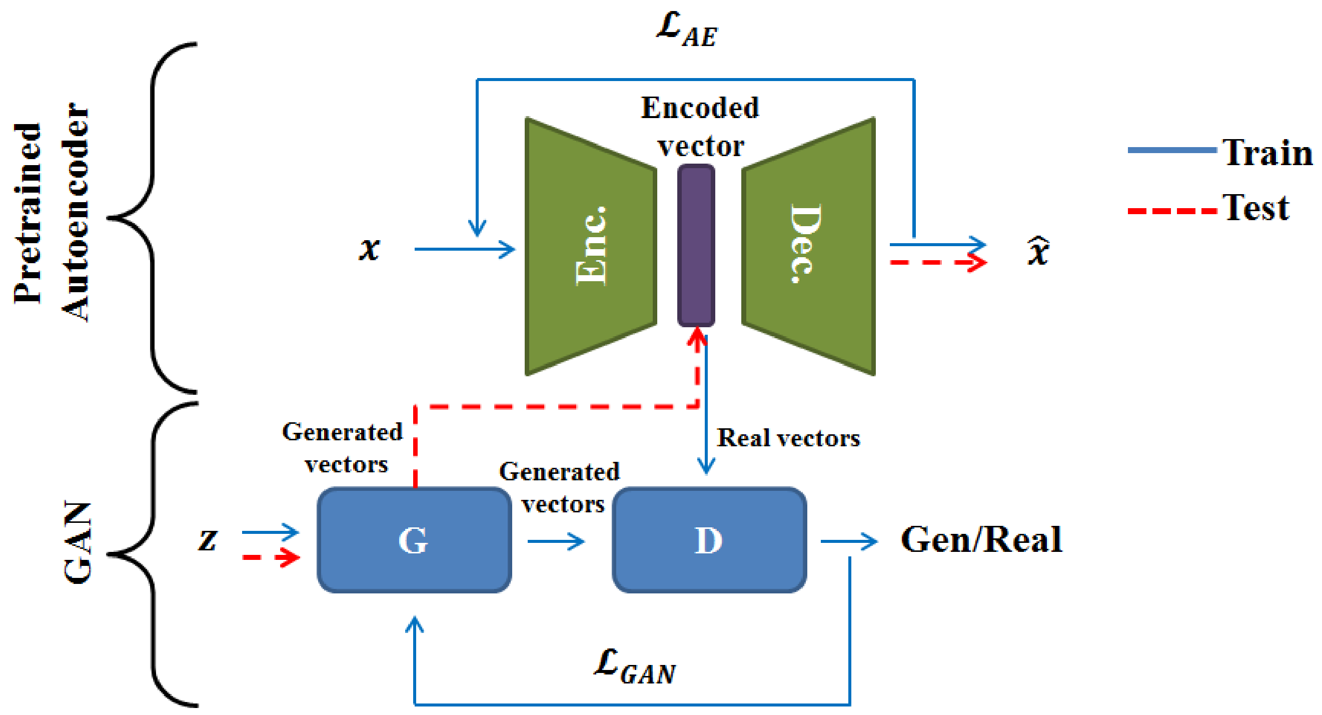

FIGURE 1. Overall architecture of the proposed image-generation model. The blue solid line shows the dataflow at the train phase. The red dotted line only works at the test phase.

These two networks finally optimize the output images as they compete with each other during the training phase; the generator tries to maximize its acceptance ratio (create a realistic image), while the discriminator attempts to minimize the generator output (find every fake image). However, this min-max saddle-point optimization process easily fails and makes it difficult to accomplish the training process of the GAN. Many GAN architectures such as Unrolled-GAN [9] have been proposed to increase the GAN's stability; however, it is difficult to directly apply these complex structures to previously established systems, or they require a large volume of training data to update the deep structures properly. Research conducted with benchmark datasets such as the MNIST does not suffer from a lack of the data; however, it works as a big entry barrier for applications based on custom datasets. Though the purpose of the GAN is to increase the number of data, a contradictory situation, that GAN requires large number of data occurs.

In this article, we propose a modified GAN structure to reduce the level of image synthesis difficulty. The overall structure of the proposed AE-combined GAN is shown in Fig. 1. We focused on the fact that the autoencoders (AEs) could project a large matrix (image) into a smaller dimension by encoding it. The pre-trained autoencoder-encoded vectors of the real images are trained to the GAN at the train phase and GAN-generated vectors (generated vectors in Fig. 1) are converted to the final image output by the decoder part of the autoencoder. Then the GAN can synthesize new images by generating only short vectors (test phase of Fig. 1), which is much easier than composing entirely new images. The next section also describes how the proposed system can be used to train the image sensor for jellyfish detection in the marine environment.

II. AUTOENCODER-COMBINED GAN

A. GAN STRUCTURE AND IMPLEMENTATION

For the implementation, the original multilayer perceptron (MLP) GAN structure used by Goodfellow *et al.* was used. The TensorFlow-based GAN source code available in the Github project was used as a basis code [10]. Both the generator and discriminator of the GAN consist of three layers. The network structural settings of the used AE and GAN are illustrated in Fig. 2.

The learning rate (lr) of 0.001 without the decay (by setting the step size option larger than the maximum iteration) was used for the optimizer. The weight decay of 0.0005 and the momentum value of 0.9 were used (default values for the MNIST training). With these options, about 5,000 iterations were conducted to optimize the network. The loss per every 100 steps was monitored to check proper termination time. Other detailed options are available online at the Caffe Github [11], [12].

The encoded vectors (the enc3neuron in Fig. 2(a)) were used for the GAN training data as mentioned above. The vectors extracted from the sigmoid layer ('neuron' layer) showed the better results in the GAN training, and the enc3neuron values were used as it is the deepest neuron layer before the

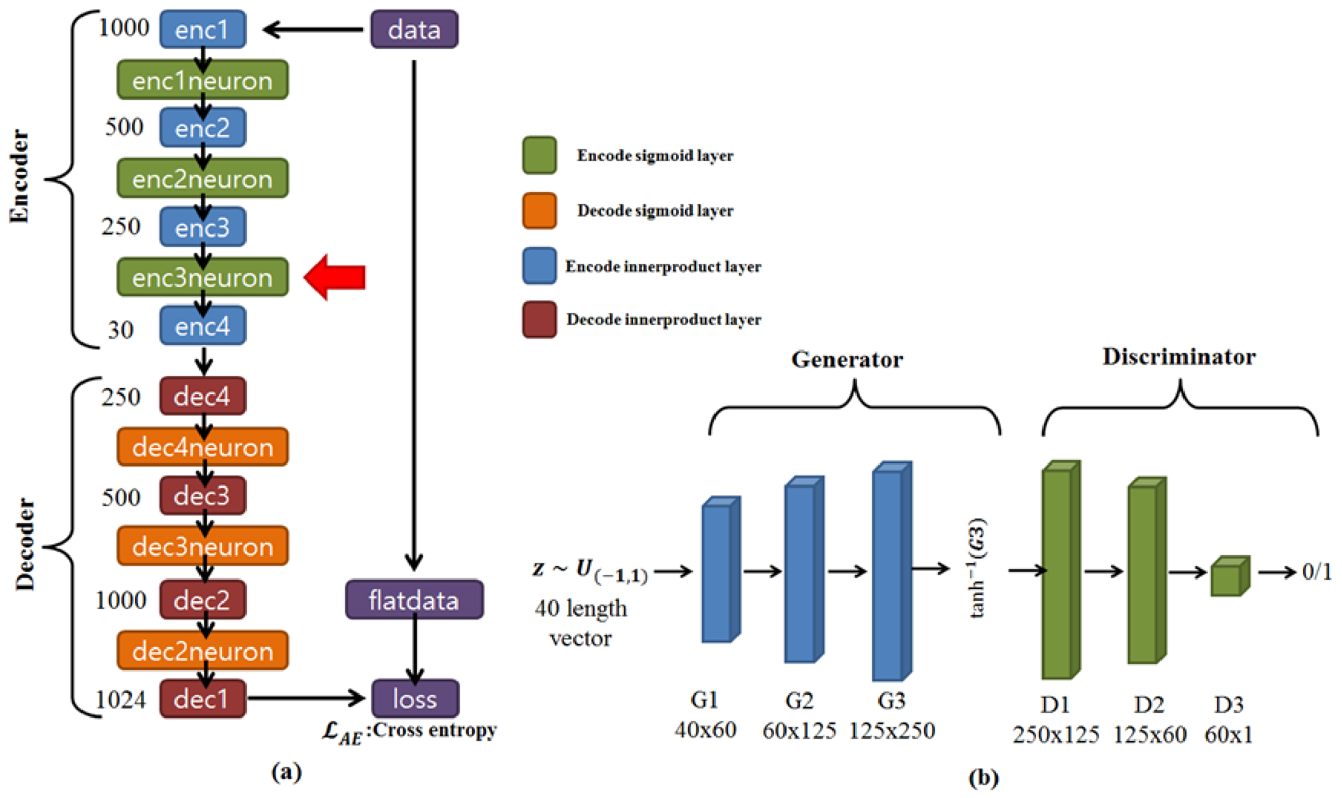

FIGURE 2. (a) Layer information of the used autoencoder (AE). The layers named with 'neuron' indicate the sigmoid layers and the others mean innerproduct (ip) layers. The number written next to the ip layer indicates the product size of the ip layer (e.g. enc2 is 1000×500). (b) Structure and dimension information of the used GANs. Big red arrow pointing enc3neuron indicates the layer where encoded vectors are extracted.

decoding starts. Before training, all training datasets were normalized to fit all values between -1 and 1. The Euclidean loss function combined with the RMSprop optimizer used in the LSGAN was applied [13]. The GANs generally use sigmoid-cross entropy losses to check a probability that with the given input, the discriminator is real or not. However as the generated outputs of the proposed GAN are size-reduced vectors instead of the image, we used the Euclidean loss, which measures the spatial distance from the training datasets for higher differentiability between the generated vectors and real vectors.

B. TRAINING IMAGE PREPARATION

The video of the jellyfish swarm proliferating in the sea near the cities of Changwon and Masan, Republic of Korea was recorded. The raw images of the ocean scene with the size of 1080×980 were collected by the Logitech c920r camera attached to the Unmanned Aerial Vehicle (UAV) hovering about 4 to 5m from the water surface. At this altitude, the adult Aurelia aurita, the jellyfish species which is the main cause of the jellyfish bloom, fits within the 32×32 square image patches. One hundred jellyfish image patches [16] were manually cropped and classified from the captured video. The background seawater image of each patch was removed and changed into a plain black background. All jellyfish images were converted into grayscale before training to reduce the diversity within the dataset as much as possible. The real exemplary ocean scene and the jellyfish patch are shown in Fig. 3.

C. SYNTHETIC-IMAGE-BASED FULLY CONVOLUTIONAL NETWORK TRAINING

The fully convolutional network (FCN) is a network used for the semantic segmentation of the image [17]. This network calculates the pixel-wise association probability and generates a contour-like probability map of the labeled data on the image. This network can infer much higher pixel-wise resolution classification results compared to the bounding box inference or simple whole-image classification network. The FCN is also combined with the regression neural network to obtain regression results such as object counting by using the FCN-generated probability map as an input [18]. Despite this usefulness, preparing the label data for the FCN is the most labor-intensive compared to other classification networks. The target objects in the original image should be carefully colored to obtain the boundary region clearly. Other non-target objects should be removed and changed into the background color. At least a few hundred images should be processed to train the simplest FCN. If the target object is a vague jellyfish-like object in the water, the labeling process becomes more difficult.

In this paper, we use the GAN-generated synthetic image to solve the addressed issue. The synthetic jellyfish images generated by the GAN were attached to random points of

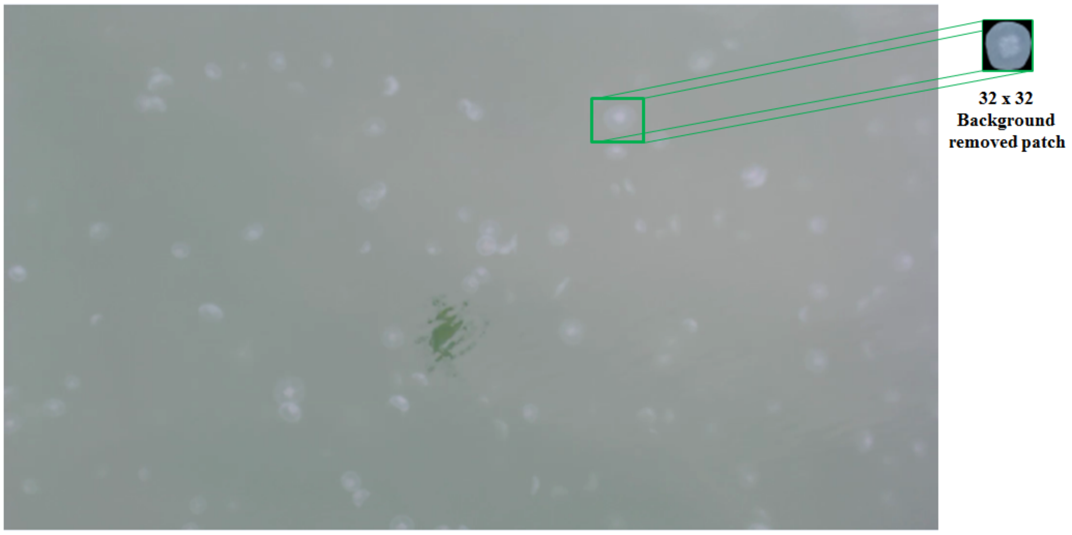

FIGURE 3. The jellyfish blooming real ocean scene image taken by the UAV and background removed image patch used for the training.

the seawater image. As the exact point, size, and contour boundary of the synthetic patches are known, the black-and-white label image can be prepared automatically during the synthetic-scene generating process.

In the Republic of Korea, the jellyfish alert given by the National Institute of Fisheries Science (NIFS) is classified into three levels (excluding 'attention' level which indicates that a new jellyfish polyp colony was found) [19]. The regression class was classified into three labels to mimic this alert system; the 'clear' level indicating the low density of the jellyfish (about 0 to 5 jellyfish), the 'advisory' level which means the medium swarm size (more than 20), and the 'alert' level notifying severe bloom of the swarm (about or more than 100). The exact alert is given by counting the actual numbers within a 10m $\times$ 10m area. However, as the recorded video did not contain flight odometry data, we only mimicked a three-level differentiation of the jellyfish swarm's density and approximate jellyfish numbers. In our experiment, random numbers (1 - 10, 20 - 40, and 70 - 100) of the random jellyfish patches were placed on the blank seawater scene (no jellyfish in the ocean) and 100 synthetic scenes and labels were generated.

The FCN was trained by the synthetic scene for about 1,300 iterations (learning rate of 0.0001 with the decay function drops to near zero at this point). After the pre-trained FCN was prepared, 100 real seawater scenes (real jellyfish exists) were passed to the pre-trained FCN to obtain the probability score map. The collected score map was classified into three density levels based on the original jellyfish number in the original real scene and trained to the regression network. An AlexNet structure with the Euclidean loss and single regression output final node were applied to the regression process (learning rate of 0.001, 6000 iteration). The overall process is shown in Fig. 4.

III. RESULTS AND DISCUSSION

A. SYNTHETIC JELLYFISH IMAGE GENERATION

The synthetic jellyfish images were generated by the proposed GAN architecture. For a comparison, the basic MLP GAN, Deep Convolutional GAN (DCGAN) [20], and Boundary Equilibrium GAN (BEGAN) [21] were chosen. The DCGAN showed good results in various benchmark datasets and its implementation code [22] has been applied to many open-source GAN implementation projects. BEGAN is one of the most cutting-edge, recently published GAN structures famous for its high-resolution generative ability. The open-source implementation of BEGAN [23], [24] was used for the test. The structure of the used GANs is shown in Table 1. A set of hyper-parameters such as the learning rate, optimizer, and batch size were selected and each combination of these parameters was tested based on the exhaustive grid search method [25]. The exhaustive grid search method tests the generated combinations of all parameters within given range to find optimal parameters. The range of each parameter was determined by referring to the published default values of the available codes. All combinations were trained for 100 epochs and generated output of every epoch was monitored to find the optimized value. The optimized parameters are shown in Table 2. The generated image outputs are shown in Fig. 5.

The results showed that the basic MLP GAN could not generate any image; only dot-like figures were generated though many trials were done. A phenomenon called mode collapse, where only identical images are generated at one test batch, was observed during the test using the MLP GAN. The DCGAN showed much better performance. The DCGAN generated jellyfish-like figures. Some image patches showed quite good results; however, the overall texture and brightness were very rough compared to the real images.

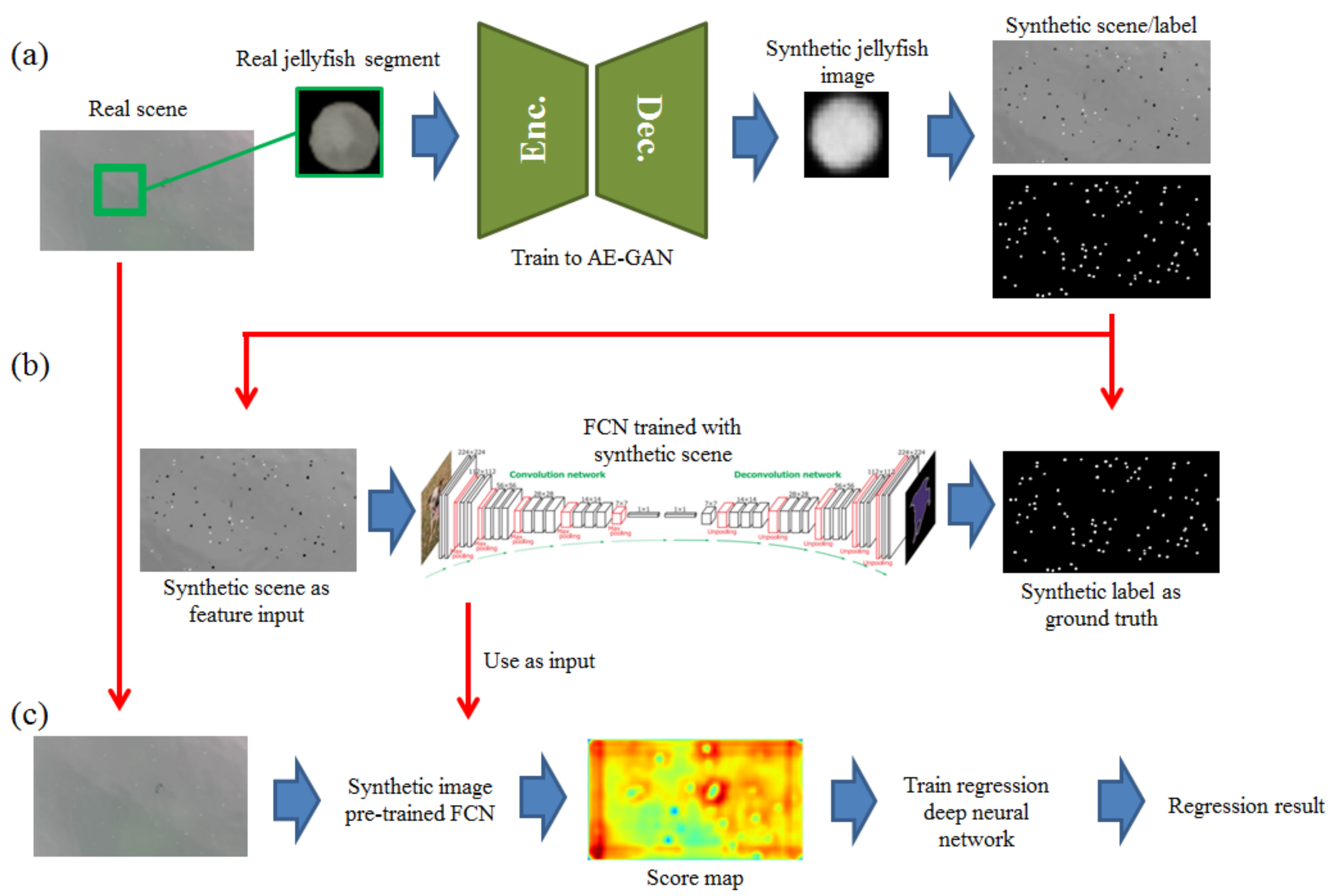
(a)
Real scene
Real jellyfish segment
Enc.
Dec.
Train to AE-GAN
Synthetic jellyfish image
Synthetic scene/label
(b)
FCN trained with synthetic scene
Convolution network
Deconvolution network
Synthetic scene as feature input
Synthetic label as ground truth
Use as input
(c)
Synthetic image pre-trained FCN
Score map
Train regression deep neural network
Regression result

FIGURE 4. Flow diagram of the jellyfish swarm size level regression system. (a) Generating synthetic jellyfish images from the real jellyfish patches, generating an artificial scene. (b) FCN training with the synthetic scene. (c) Regression of the probability score map obtained by processing the real seawater scene into the synthetic-image-trained FCN.

TABLE 1. Used GANs structure information.

Network [a]	Structure [b]
MLP G	$100 \times 150fc - 150 \times 300fc - 300 \times 1024fc$
MLP D	$1024 \times 300fc - 300 \times 150fc - 150 \times 1fc$
DC G	$100 \times 512dc - 512 \times 256dc - 256 \times 128fc - 128 \times 64dc - 64 \times 3dc$
DC D	$3 \times 64c - 64 \times 128c - 128 \times 256c - 256 \times 512c - 512 \times 1c$
BE G	$64 \times (64 \times 32 \times 32)fc - (CIE - CIE - U) \times 4 - CIE - 64 \times 3c$
BE D enc	$3 \times 64c - CIE1 \times 2 - CIE2 \times 2 - CIE2 \times 3 - CIE3 \times 3 - CIE3 \times 4 - CIE4 \times 4 - CIE4 \times 5 - CIE5 \times 5$
BE D dec	$CIE - U - (CIE - CIE - U) \times 3 - CIE - 63 \times 3c$
BE D	BE D enc $-(64 \times 5 \times 32 \times 32) \times 64fc - 64 \times (64 \times 32 \times 32)fc$ - BE D dec

[a] G=Generator structure of the given GANs. D=Discriminator structure of the given GANs. BE D enc= Encoder module of BEGAN discriminator. BE D dec= Decoder module of BEGAN discriminator.
[b] $(n \times m)fc = n$ by m fully connected layer. $(n \times m)dc = n$ by m deconvolution layer. $(n \times m)c = n$ by m convolution layer. $CIE = 64 \times 64$ convolution layer, 2D instance norm-exponential linear unit, and exponential linear unit(ELU) activation layer compound. $CIE(i \times j) = i \times 64$ by $j \times 64$ convolution layer CIE unit. $U =$ upsample layer.

TABLE 2. Training hyper-parameters of the GANs.

	MLP GAN	DCGAN	BEGAN	Proposed
Learning rate	0.0002	0.0002	0.0008	0.001
Batch size	16	16	16	16
Optimizer	Adam	Adam	Adam	RMSprop
Train epoch	100	90	50	70
Train result[a]	×	○	×	○

[a] Indicates whether the GAN was successfully trained or showed mode collapse.

Also, wrong combination of the learned features was observed. Though some part of the generated image resembles the real image, the whole image becomes awkward as the assembled image does not look like a jellyfish. The BEGAN showed the best realities in the texture generation but failed to generate the images and showed mode collapse. Under limited dataset size conditions, suitably training the deep structures of the BEGAN was hard and shallow GANs such as the DCGAN showed the better results. Compared to the previous methods, the proposed AE-combined GAN generated more successful results. The synthetic image still lacked details compared to the original dataset. The inner texture of the jellyfishes was not fully reconstructed. However, the proposed method successfully generated round and smooth elliptical shapes, which is a much better result compared to the conventional methods. For a quantitative comparison, the cosine similarity and mean squared errors (MSE) of the test-batch output of each GAN were compared with the real dataset. Five batch outputs were selected and compared

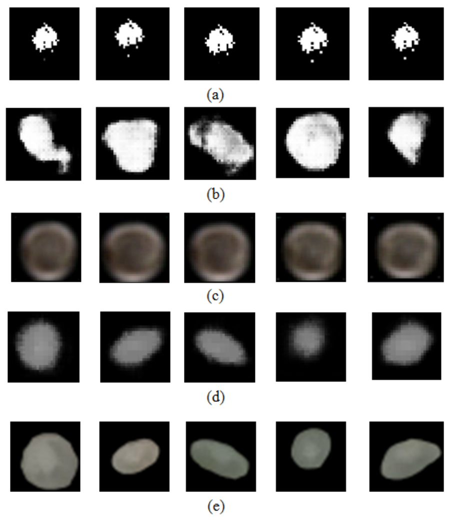

FIGURE 5. Generated image outputs and comparison with the real image. (a) Basic MLP GAN (b) DCGAN (c) BEGAN (d) Proposed AE-combined MLP GAN and (e) Real jellyfish image.

TABLE 3. Mean squared errors and cosine similarity of the generated image compared with the real data.

Method	Top-5 MSE	Top-5 cosine similarity
MLP GAN	13.31 ± 1.46	0.702 ± 0.043
DCGAN	40.81 ± 6.87	0.90 ± 0.057
BEGAN	68.32 ± 3.5	0.91 ± 0.02
Proposed	35.28 ± 1.40	0.94 ± 0.02

with the whole train data. The maximum cosine similarity and the minimum MSE (best cases) value per comparison were collected. The results are summarized in Table 3.

The proposed method showed the second-best MSE result and the best cosine similarity result compared to the other conventional methods. The MLP GAN showed an abnormally low MSE error because of the matching between black backgrounds of the original images. The DCGAN showed good results in both cosine similarity and MSE as it generated jellyfish-like figures in despite of the rough texture. The BEGAN showed good cosine similarity score due to small difference in the generated image pool. This is due to the fact that as the best cases are collected, if all the images look similar due to the mode-collapse, the similarity value tends to be biased despite the bad image quality.

The single jellyfish patch is only a 32×32 sized object. The image of the jellyfish was taken with the gimbal-mounted camera of a quadcopter platform. Thus, the image was very small and blurred compared to the other benchmark datasets. We assume that the DCGAN-based modern GANs failed because it could not find the appropriate features to learn from the given jellyfish datasets. Excessively simple shapes of the train images caused bad results in complex structured neural networks. The merits of the AEs can be found in these situations. The AEs can be easily trained; rather small numbers of data are enough to train AEs compared to the GANs. Though the GAN generates a relatively poor output, the AE can project the generated vectors to an image, which helps in increasing the quality of the output images.

The usage of the AEs in the GAN also brings other merits. The AE plays key roles in image generation in the proposed system. As the AE can reconstruct more realistic images from the encoded vectors, the quality of the overall system increases even though the GAN generates the vectors with similar qualities. The output of the GAN is unpredictable before training. However, the proposed system's output is ensured if the AE-generated images have acceptable qualities. If the benchmark dataset with many well-qualified images is used, other cutting-edge GAN structures are expected to show competitive performance with the proposed system. However, collecting and labeling data is a costly job, and obtaining sufficient data is sometimes impossible. The proposed method has a comparative advantage over previous methods, as it requires a small number of datasets to generate the data. Further, the jellyfish image differs based on the recording drone's flying altitude and regions. The requirement of less data indicates that the proposed system can generate a custom image dataset mimicking the recorded condition of different regions more easily as compared to other systems.

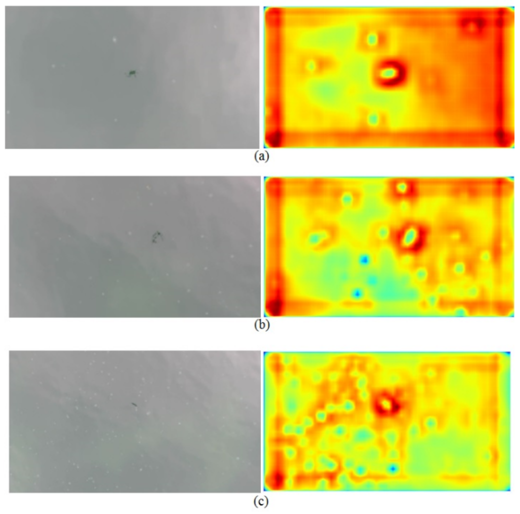

FIGURE 6. Real ocean scene and probability map generated by the synthetic image pre-trained FCN. The pale white dots in the ocean scene images are real jellyfishes. (a) Low level of jellyfish density (clear). (b) Medium level of jellyfish density (advisory). (c) High level of jellyfish density (alert).

B. JELLYFISH SWARM DENSITY LEVEL ESTIMATION WITH A SYNTHETIC-IMAGE-TRAINED NETWORK

The FCN-based regression method is very useful in estimating the distribution of the target objects in the scene with very low computational cost, as it does not require classification (neural network feed-forward computation) of every candidate figure in the image. In this paper, we tested whether the FCN trained with the synthetic ocean scene can generate a good probability score map even though the realistic images are given. The results are shown in Fig. 6.

Hundred real ocean images were classified into three classes based on the number of the jellyfish in the image.

The three level criteria was used to mimic actual jellyfish swarm alert criteria explained in Section 2.3.

The accuracy of the regression network was measured after processing test datasets, which was not used during the training phase. The results showed an accuracy of 83.8%; total 52 correct classifications were obtained among 62 test data. The previous research on classifying jellyfish image patch with the deep-learning architecture showed 80.5% accuracy [6]. Though the final target is slightly different, the obtained accuracy gave acceptable results compared to the previous research. In the case of the number regression, the counting method using the classification of every object candidate in the image shows a relatively low accuracy even though the classification accuracy is high (over 80%). If there are number of objects in the image, even a very accurately trained classifier can output a high error rate due to multiple classification operations in a single image. The FCN combined with the regression networks shows better performance in this case as it can locate possible targets at a single inference.

For further comparison, the state-of-the-art FCN network [17] combined with the regression neural network, a well-known combination for the distribution estimation task, was used. The same real jellyfish bloom scene images were manually labeled and trained using the FCN network; a well-known method to estimate the distribution label with the neural networks (the processes shown in Fig. 4(b) and Fig. 4(c)). The results were compared with the proposed synthetic image-based results. The real image trained regression network showed the accuracy of 71.4%. The proposed synthetic jellyfish-based method showed better results even though the real jellyfish scene was labeled and given as a training input to the network. Though labeled by a human manually, the labeled data is not perfect and even the pixel-wise errors can confuse the FCN during the training phase. This phenomenon is more noticeable when the size of the data increases and the data is distributed to many workers as the labeling criteria may differ among workers. The synthetic data is much freer on this issue as near-perfect labels can be generated. The amount of the data also plays an important role in the accuracy of the deep neural networks. The synthetic data-based method can easily increase the number of the labeled data with much less cost compared to the real data which requires human labor to be labeled.

The proposed synthetic-scene-based method can gradually aid overall process by automatically generating labels and features for training the FCNs. Moreover, the dots found in the probability map in Fig. 6 are well-matched with the real jellyfish distribution. These results and the high accuracy with real data show that the synthetic jellyfish image is reasonably similar to the real jellyfish in view of the neural networks.

IV. CONCLUSIONS AND FUTURE WORK

This paper presented a newly developed technique using the AE and GAN to solve the image generation issue in a small-numbered, simple-featured image dataset. The jellyfish was used as an example to show that the proposed method successfully works in cases that cannot be solved by using conventional GANs. The training of the deep-learning-based jellyfish swarm density level estimation with the synthetic image was shown as an application of the proposed method to realistic problems. The proposed method increased the stability of the GAN by adding the pre-trained autoencoder to make a boundary to the GAN output image, and the increase in stability broadens its application not only to situations where there is little data but also to other harsh training conditions. Though the training dataset is sufficient, the GAN has its innate instability due to usage of the Kullback-Leibler divergence or Jensen-Shannon divergence as pointed out in the parallel line learning problem in [26]. The stability issue solvers including the proposed method help GAN's training by decreasing failure rate and tuning time.

As future work, we are considering the upgrade of the autoencoder's image restoration ability by applying more sophisticated autoencoders, such as the deep convolutional autoencoder. The autoencoder-generated vectors, which are used as a training input to the GAN, are rather closer to a sequence than an image. We are considering the application of sequence-specific GANs to increase the GAN's functionalities. Further application of the proposed GAN structure to other image datasets [27], is also under consideration. Experimental trials helped acquiring results are documented in [28].

REFERENCES

[1] M. N. Arai, *A Functional Biology of Scyphozoa*. London, U.K.: Chapman & Hall, 1997, p. 316.

[2] J. E. Purcell and M. N. Arai, "Interactions of pelagic cnidarians and ctenophores with fish: A review," *Hydrobiologia*, vol. 451, pp. 27–44, May 2001.

[3] *Hazardous Marine Life Jellyfish Damage Prevention Research Projects*, Ministry Land, Infrastruct. Transp., Sejong, South Korea, 2012.

[4] B.-T. Kim, K.-H. Eom, I.-S. Han, and H.-J. Park, "An analysis of the impact of climatic elements on the jellyfish Blooms," *J. Fisheries Mar. Educ.*, vol. 27, no. 6, pp. 1755–1763, 2015.

[5] H. Kim, D. Kim, H. Kim, J.-U. Shin, and H. Myung, "An extended any-angle path planning algorithm for maintaining formation of multi-agent jellyfish elimination robot system," *Int. J. Control, Autom. Syst.*, vol. 14, no. 2, pp. 598–607, 2016.

[6] H. Kim *et al.*, "Image-based monitoring of jellyfish using deep learning architecture," *IEEE Sensors J.*, vol. 16, no. 8, pp. 2215–2216, Apr. 2016.

[7] K. Simonyan and A. Zisserman. (Apr. 2015). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[8] I. J. Goodfellow *et al.* (Jun. 2014). "Generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1406.2661

[9] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. (Nov. 2016). "Unrolled generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1611.02163

[10] *GAN Implementation*. Accessed: Nov. 2, 2017. [Online]. Available: https://github.com/ckmarkoh/GAN-tensorflow/

[11] Y. Jia *et al.* (Jun. 2014). "Caffe: Convolutional architecture for fast feature embedding." [Online]. Available: https://arxiv.org/abs/1408.5093

[12] *Caffe MNIST Autoencoder*. Accessed: Nov. 2, 2017. [Online]. Available: https://github.com/BVLC/caffe/tree/master/examples/mnist/

[13] X. Mao, Q. Li, H. Xie, R. Lau, Z. Wang, and S. P. Smolley. (Nov. 2016). "Least squares generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1611.04076

[14] A. B. L. Larsen, S. Sønderby, H. Larochelle, and O. Winther. (Dec. 2015). "Autoencoding beyond pixels using a learned similarity metric." [Online]. Available: https://arxiv.org/abs/1512.09300
[15] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. (Nov. 2015). "Adversarial autoencoders." [Online]. Available: https://arxiv.org/abs/1511.05644
[16] *Used Jellyfish Image Dataset*. Accessed: Nov. 2, 2017. [Online]. Available: http://urobot.synology.me/data/evgan_jellyfish_img.zip/
[17] E. Shelhamer, J. Long, and T. Darrell. (May 2017). "Fully convolutional networks for semantic segmentation." [Online]. Available: https://arxiv.org/abs/1605.06211
[18] S. W. Chen *et al.*, "Counting apples and oranges with deep learning: A data-driven approach," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 771–788, Apr. 2017.
[19] *Fishery Damage due to Massive Jellyfish Occurrence*, (in Korean), Ministry Oceans Fisheries, Sejong, South Korea, 2013.
[20] A. Radford, L. Metz, and S. Chintala. (Nov. 2015). "Unsupervised representation learning with deep convolutional generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1511.06434
[21] D. Berthelot, T. Schumm, and L. Metz. (Mar. 2017). "BEGAN: Boundary equilibrium generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1703.10717
[22] *DCGAN Implementation*. Accessed: Nov. 2, 2017. [Online]. Available: https://github.com/carpedm20/DCGAN-tensorflow/
[23] *BEGAN Implementation*. Accessed: Nov. 2, 2017. [Online]. Available: https://github.com/TuXiaokang/BEGAN.PyTorch/
[24] *BEGAN Implementation*. Accessed: Nov. 2, 2017. [Online]. Available: https://github.com/carpedm20/BEGAN-tensorflow/
[25] J. Zhao, M. Mathieu, and Y. LeCun. (Mar. 2017). "Energy-based generative adversarial network." [Online]. Available: https://arxiv.org/abs/1609.03126
[26] M. Arjovsky, S. Chintala, and L. Bottou. (Dec. 2017). "Wasserstein GAN." [Online]. Available: https://arxiv.org/abs/1701.07875
[27] K. Kim, H. Kim, H. Lim, and H. Myung, "A low cost/low power open source sensor system for automated tuberculosis drug susceptibility testing," *Sensors*, vol. 16, no. 6, p. 942, 2016.
[28] K. Kim, "Autoencoder-combined generative adversarial networks for synthetic image data generation and application for the oceanic environments," (in Korean), M.S. thesis, Korea Adv. Inst. Sci. Technol., Daejeon, South Korea, 2018.

KYUKWANG KIM received the B.S. degree in bio and brain engineering and the M.S. degree with the Robotics Program from the Korea Advanced Institute of Science and Technology, where he is currently pursuing the Ph.D. degree in biological sciences. His research includes bioinformatics, biosensors, environmental robotics, and machine learning.

HYUN MYUNG received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1992, 1994, and 1998, respectively. He was a Senior Researcher with the Electronics and Telecommunications Research Institute, Daejeon, from 1998 to 2002, the CTO and the Director with the Digital Contents Research Laboratory, Emersys Corporation, Daejeon, from 2002 to 2003, and a Principle Researcher with the Samsung Advanced Institute of Technology, Yongin, South Korea, from 2003 to 2008. Since 2008, he has been a Professor with the Department of Civil and Environmental Engineering, KAIST, where he is currently the Head of the Robotics Program. His current research interests include structural health monitoring using robotics, soft computing, simultaneous localization and mapping, robot navigation, machine learning, deep learning, and swarm robot.